

Nagarajan, Narenraju (2026) *Physics-informed representation learning for gravitational wave discovery*. PhD thesis.

https://theses.gla.ac.uk/85916/

Copyright and moral rights for this work are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This work cannot be reproduced or quoted extensively from without first obtaining permission from the author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Enlighten: Theses
https://theses.gla.ac.uk/
research-enlighten@glasgow.ac.uk

Physics-informed Representation Learning for Gravitational Wave Discovery

Narenraju Nagarajan

Submitted in fulfillment of the requirements for
the Degree of Doctor of Philosophy

School of Physics and Astronomy
College of Science and Engineering
University of Glasgow



September 2025

Worldbuilding

"Exactly!" said Deep Thought. "So once you do know what the question actually is, you'll know what the answer means."

The Hitchhiker's Guide to the Galaxy
Douglas Adams

Before heading straight into the what and how of my work, we must address the most important aspect of this thesis, the *Reader*. The abstract is arranged as follows: [i] what exists in the observable universe of this thesis [ii] who this thesis is written for, the unresolved challenges in their field, and why reading it may benefit *their* work. [iii] what tangible insights the thesis offers and the approach taken to develop them.

The Observable Universe: A Reader who is capable of observing the ripples of spacetime caused by the inspiral and collision of massive, compact objects known as gravitational-waves (GWs). We assume that they have the means to begin comprehending these phenomena rigorously, quantify them, and finally analyse them.

The Reader: This thesis is intended for three key research communities, but is open to any inquisitive mind. The first community is the group of researchers working on matched-filtering to detect and statistically rank the significance of GW events. They have overcome several challenges posed by the detection problem, developed the statistical framework required to be confident in our detection, and successfully detected 218 official GW candidates as of their most recent catalogue [1].

Secondly, this is for the community of interdisciplinary researchers dedicated to the application of machine-learning (ML) techniques for the detection and characterisation of GWs. This is a budding field tackling the complexities of the black-box nature of ML while making it trustworthy and robust for a field as precise as GW science. Finally, this thesis is also for the application agnostic ML researcher who treats GWs as nothing more than a time series waiting to be classified.

The Challenge: Although matched filtering has had significant success in the past decade [2, 3, 4, 5, 1], there are known issues that exist in their formalism which makes them non-optimal for the GW detection problem [6]. Can ML be used to support matched filtering pipelines in GW detection by complementing them when they deviate significantly from their optimal properties?

Although building ML-based solutions for complex problems has become relatively simple over the years, their issues are notoriously difficult to identify or mitigate. If we want them to perform as close as possible to algorithms with theoretical guarantees, it requires considerable effort and bespoke

fine-tuning of the learning procedure. What do we need to understand to help up close this gap and improve trustworthiness?

The promise of ML is the ability to create a data-driven algorithm capable of adapting to unseen situations without attenuation of performance. With careful tuning, ML is thought to have the potential to rival traditional algorithms in capability while accelerating scientific discovery. However, in practice, there are well known caveats and real ML-based scientific discovery is rarely seen. Can we prove that ML is indeed capable of contributing novelty to science via discovery?

Value to the Reader: This thesis directly tackles the above challenges and provides clear evidence for its claims. Chapter 4 provides significant insight into the challenges that exist beyond the simple framing of the detection problem in ML. We directly tackle and mitigate these issues to produce a framework capable of filling the gaps left by matched filtering. We empirically prove that incorporating domain knowledge into the architecture and learning process of a ML model improves its interpretability and performance. To complement this insight, we delve deep into the exact mechanisms that lead to large inflections in performance and provide a roadmap of remaining issues. Chapter 5 applies the framework developed in chapter 4 to a real-world detection scenario. It provides strong evidence for the scientific capability of our framework by detecting most of the events in existing GW catalogues within its training distribution. Furthermore, we discover 5 new GW events and characterise them to showcase the capability of ML in contributing to present and future production-level GW searches.

To support pedagogy and reproducibility, we provide publicly available supplementary code for all introductory experiments and figures in Chapters 1–3, as well as visualization tools for the newly discovered events in Chapter 5. In particular, Chapter 3 builds ML concepts progressively—from a single neuron to a full network capable of detecting weak, noisy signals. Readers are encouraged to experiment with, stress-test, and extend the models, with contributions welcomed via the project's dedicated GitHub repository: https://github.com/nnarenraju/phd-thesis-supplement.

The Investigation: This thesis advances GW detection by addressing the long-standing challenge of bias in ML pipelines and demonstrating how physics-informed representation learning and inductive biases can mitigate these bias and provide detection efficiencies comparable to matched filtering. We introduce Sage, a detection framework that incorporates our research to simplify preprocessing, enhance robustness to non-Gaussian noise, and achieves superior sensitivity compared to a benchmark in matched filtering and prior ML approaches to the GW detection problem. By explicitly targeting bias sources and leveraging architecture-informed design, Sage delivers efficient, scalable, and trustworthy detection—validated through real data discoveries. These results highlight how embedding physical insight into machine learning not only boosts accuracy and efficiency but also provides a path toward reliable, next-generation GW searches.

A flowchart of the entire thesis is shown in Fig. 1 to provide a bird's eye view of the contents. I hope you, the Reader, find at least one insight within this thesis that gives you an *"aha!"* moment.

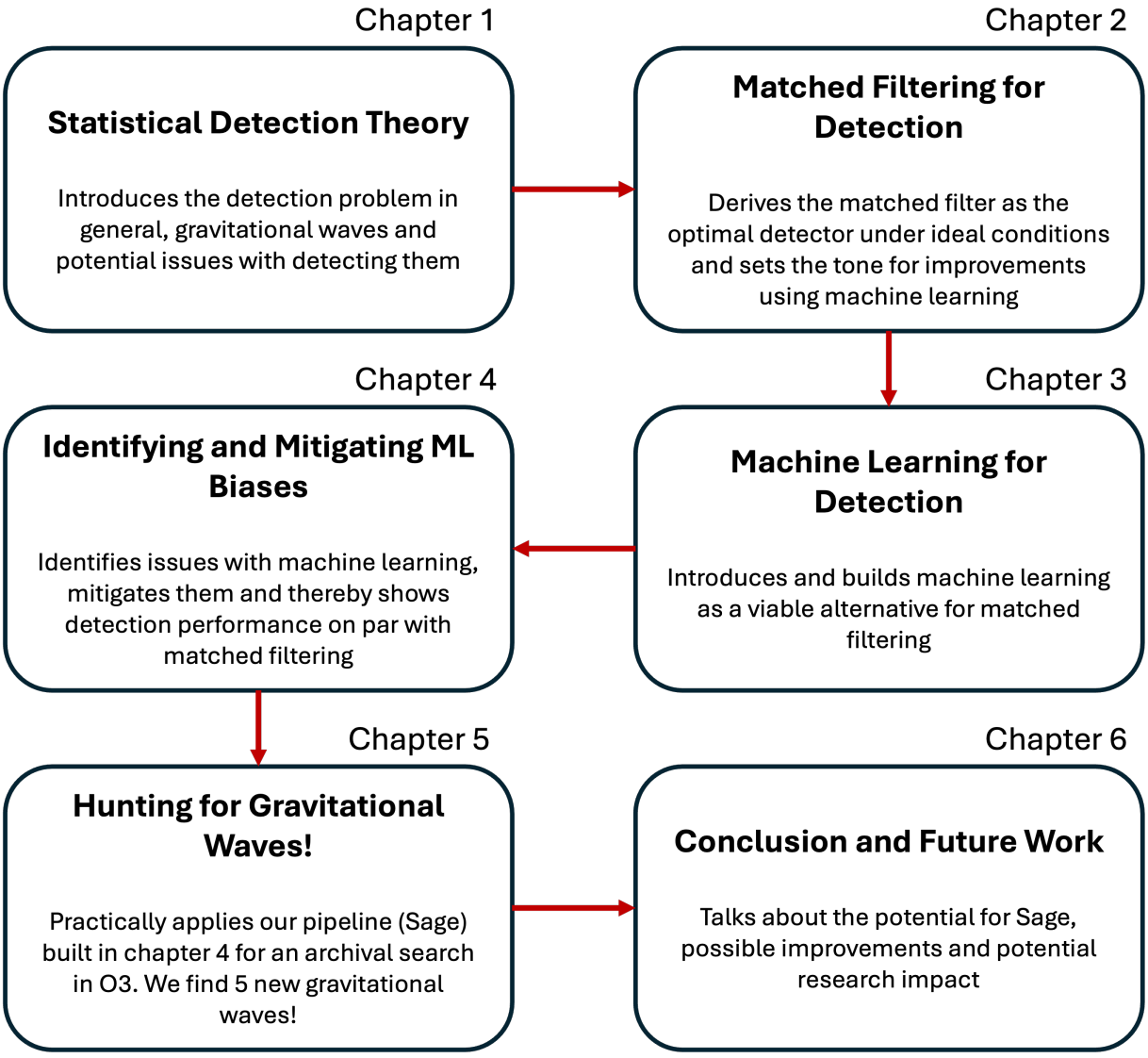


Figure 1: Flowchart of the thesis. Chapter 1-3 are introductory chapters that motivates the work done in chapter 4 to build a robust ML pipeline for GW detection known as `Sage`. Chapter 5 applies `Sage` to search for GWs in real data and proves its real world applicability by finding new GWs. The final chapter shows what `Sage` could potentially be and paints a picture for the future of ML-based GW discovery.

Contents

Worldbuilding i

List of Figures vi

List of Tables ix

Acknowledgments xi

I Motivational and Mathematical Prelude 1

1 Statistical Detection Theory 2
- 1.1 The Detection Problem . . . 2
- 1.2 Neyman-Pearson Theorem . . . 7
  - 1.2.1 Derivation of the Neyman–Pearson Lemma . . . 8
- 1.3 The Gravitational Wave Detection Problem . . . 12
  - 1.3.1 Gravitational Waves . . . 12
  - 1.3.2 The LIGO-Virgo-KAGRA Detectors . . . 12
  - 1.3.3 Waveform Morphology . . . 14
  - 1.3.4 Properties of Detector Data . . . 20
- 1.4 Takeaways and Open Questions . . . 30
  - 1.4.1 Key Takeaways . . . 30
  - 1.4.2 Open Questions . . . 30

2 Matched Filtering for Detection 31
- 2.1 Model Comparison . . . 31
- 2.2 Neyman-Pearson Detector and the Matched Filter . . . 33
- 2.3 Matched Filtering for GW Detection . . . 35
- 2.4 Data Preprocessing . . . 38
- 2.5 Template Bank Construction . . . 42
- 2.6 Visualising the signal-to-noise ratio (SNR) Structure for GW150914 . . . 46
- 2.7 False Alarm Rate Estimation . . . 47
- 2.8 Signal Consistency Tests and Vetoing . . . 49
- 2.9 Handling Glitches and Non-Gaussian Noise . . . 50
- 2.10 Limitations and Bottlenecks . . . 50
- 2.11 Summary and Open Questions . . . 54

3 Machine Learning for Detection 55
- 3.1 The Promise of Deep Learning . . . 55
- 3.2 Representation Learning . . . 56
- 3.3 Emulating a Neuron . . . 57
- 3.4 Emulating a Network of Neurons . . . 61
- 3.5 Loss Functions . . . 63

3.6 Stochastic Gradient Descent . . . 68
3.7 Back Propagation . . . 71
3.8 Training Strategies . . . 72
3.9 Function Space of Neural Networks . . . 75
3.9.1 What can a single neuron represent? . . . 75
3.9.2 What can *n* neurons represent? . . . 77
3.9.3 What can multiple layers do? . . . 79
3.10 Representational Capacity . . . 81
3.10.1 Effect of Layer Width . . . 81
3.10.2 Effect of Network Depth . . . 81
3.10.3 Effects of Different Activation Functions . . . 82
3.10.4 Parameter Count vs Effective Capacity . . . 85
3.10.5 Memorisation and Overfitting . . . 85
3.10.6 Exploring the vanishing and exploding gradients problem . . . 86
3.10.7 Data-Limited vs Model-Limited Regimes . . . 90
3.10.8 Computational Cost of Representational Capacity . . . 91
3.11 Inductive Bias . . . 92
3.11.1 Equivariance . . . 93
3.11.2 Invariance . . . 98
3.11.3 Attention Mechanism . . . 100
3.11.4 Implicit Bias of Neural Networks to Learn Simple Functions . . . 103
3.11.5 What *is* Data Augmentation? . . . 104
3.12 Detecting a Simple Time Series in Gaussian Noise . . . 106
3.13 Machine Learning for Gravitational Wave Detection . . . 107
3.14 Takeaways and Open Questions . . . 111

II The Power of Inductive Biases 112

4 Identifying and Mitigating Machine Learning Biases 113
4.1 Introduction . . . 114
4.1.1 Matched Filtering for GW Detection . . . 115
4.1.2 Machine Learning for GW Detection . . . 116
4.1.3 Acknowledging Potential Problems of Machine Learning Models . . . 118
4.1.4 Key Contributions to the Field . . . 127
4.2 Methods . . . 128
4.2.1 Testing Dataset Parameter Distributions . . . 128
4.2.2 Training Dataset Parameter Distributions . . . 132
4.2.3 Data Generation . . . 136
4.2.4 Data Augmentation . . . 137
4.2.5 Data Transformation . . . 138
4.2.6 Optimisation . . . 141
4.2.7 Network Architecture . . . 142
4.3 Results and Discussion . . . 148
4.3.1 Comparison with Matched Filtering . . . 150
4.3.2 Comparison with Other ML Pipelines . . . 164

4.3.3 Ablation Study on SAGE . . . 168
4.4 Running Sage on Live Data . . . 171
4.5 Limitations of SAGE . . . 172
4.6 Conclusion . . . 173
4.7 Appendix A: Testing Dataset Parameter Distributions . . . 174
4.8 Appendix B: Sage-Broad Detection Sensitivity . . . 174
4.9 Appendix C: Network Behaviour for Different Glitch Types . . . 175
4.10 Appendix D: Parameter Point Estimate . . . 175

III Gravitational Wave Search in O3 & Closing Remarks 180

5 Hunting for Gravitational Waves! 181
5.1 Introduction . . . 182
5.2 Designing the Search . . . 183
5.2.1 O3 Search Data . . . 183
5.2.2 Data Acquisition and Creating Search Samples . . . 185
5.2.3 SAGE for Offline Search . . . 186
5.2.4 Background Estimation . . . 189
5.2.5 Estimating `p_astro` . . . 194
5.3 Catalogue of BBH events in O3 . . . 201
5.3.1 Expected to Re-discover . . . 201
5.3.2 Opening the Box . . . 202
5.4 Catalogue Coincidence . . . 204
5.4.1 Coincidence with GWTC, Subthreshold and Community Catalogues . . . 204
5.4.2 Missed Events . . . 207
5.5 Discovering New Events in O3! . . . 210
5.5.1 Events with False Alarm Rate < 2 per year . . . 210
5.5.2 Parameter Estimation . . . 212
5.5.3 Why did we find new events? . . . 223
5.6 Conclusion . . . 227

6 Contributions & Future Extension 229
6.1 Thesis Summary . . . 229
6.2 Key Contributions . . . 230
6.3 (near-) Future Extensions . . . 231
6.3.1 Predicting the Odds Ratio . . . 231
6.3.2 Equivariant Machine Learning Detection . . . 232
6.3.3 Uncertainty Quantification for the Ranking Statistic . . . 233
6.3.4 On-the-fly Background Estimation . . . 234
6.3.5 Learning Long Term Dependencies . . . 234
6.3.6 OOD Detection . . . 234
6.3.7 Sub-Nyquist Waveform Compression . . . 235
6.3.8 Big Picture and Research Impact . . . 235
6.4 Coda . . . 238

List of Figures

1 Abstract: Flowchart of the thesis contents iii

1.1 Chapter 1: Histograms of null and alternate hypothesis (varying signal amplitude) . . 4
1.2 Chapter 1: Histograms of null and alternate hypothesis (varying decision boundary) . 7
1.3 Chapter 1: 2D histogram of component masses with contour lines for $\mathcal{M}_c$ and q . . 16
1.4 Chapter 1: Effect of changing GW parameters on time domain waveform 18
1.5 Chapter 1: Effect of changing sky position on time domain waveform 21
1.6 Chapter 1: Sampling a sinusoid above and below the Nyquist limit 23
1.7 Chapter 1: Fourier transform of single and multi-frequency sinusoid 24
1.8 Chapter 1: Comparison of different power spectral densities 27
1.9 Chapter 1: Q-transform of the glitch on GW170817 29

2.1 Chapter 2: Raw strain data around GW150914 from H1 and L1 38
2.2 Chapter 2: Comparison of Fourier spectra using rectangular/Tukey window 39
2.3 Chapter 2: Power spectral density around GW150914 41
2.4 Chapter 2: Whitened strain data around GW150914 42
2.5 Chapter 2: Whitened and bandpassed strain around GW150914 43
2.6 Chapter 2: 2D histogram of peak matched filter SNR in (m_1, m_2) for GW150914 . . 48
2.7 Chapter 2: Q-transform of a whistle glitch in O3 data 51
2.8 Chapter 2: 2D histogram of peak matched filter SNR in (m_1, m_2) for whistle glitch . 52

3.1 Chapter 3: Flowchart comparing different types of detection systems 57
3.2 Chapter 3: Comparing Cartesian and Polar coordinates for representation learning . . 58
3.3 Chapter 3: Depiction of a single neutron with input and output 59
3.4 Chapter 3: Training a single neuron classifer to showcase Neyman-Pearson decision . 60
3.5 Chapter 3: Cartoon of a fully connected 3-layer neural network 62
3.6 Chapter 3: Decision surface learn by a two-layer neural network 64
3.7 Chapter 3: Schematic representation of possible sub-network topologies. 65
3.8 Chapter 3: Cartoon depicting the loss landscape 68
3.9 Chapter 3: Cartoon of a 3D parabolic loss manifold 69
3.10 Chapter 3: Subset of the function space of 1 tanh neuron 76
3.11 Chapter 3: Subset of the function space of a 1-layer 2 tanh neuron network 78
3.12 Chapter 3: Function space of a 1-layer 2 tanh neuron network close to target func . . 79
3.13 Chapter 3: Subset of the function space of a 2-layer 2 tanh neuron network 80
3.14 Chapter 3: Effect of varying number of hidden neurons in 1-later network 82
3.15 Chapter 3: Effect of varying number of hidden layers in a neural network 83

3.16 Chapter 3: Effect of varying activation functions on a target function . . . 84
3.17 Chapter 3: Effect of varying network architecture on MSE loss . . . 86
3.18 Chapter 3: Loss curve for an overfitting network . . . 87
3.19 Chapter 3: Comparing true vs predicted function for an overfitting network . . . 87
3.20 Chapter 3: Comparing different network initialisation strategies . . . 88
3.21 Chapter 3: Comparing ReLU and SiLU activations and their derivatives . . . 89
3.22 Chapter 3: Layer-wise map of dead neurons when using ReLU activations . . . 90
3.23 Chapter 3: Comparison of data-limited and model-limited training regimes . . . 91
3.24 Chapter 3: MSE error vs trainable parameters for a fully connected network . . . 92
3.25 Chapter 3: Feature maps in a CNN representing equivariance . . . 97
3.26 Chapter 3: Input images with a square placed at different positions . . . 99
3.27 Chapter 3: Activation values showing translation invariance . . . 99
3.28 Chapter 3: Output of an attention mechanism in a simple network . . . 102
3.29 Chapter 3: 100-degree polynomial fit vs 100-parameter network . . . 104
3.30 Chapter 3: Showing decent learnt translation invariance . . . 106
3.31 Chapter 3: Showing badly learnt translation invariance . . . 106
3.32 Chapter 3: Bessel function in white Gaussian noise . . . 107
3.33 Chapter 3: Histogram of ranking statistics by network for signal and noise classes . . . 108
3.34 Chapter 3: Literature survey of ML applied to GW detection . . . 109

4.1 Chapter 4: Diagram of supervised training procedure . . . 117
4.2 Chapter 4: Strain vs time plot showing signal morphology variations . . . 122
4.3 Chapter 4: 2D (m_1, m_2) space showing lines of constant $\mathcal{M}_c$ . . . 123
4.4 Chapter 4: ChatGPT exhibiting bias due to limited sample representation-1 . . . 124
4.5 Chapter 4: ChatGPT exhibiting bias due to limited sample representation-2 . . . 125
4.6 Chapter 4: ChatGPT exhibiting bias due to limited sample representation-3 . . . 126
4.7 Chapter 4: Flowchart of the Sage methodology . . . 129
4.8 Chapter 4: Power spectral densities in the training and testing datasets . . . 130
4.9 Chapter 4: 2D histogram of the joint distribution $\mathrm{P}(m_1, m_2)$ from $U(\tau_0, \tau_3)$ . . . 134
4.10 Chapter 4: Time spans of training, validation and testing datasets . . . 134
4.11 Chapter 4: Frequency vs time curves of GWs generated from the testing dataset . . . 139
4.12 Chapter 4: Application of multirate sampling on IMRPhenomXPHM template . . . 140
4.13 Chapter 4: Diagram of multiscale frontend feature extractor . . . 144
4.14 Chapter 4: Diagram of CBAM mechanism for attention . . . 146
4.15 Chapter 4: Recap of Sage methodology with biases addressed . . . 147
4.16 Chapter 4: Sensitive distance vs FAR for different Sage runs and PyCBC . . . 151
4.17 Chapter 4: 1D chirp mass histograms of Sage runs at FAR of 1 per month . . . 152
4.18 Chapter 4: Sensitive distance vs FAR for Sage and PyCBC with error bar . . . 154
4.19 Chapter 4: ROC curves for Sage vs PyCBC . . . 155

4.20 Chapter 4: Histograms of Sage vs PyCBC detections at FAR 1/per month 156
4.21 Chapter 4: BCE loss vs training epoch for different validation datasets 158
4.22 Chapter 4: Log FAR vs glitch SNR for Sage runs and PyCBC 159
4.23 Chapter 4: Normalised histograms of ranking statistics generated by Sage runs 163
4.24 Chapter 4: Histograms of Sage vs AresGW detections at FAR 1/per month 165
4.25 Chapter 4: Histograms of Sage vs AFrame detections at FAR 1/per month 167
4.26 Chapter 4: Sensitive distance vs FAR for Sage, PyCBC and other ML pipelines 168
4.27 Chapter 4: Chirp mass histograms of Sage vs ablation studies 170
4.28 Chapter 4: 1D histograms of GW parameters in testing dataset 176
4.29 Chapter 4: 1D histogram of detected GW parameters for Sage vs PyCBC 177
4.30 Chapter 4: Glitch triggers from different glitch types for Sage vs PyCBC 178
4.31 Chapter 4: Actual value of $\mathscr{M}_c$ and t_c vs predicted values 179

5.1 Chapter 5: Power spectral densities for H1, L1 and V1 in O3 184
5.2 Chapter 5: Histogram of ranking stats for signal in (H1, L1) vs signal in (H1) 188
5.3 Chapter 5: Histogram of ranking stats for same signal in (H1, L1) vs different signals . 189
5.4 Chapter 5: Background estimation for H1, L1, and V1 191
5.5 Chapter 5: Background estimation for coloured Gaussian noise vs glitches 193
5.6 Chapter 5: KDEs of signal and noise likelihoods 200
5.7 Chapter 5: Posterior distributions of GWTC-3 BBHs in O3a 202
5.8 Chapter 5: Posterior distributions of GWTC-3 BBHs in O3b 203
5.9 Chapter 5: Zerolag ranking stat distributions for HV, LV and HL 204
5.10 Chapter 5: Posterior distributions of missed in-distribution events in GWTC-3 Ø3 . . 208
5.11 Chapter 5: Posterior distributions of missed in-distribution events in external catalogues 208
5.12 Chapter 5: Q-transform of candidate SC190617_193742 - a whistle glitch 211
5.13 Chapter 5: PE summary of Sage candidate SC190421_153038 - likely glitch 213
5.14 Chapter 5: PE summary of Sage candidate SC190502_113616 - likely glitch 214
5.15 Chapter 5: PE summary of Sage candidate SC190522_130915 - claimed GW signal! . . 216
5.16 Chapter 5: PE summary of Sage candidate SC190902_232241 - claimed GW signal! . . 217
5.17 Chapter 5: 1D Spin distributions for SC190902_232241 218
5.18 Chapter 5: PE summary of Sage candidate SC190929_095627 - claimed GW signal! . . 219
5.19 Chapter 5: PE summary of Sage candidate SC191105_051753 - likely glitch 220
5.20 Chapter 5: PE summary of Sage candidate SC200103_214330 - claimed GW signal! . . 221
5.21 Chapter 5: PE summary of Sage candidate SC200205_085013 - claimed GW signal! . . 222
5.22 Chapter 5: PE summary of Sage candidate SC190623_192541 - subthreshold candidate 224
5.23 Chapter 5: Q-transforms for all new events 225
5.24 Chapter 5: Q-transforms for all other candidates 226

6.1 Chapter 6: Schematic of a future potential Sage pipeline 236

List of Tables

4.1 Chapter 4: Summary of testing dataset GW parameter space 128
4.2 Chapter 4: Summary of Sage runs in the results section 148
4.3 Chapter 4: GW parameters made for testing Sage ranking statistics 161
4.4 Chapter 4: Summary of run configs used to test Sage ranking statistic distributions . 162
4.5 Chapter 4: Summary of sensitive distance and number of signals for pipelines 168

5.1 Chapter 5: In-distribution events found by Sage in GWTC-3 205
5.2 Chapter 5: Out-of-distribution events found by Sage in GWTC-3 206
5.3 Chapter 5: Sub-threshold event by GstLAL found by Sage 207
5.4 Chapter 5: New candidate events found by Sage in O3 211
5.5 Chapter 5: New claimed Sage events in O3 at FAR < 1/year! 218

Acknowledgments

I would like to express my deepest gratitude to everyone who helped me through my PhD journey. I could write an entire thesis filled with all the things you have done for me, but I hope I do you justice within this short acknowledgements section.

To my parents: There is no possible way to properly summarise everything you have given me. Both of you are the sole reason I am who I am. Thank you for giving me the freedom to question anything, for providing unconditional love even when I did not deserve it, unwavering support when I was weak and absolute confidence when I doubted myself. As a child, seeing you two sit at home and actively learn made learning feel like a natural extension of life, rather than something I should resist. Thank you for nurturing my childlike nature and for always giving me the choice to be whomever I wanted to be. Your upbringing allowed me to be the researcher I am today. I hope I made you proud.

To Sven: You are the best dog in the world, the goodest boi and a bundle of joy. I hope I was able to make you as happy as you made me. Even though I have been far away from you, my memories of you constantly bring me joy and happiness even when I am feeling lost. I think of you every time I throw a ball to the brown dog I have on vscode-pets.

To Chom: I am constantly reminded of how much you do for me without my knowledge every time you are away. Thank you for holding my life together, caring about my health, and loving me when I needed it most. You are one of the most selfless people I have ever met in my life, so much so that you don't realise half the things you do for others. Thank you for being your goofy, cheerful self and for making me laugh in the hardest of times. I strongly believe that you living with me is the reason I was able to concentrate properly on Sage and get it to where it is today. Thank you for being my best friend and for agreeing to be my wife.

To Chris: You are, hands down, the best supervisor a student could ask for. Sage could not have been made anywhere else in the world. You encouraged my way of thinking, polished my abilities, built my confidence, and taught me what it means to be a good researcher. Discussing ideas with you is one of my favourite things in the world, and I hope I get to do that for many more years to come. Thank you, papa Chris/Genius.

To my examiners: I want to sincerely thank my examiners, Ian and Andrew, and the exam convener, Graham, for taking the time to conduct my viva. I am especially grateful to my examiners for reading my thesis carefully and for all the challenging questions—they really made me think and helped me see my work from new perspectives. I half-expect to develop PTSD every time I visit an interferometer site.

To my other teachers: There are lots of people who have influenced my research career, and I would like to sincerely thank all of you for shaping my life enough for me to follow my passion. Dr Martin Hendry, you supported me throughout my PhD and believed in my work. Dr Ai-Lei Sun, you are the first one in this field to trust in my abilities and are the reason I was able to transition into astrophysics. Thank you for taking the time to teach me so much, so patiently. Subasree ma'am and Vijayalakshmi ma'am, from Chettinad Vidyashram, both of you are the reason I am passionate about physics and computer science. Mrs Von Blerk, from Westmount Park School, you were the first teacher to ever appreciate me for my work, and you constantly encouraged me to be the best that I can be. I would also like to thank every single other teacher that I had in my life. All of you shaped my way of thinking into what it is now.

To my friends: Thank you all for all the fun times and joy you have brought into my life. My PhD life would have been a dull mess without seeing all of your happy faces every day: Michael, Joe, Simon, Jack, Maya, Thomas, Rhona, Christian, Federico, Ross, Jess, Leigh, Thejas, Storm, Victoria, and Harry. I would also like to thank every friend outside of work for all the happiness you have given me: Stav, Andrew, Miri, Joachim and Panna. Thank you for being my friends, and I hope we stay friends forever.

To my other companions: M.S., K.M., and S.M., thank you very much for constantly supporting me throughout my PhD. I hope we do get to do lots of work together in the future.

To Minecraft: I wish I started playing Minecraft sooner. Minecraft has this way of calming me down no matter the situation. My world seemed so much bigger, so much more colourful and full of infinite possibility, even though I was living in a tiny room with a single bed and a table with barely any place to walk. This will forever be my favourite game.

To myself: Last but not least, I want to thank myself. I want to thank myself for believing in this work even on the days it felt impossible, for putting in the long hours and the late nights, for pushing through challenges when giving up would have been easier, and for staying consistent even when progress seemed slow. I want to thank myself for having the discipline to keep going, for trusting my instincts, and for being curious, creative, and stubborn enough to see this project through. Most of all, I want to thank myself for being me every single step of the way.

Part I

Motivational and Mathematical Prelude

1

Statistical Detection Theory

Decision making and information extraction are foundational tasks in the design of signal processing systems. The former—determining whether an event of interest has occurred—is the primary focus of this thesis. We frame our discussion of detection theory in the context of its practical applications in gravitational wave science, which is, at its core, an instance of statistical hypothesis testing based on discrete observations of a continuous-time process. This framing enables us to leverage the full suite of tools developed within both classical and Bayesian detection theory (we will mainly follow [6] for introducing these topics).

Contents: We begin in Section 1.1 with a brief overview of detection theory and statistical hypothesis testing, establishing the foundation of the detection problem. Section 1.2 introduces the Neyman–Pearson theorem and its derivation, which characterises the uniformly most powerful test under specified conditions. Section 1.3 then provides a primer on gravitational wave detection, introducing the basic properties of gravitational waves, the variation of waveform morphology, and the behaviour of the noise. Building on this, we present in Section 1.3.4 a signal processing primer, covering the Nyquist limit, Fourier transforms, and power spectral densities, followed by a discussion of non-stationary noise and glitches. The chapter concludes with a summary of the key takeaways and a discussion of open questions that will guide the developments in subsequent chapters.

1.1 The Detection Problem

Mathematically, the discrete observations of some continuous-time process can be represented as some N-point dataset $x(0),\ x(1),\ \ldots,\ x(N-1)$. The primary task of decision making is to transform this data as $T(x(0),\ x(1),\ \ldots,\ x(N-1))$ using some transformation function T, such that the resulting output can be used to infer the presence or absence of a signal. Determining the form of the function T that yields an optimal decision-making process is the central objective of detection theory. This entails defining an appropriate performance criterion and deriving the decision rule that optimises this criterion under specified statistical assumptions.

Simplest Hypothesis Testing Problem: Let us consider the binary detection problem of determining the presence or absence of a signal in background noise. The null and alternative hypothesis for this

case can be defined as,

$$\begin{aligned} \mathcal{H}_0 &: x(n) = w(n) \\ \mathcal{H}_1 &: x(n) = s(n) + w(n) \end{aligned} \tag{1.1}$$

where w is pure noise and s is the signal of interest, with each discrete observation point indexed by n. $\mathcal{H}_0$ is the Null hypothesis that the signal is absent and $\mathcal{H}_1$ is the alternate hypothesis that the signal is present in the data x. Any individual realisation of the data (for example, a time series segment) is just one possible outcome from an underlying probability distribution. Uncertainty is assumed to be built into the problem, and therefore any principled approach to detection must be statistical—using probability theory to describe, model, and reason about the data. We present a simple example below as an application of this problem setup and represent some of the complexities that might arise from this setup:

Example 1: Binary Detection Problem

Consider a simple example: you are an F1 driver at Monza, waiting for the race to begin. The start is indicated by a constant-amplitude signal A appearing amid the background engine noise. We model the observations as

$$\mathcal{H}_0 : x(n) = w(n), \qquad \mathcal{H}_1 : x(n) = A + w(n), \tag{1.2}$$

where $w(n) \sim \mathcal{N}(0,\, \sigma^2)$ is white Gaussian noise with zero mean and variance σ^2. For concreteness, we take $\sigma^2 = 0.5$, and assume the signal amplitude A is known.
Suppose we observe a single sample $x(0)$ and define the test statistic $T = x(0)$. We use a simple decision rule by placing a threshold between the known signal and noise amplitude at $A/2$,

$$\mathcal{H}_0 : T \leq \frac{A}{2}, \qquad \mathcal{H}_1 : T > \frac{A}{2}. \tag{1.3}$$

Here, the inequality that we see is the decision rule $\phi(x)$ based on the test statistic T. This is a simple binary decision rule, where $\phi(x)$ can either be 0 or 1 depending on which condition is satisfied in equation 1.3. With only one observation, noise can cause incorrect decisions. Figure 1.1 (*left*) shows the case $A = 1$, with threshold $A/2 = 0.5$, where misclassifications are common. Increasing the signal amplitude to $A = 2$ (Fig. 1.1, *right*) reduces the error rate. Similarly, reducing the noise variance σ^2 while keeping the mean fixed also decreases misclassifications, as it reduces the overlap between the two hypotheses.

Example 1 informally introduced the concept of correct and incorrect decisions. The following definition formalises these concepts and establishes the terminology used throughout this thesis.

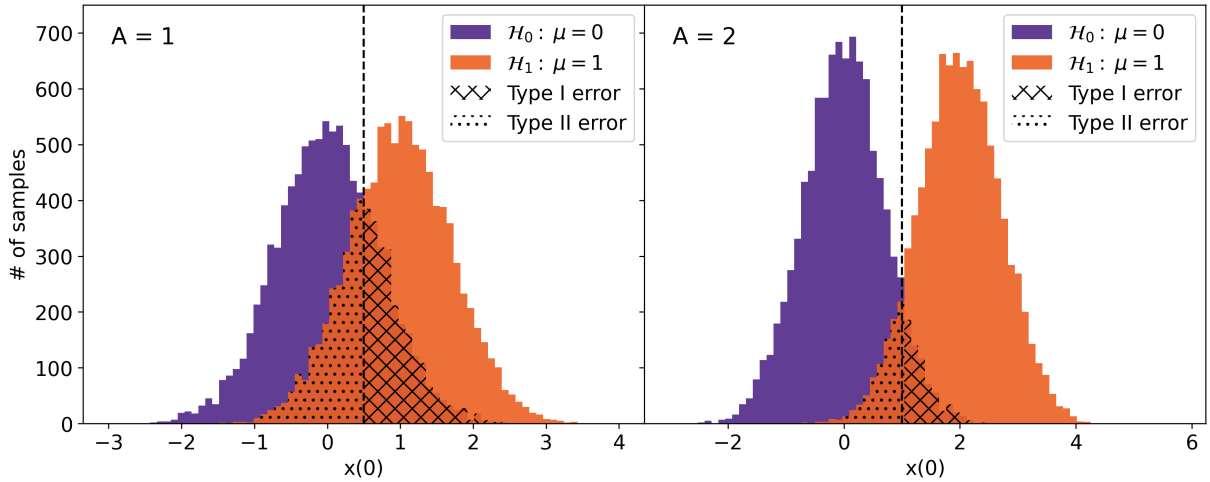


Figure 1.1: Histograms of samples from different Gaussian distributions representing the hypotheses presented in example 1. Each Gaussian distribution is generated using $\sigma^2 = 0.5$ and 10,000 samples. The *left* and *right* panels show the decision making process when $\mu = 1$ and $\mu = 2$ respectively for the alternate hypothesis.

Definition 1: Decision Outcomes

A decision is a **true positive** or a **true negative** when the chosen hypothesis matches the true state. When the null hypothesis $\mathcal{H}_0$ is true but the decision incorrectly declares a signal, the outcome is a **false positive** or **type-I error**. Conversely, when the alternative hypothesis $\mathcal{H}_1$ is true but the decision incorrectly fails to declare a signal, the outcome is a **false negative** or **type-II error**. In Fig. 1.1, the checked-marked and dotted regions illustrate the proportions of false positive and false negative decisions, respectively.

We quantify these outcomes using the following probabilities:

$$P_{\mathrm{D}} = \Pr(\text{decide } \mathcal{H}_1 \mid \mathcal{H}_1 \text{ true}) \quad \text{(probability of detection / true positive probability)} \tag{1.4}$$

$$P_{\mathrm{FA}} = \Pr(\text{decide } \mathcal{H}_1 \mid \mathcal{H}_0 \text{ true}) \quad \text{(probability of false alarm / false positive probability)} \tag{1.5}$$

$$P_{\mathrm{M}} = \Pr(\text{decide } \mathcal{H}_0 \mid \mathcal{H}_1 \text{ true}) \quad \text{(probability of miss / false negative probability)} \tag{1.6}$$

$$P_{\mathrm{R}} = \Pr(\text{decide } \mathcal{H}_0 \mid \mathcal{H}_0 \text{ true}) \quad \text{(probability of rejection / true negative probability)} \tag{1.7}$$

The corresponding *rates* are defined as these probabilities expressed as fractions over repeated trials. For example, the false-alarm rate (FAR) is defined as the count of false alarm per unit time, with false alarm uniformly distributed in time and following Poisson statistics. Note that

$$P_{\mathrm{D}} + P_{\mathrm{M}} = 1, \qquad P_{\mathrm{FA}} + P_{\mathrm{R}} = 1. \tag{1.8}$$

Now that we have defined the different types of detection outcomes, it is natural to seek a quantitative measure that increases monotonically with detection performance, under the constraint of a

fixed false positive rate. Such a measure should capture the degree of separation between the two hypotheses, and it is in this context that the SNR provides a natural and widely used characterisation. Say we define the SNR as, A^2/σ^2 (this is *not* a standard definition in GW science, but we will use this temporarily for simplicity). The detection performance improves as this definition of SNR increases or, in other words, as the distance between the probability density functions (PDFs) of the hypotheses increases. Thus the detection performance is dependent on the degree of *discrimination* between the two hypotheses. More formally, we can define the PDFs of the two hypotheses in the Gaussian case as:

$$\begin{aligned} p(x(0)|\mathcal{H}_0) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}x^2(0)\right) \\ p(x(0)|\mathcal{H}_1) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x(0)-A)^2\right) \end{aligned} \tag{1.9}$$

where $A > 0$. Discriminating between the two hypothesis in this detection formalism can thus be viewed as a *parameter test* of the PDF for $A = 0$ (noise-only) and some arbitrary value $A \geq 1$ (noisy signal).

Decision making in low SNR regime: In example 1 we use a simple decision rule that relies on a single observation of $x(0)$. In a high SNR regime where the type-I and type-II errors are negligible and the amplitude of the signal is known beforehand, relying on one observation is fine in practice and there is little need for decision theory. We can truly appreciate its significance only in the low SNR regime where the false positive and false negative probabilities are much larger, introducing significant confusion.

Say the signal amplitude has a small non-zero amplitude $A = 0.5$ and $\sigma^2 = 1$. This significantly increases the overlap between $\mathcal{H}_0$ and $\mathcal{H}_1$, simulating a low SNR environment. In this situation, it makes intuitive sense that a better and more informed decision can be made using a series of N discrete independent observations of x like $x(0)$, $x(1)$, ..., $x(N-1)$ instead of just $x(0)$. Let us formalise and justify this intuition by updating the decision rule $\phi(x)$ for the alternate hypothesis from $T = x(0) > \gamma$, where γ is some threshold on the amplitude, to a more general form,

$$T = \frac{1}{N}\sum_{n=0}^{N-1} x(n) > \gamma \tag{1.10}$$

where γ is the decision threshold.

Intuition 1: Incorporating more information leads to better decisions

Under the Null and alternate hypothesis,

$$\begin{aligned} \mathcal{H}_0 &: x(n) = w(n) \approx \mathcal{N}(0,\, \sigma^2) \\ \mathcal{H}_1 &: x(n) = A + w(n) \approx \mathcal{N}(A,\, \sigma^2) \end{aligned} \tag{1.11}$$

Let μ denote the mean of a single observation (so $\mu = 0$ under $\mathcal{H}_0$ and $\mu = A$ under $\mathcal{H}_1$). The sum of N independent measurements is

$$\sum_{n=0}^{N-1} x(n) \approx \mathcal{N}(N\mu,\, N\sigma^2) \tag{1.12}$$

and the average of these measurements is,

$$\frac{1}{N}\sum_{n=0}^{N-1} x(n) \approx \mathcal{N}\left(\mu,\, \frac{\sigma^2}{N}\right) \tag{1.13}$$

which, when applied to our hypotheses, becomes,

$$\begin{aligned} \mathcal{H}_0 &: T \approx \mathcal{N}\left(0,\, \frac{\sigma^2}{N}\right), \\ \mathcal{H}_1 &: T \approx \mathcal{N}\left(A,\, \frac{\sigma^2}{N}\right). \end{aligned} \tag{1.14}$$

Thus, as N increases the means remain unchanged while the variance decreases as $1/N$. Using the SNR definition $\text{SNR} = A^2/\sigma^2$, the SNR of the averaged statistic T is

$$\text{SNR}' = \frac{A^2}{\sigma^2/N} = N\frac{A^2}{\sigma^2}, \tag{1.15}$$

so the SNR improves linearly with N.

From Intuition 1, we see that incorporating additional information into the decision process, through a suitable choice of transformation function T, can enhance the effective SNR. This increases the separability of the hypotheses and thereby improves detection performance. However, once the statistical model and the observation space are fixed, we can no longer freely modify the data representation. This leads to an important question: given a detection problem with specified statistical assumptions, is there a decision rule that is provably the best? In other words, can we formally define what it means for a detector to be optimal within a fixed problem setup? We discuss this question in the next section.

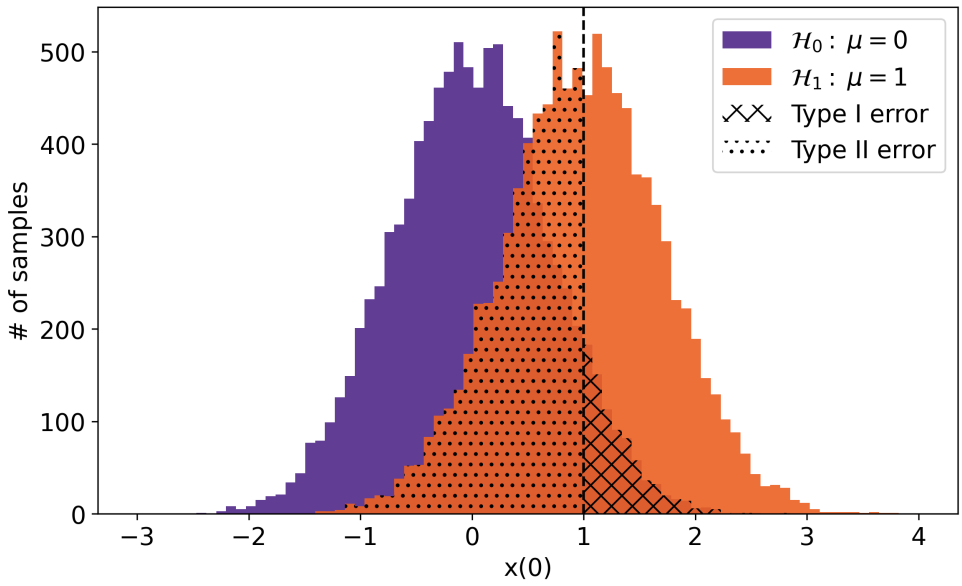


Figure 1.2: Histograms of $x(0)$ under the Null and alternate hypothesis. The vertical dashed line depicts a sub-optimal decision boundary γ. The two hatched regions that represent the type-I and type-II errors show a clear trade-off between the two with changing γ.

1.2 Neyman-Pearson Theorem

For any given transformation function, the resulting type-I (false alarm) and type-II (missed detection) errors form a *zero-sum game*: reducing one typically increases the other. As shown in Fig. 1.2, increasing the decision threshold reduces the probability of false alarms but inevitably leads to more missed detections. Importantly, while the false alarm probability depends on the chosen decision rule, fixing its value does not uniquely specify the rule. Multiple decision rules can yield the same false alarm probability but differ in how well they detect the signal. The Neyman-Pearson approach [7] addresses this by selecting, among all rules that satisfy the false alarm constraint, the one that maximises the detection probability — providing a principled definition of optimality in this setting.

Intuition 2: Can a "better" decision rule not reduce both errors?

It might seem that a cleverer decision rule could simultaneously reduce both types of errors, but such gains usually reflect a fundamental change in the detection setup. For instance, a better transformation function or statistic T (as seen in intuition box 1) may extract more information from the data, effectively reformulating the problem. The Neyman-Pearson framework assumes the statistical model is fixed and identifies the best possible rule within that context. Improvements beyond that stem from better use of information, not from better rules alone.

Neyman-Pearson says that if you're going to allow only a certain amount of false alarms — say, 1 in 1000 — then the best detector is the one that gives you the highest possible probability of catching a

real signal. More formally, the optimal detector is the one that maximises the probability of detection,

$$P_{\mathrm{D}} = p(\mathcal{H}_1|\ \mathcal{H}_1), \tag{1.16}$$

for a fixed false alarm probability

$$P_{\mathrm{FA}} = p(\mathcal{H}_1|\ \mathcal{H}_0) = \alpha. \tag{1.17}$$

Given the Gaussian assumption to the data, we can rewrite equation 1.17 as,

$$\begin{aligned} P_{\mathrm{FA}} &= P(T > \gamma;\ \mathcal{H}_0) \\ &= \int_{\gamma}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}t^2\right) dt \\ &= Q(\gamma). \end{aligned} \tag{1.18}$$

Equation 1.18 is simply the area under the curve (the Gaussian) to the right of the decision threshold γ. $Q(\cdot)$ represents the probability that a standard normal random variable takes a value larger than $\cdot$ (the survival function) which we define in terms of the error function as follows,

$$Q(\gamma) = \frac{1}{2} - \frac{1}{2}\,\mathrm{erf}\left(\frac{\gamma}{\sqrt{2}}\right). \tag{1.19}$$

Similarly, we can rewrite the expression for P_D as,

$$\begin{aligned} P_{\mathrm{D}} &= P(T > \gamma;\ \mathcal{H}_1) \\ &= \int_{\gamma}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(t-1)^2\right) dt \\ &= Q(\gamma - 1), \end{aligned} \tag{1.20}$$

which is the same as equation 1.18 but shifted by 1 in $x(0)$. For $\alpha = 10^{-3}$, we get $\gamma \approx 3$ and we can compute $P_{\mathrm{D}} = Q(\gamma - 1) = Q(2)$ to be 0.02275. So far, we have made some intuitive leaps in formulating the decision rule from the assumption that the data is normally distributed with a known mean and variance. But, how can we prove that $P_{\mathrm{D}} = 0.02275$ is indeed the best possible P_{D} for the given fixed $P_{\mathrm{FA}} = \alpha = 10^{-3}$?

1.2.1 Derivation of the Neyman–Pearson Lemma

Let us now not make any assumptions about the probability distribution functions associated with the two hypothesis of a binary hypothesis testing problem:

$$\begin{aligned} \mathcal{H}_0 &: x \sim p_0(x) \\ \mathcal{H}_1 &: x \sim p_1(x) \end{aligned} \tag{1.21}$$

Our goal is to find a *general* decision rule $\phi(x)$ that maximises the probability of detection. Here, the decision rule is not simply binary, but is a probability that is allowed to take any value within the interval: $\phi(x) \in [0, 1]$, so as to generalise the problem. This nuance is useful when considering discrete or finite-sample problems. For a continuous probability distribution, the probability of observing *exactly* the threshold value is 0. However, in the discrete case, some observations can occur with non-zero probability exactly at the decision boundary. If we use a binary decision function that can only output 0 or 1, the false alarm probability would not be exactly what we want. In this case, we can instead use a $\phi(x) \in [0, 1]$, which essentially says that we flip a biased coin if the observation lands at a boundary. We can adjust the bias so that we can achieve exactly the desired false alarm rate even in discrete settings, thus using the full false alarm budget. In practice, this rarely occurs, but must still be included for completeness.

Under $\mathcal{H}_1$, the data is drawn from the probability density $p_1(x)$. So the probability of detection can be written as:

$$P_{\mathrm{D}} = \mathop{\mathbb{E}}_{x \sim p_1} [\phi(x)] = \int \phi(x)\, p_1(x)\, dx \tag{1.22}$$

which is the probability of deciding $\mathcal{H}_1$ when $\mathcal{H}_1$ is true (more precisely, it is the average over all the cases where you might detect the signal, weighted by how likely those cases are when the signal is actually present). We want to subject equation 1.22 to a constraint on the probability of false alarm:

$$P_{\mathrm{FA}} = \int \phi(x)\, p_0(x)\, dx \leq \alpha \tag{1.23}$$

The keen-eyed might notice that we use $\leq$ instead of $=$ in this expression as opposed to previous definitions. This is to generalise the problem and say that we need not necessarily use the whole false alarm budget. Later in this derivation, when we try to *maximise* P_{D}, we will see that setting this to an equality is Neyman-Pearson optimal.

Step 1: Set Up the Lagrangian

Langrange multipliers can be used to solve problems of the form max $f(x)$ such that $g(x) \leq \alpha$. Our goal with Neyman-Pearson is to maximise P_{D} for a given condition on P_{FA} as given in equation 1.23. To solve this problem, we convert it into a functional optimisation problem using the Lagrange multiplier for the constraint λ. The Lagrangian functional for this problem can be written as:

$$\mathcal{L}(\phi) = \int \phi(x)\, p_1(x)\, dx - \lambda \left(\int \phi(x)\, p_0(x)\, dx - \alpha \right) \tag{1.24}$$

where the term on the left is the objective under the constraint that P_{FA} must equal α where the Lagrange multiplier λ penalises any deviations from this constraint. Equation 1.24 is simply a finite

dimensional constrained optimisation written using Lagrange's theorem. This can be re-written as:

$$\mathcal{L}(\phi) = \int \phi(x) \left[p_1(x) - \lambda p_0(x) \right] dx + \lambda \alpha \tag{1.25}$$

which we want to maximise with respect to $\phi(x) \in [0, 1]$.

Step 2: Pointwise Maximisation

The function $\phi(x)$ assigns a value for every input x. Optimising over all such functions is an infinite-dimensional optimisation problem and is not practically feasible. However, the structure of equation 1.25 makes things simpler. It is an integral over x, and $\phi(x)$ *only* shows up in the integrand multiplied by a function of x (this is $p_1(x) - \lambda p_0(x)$ in equation 1.25). This means that $\phi(x)$ at x does not depend on $\phi(x')$ at x' and thus the optimisation problem at x does not affect the one at x'. More formally, this means that the integrand is separable in x and we want to maximise $\mathcal{L}(\phi)$ pointwise in $\phi(x)$. So, we can break the global problem into many small problems:

$$\max_{\phi(x) \in [0,1]} \phi(x) \left[p_1(x) - \lambda p_0(x) \right] \tag{1.26}$$

where for each x, we want to get the value of $\phi(x)$ that maximises the above expression. We can then patch together all those values to to find the optimal decision rule $\phi^*(x)$. Since the integral is linear in $\phi(x)$, the maximum of equation 1.26 is always at the extremums of $\phi(x) \in [0, 1]$. Let's denote the coefficient as $C(x) = p_1(x) - \lambda p_0(x)$. Now,

- If $C(x) > 0$, the integrand increases with $\phi(x)$, so the maximum is at $\phi(x) = 1$
- If $C(x) < 0$, the integrand decreases with $\phi(x)$, so the maximum is at $\phi(x) = 0$
- If $C(x) = 0$, the integrand is invariant to changes in $\phi(x)$, so the maximum is at $\phi(x) \in [0, 1]$

Step 3: Likelihood Ratio Test

We can combine all the conditions together to write the optimal decision rule as,

$$\phi^*(x) = \begin{cases} 1 & \text{if } \frac{p_1(x)}{p_0(x)} > \lambda \\ 0 & \text{if } \frac{p_1(x)}{p_0(x)} < \lambda \\ \in [0, 1] & \text{if } \frac{p_1(x)}{p_0(x)} = \lambda \end{cases} \tag{1.27}$$

This is the Neyman-Pearson Lemma and it says that the *uniformly most powerful test* at a fixed false alarm rate α is the likelihood ratio test,

$$\Lambda(x) = \frac{p_1(x)}{p_0(x)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \lambda \tag{1.28}$$

for some decision threshold λ. In the context of binary hypothesis testing, the Neyman-Pearson lemma guarantees that the likelihood ratio test (LRT) yields the most powerful detector for a fixed false alarm probability. This derivation shows that the choice of threshold γ and the resulting detection performance are directly and optimally linked through the likelihood ratio.

Definition 2: The Neyman-Pearson Lemma

The Neyman-Pearson lemma provides the most powerful test for deciding between two simple hypotheses $\mathcal{H}_0$ and $\mathcal{H}_1$ at a fixed false alarm probability α. It states that the optimal decision rule is the likelihood ratio test:

$$\Lambda(x) = \frac{p(x|\mathcal{H}_1)}{p(x|\mathcal{H}_0)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \lambda \tag{1.29}$$

where the threshold λ is chosen such that the false alarm probability satisfies $P_{FA} = \alpha$. Among all tests with this constraint, this test maximises the probability of detection P_D.

Let us now get back to the Gaussian-Gaussian binary hypothesis testing problem where we make a decision from a single observation $x(0)$ using some decision threshold γ. We can relate this to the Neyman-Pearson theorem to see how the LRT is optimally linked to the decision making process. The Neyman-Pearson test for deciding $\mathcal{H}_1$ can be written using the likelihood ratio as,

$$\frac{p(x|\mathcal{H}_1)}{p(x|\mathcal{H}_0)} = \frac{\frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x(0)-1)^2\right]}{\frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}x(0)^2\right]} > \gamma \tag{1.30}$$

so,

$$\exp\left[-\frac{1}{2}(x(0)^2 - 2x(0) + 1 - x^2(0))\right] > \gamma \tag{1.31}$$

which can be re-written as,

$$\exp\left[x(0) - \frac{1}{2}\right] > \gamma \tag{1.32}$$

We can now use the constraint on the false alarm probability to determine the decision boundary γ using,

$$P_{\text{FA}} = P\left(\exp\left[x(0) - \frac{1}{2}\right] > \gamma;\ \mathcal{H}_0\right) = 10^{-3} \tag{1.33}$$

where the choice of P_{FA} comes from the calculation made in the beginning of section 1.2. Re-writing equation 1.32 by taking a logarithm on both sides,

$$x(0) > \log\gamma + \frac{1}{2} = \gamma' \tag{1.34}$$

where we are redefining the threshold on the likelihood ratio into a corresponding threshold on the

raw statistic $x(0)$. When this is incorporated into equation 1.33, it becomes

$$P_{\text{FA}} = P\left(x(0) > \gamma';\ \mathcal{H}_0\right) = 10^{-3}. \tag{1.35}$$

It is clear to see the equivalence between equations 1.35 and 1.18 proving that our intuitive choice of decision-making in equation 1.18 is indeed optimal in the Neyman-Pearson sense. Under this P_{FA} constraint, any other decision rule will *provably* be worse in P_{D}. To increase or decrease P_{D} with the usual trade-off with P_{FA}, we simply change the decision boundary γ'.

1.3 The Gravitational Wave Detection Problem

So far, we have discussed statistical decision theory using idealistic examples of a known signal in white Gaussian noise. This assumption is almost never true in real-world detection scenarios. In this section, we will discuss the deviations from ideal data in the context of gravitational wave science and how this affects the theory developed in this chapter. However, we will reserve the discussion related to the optimality of the Neyman-Pearson detector—and the implications of its failure in the context of gravitational wave strain data—for the following chapter, where we explore matched filtering and its limitations in realistic noise environments.

NOTE: Given the focus of this thesis, we will primarily consider gravitational waves originating from compact binary coalescences observed by ground based detectors, approaching the problem from the perspective of a data analyst seeking to construct a near-optimal detection framework.

1.3.1 Gravitational Waves

Gravitational waves are ripples in spacetime generated by asymmetric, time-varying mass distributions of massive astrophysical systems, such as coalescing black holes or neutron stars [8, 9]. Observationally, they appear as weak, transient strain signals—minute changes in the relative lengths of interferometer arms—embedded in noisy time series data. The goal of gravitational wave data analysis is to detect these signals reliably and efficiently. This requires combining theoretical knowledge of waveform morphology with classical and Bayesian statistical methods to construct near-optimal detection pipelines [10, 11]. In this thesis, we treat gravitational waves as real-valued time series governed by physical parameters and focus on developing scalable algorithms capable of identifying them in realistic detector noise.

1.3.2 The LIGO-Virgo-KAGRA Detectors

The Laser Interferometer Gravitational-Wave Observatory (LIGO) [12, 13, 14, 15], Virgo [16, 17, 18, 19, 20], and KAGRA [21, 22, 23, 24] detectors record time-series strain data, with sensitivity spanning approximately 10 Hz to a few kHz, covering the characteristic oscillations from astrophysical sources

such as binary black holes and neutron stars [25, 26, 27]. The resulting data [2, 3, 5, 28, 29] has enabled a wide range of astrophysical and fundamental physics investigations.

The detectors have been pivotal in confirming and testing general relativity. The first direct detection of gravitational waves in 2015, from a binary black hole merger, validated Einstein's prediction [30]. They have also been crucial in studying compact object mergers, providing precise measurements of the properties of binary black holes and neutron stars [30, 29, 28]. Additionally, these observations have constrained astrophysical models, refining our understanding of stellar evolution and compact object formation [31, 32]. The data has also been instrumental in testing fundamental physics [33, 34], probing gravitational wave propagation [35], cosmology and Hubble tension [36, 37, 38], exploring potential extensions to general relativity [39] and gravitational lensing of GW [40, 41, 42].

Why multiple detectors? By combining the data from the LIGO, Virgo and KAGRA detectors, more accurate localisation of gravitational wave sources is achieved, facilitating multi-messenger astronomy, such as electromagnetic follow-up observations of neutron star mergers [29, 43, 44, 45]. Using multiple detectors improves the significance of any found events via detector coincidence and parameter consistency. More importantly, using multiple detectors is crucial for unmodelled burst and stochastic searches [46, 47].

Current status of GW detections: The GWTC-1 [2], GWTC-2.1 [3], and GWTC-3 [5] catalogues report 90 gravitational wave candidates with a probability of being astrophysics in origin, `p_astro` > 0.5, offering a detailed view of black hole and neutron star mergers. GWTC-4 [48] subsequently expanded this dataset with an additional 158 mergers, with an updated threshold on astrophysical probability `p_astro` > 0.9.

Why more gravitational waves? The detection of additional gravitational wave signals is critical for advancing our understanding of the universe, particularly from a Bayesian analysis perspective. In GW parameter inference, the quality of posterior estimates improves significantly with the number of observations, as each additional signal helps refine the prior assumptions and provides better constraints on model parameters. With a limited number of events, the posterior distributions of astrophysical and cosmological parameters remain uncertain, which hinders the precision of astrophysical models. More detections would allow for tighter constraints on these parameters, enabling more accurate population studies of black holes, neutron stars, and other compact objects. Furthermore, detecting more events increases the statistical significance of rare phenomena, allowing us to test alternative theories of gravity, probe fundamental physics, and explore the full diversity of compact object mergers.

Next-generation gravitational wave detectors aim to extend sensitivity and frequency coverage beyond current instruments. Ground-based third-generation detectors, such as the Einstein Telescope [49] and Cosmic Explorer [50], are designed to probe lower frequencies and reach greater distances, enabling observations of a larger population of compact binaries and fainter sources. Space-based observatories, including LISA [51] and TianQin [52], target millihertz frequencies inaccessible from the

ground, allowing the detection of massive black hole mergers, extreme-mass-ratio inspirals, and other long-period sources. Together, these future detectors will provide a more complete view of the gravitational wave universe across a broad range of astrophysical phenomena.

These considerations provide strong motivation for the continued study of gravitational waves and underscore the importance of increasing the number of detections. In the remainder of this chapter, we will explore the parameter space of gravitational wave signals, examine the statistical and structural properties of detector data, and discuss how these characteristics influence detection theory.

1.3.3 Waveform Morphology

The morphology of a gravitational wave describes the detailed shape and structure of its strain signal. These waveforms encode information about the astrophysical processes that generated them, with their time-frequency evolution reflecting the physical parameters of the source. For compact binary coalescences, which involve two closely orbiting black holes or neutron stars, the waveform follows a characteristic sequence of phases. During the *inspiral*, the objects orbit each other at decreasing separation due to the emission of gravitational radiation, causing both the orbital frequency and gravitational wave amplitude to increase gradually. As the objects approach collision, the *merger* occurs, producing a sharp peak in amplitude and capturing the highly non-linear, strong-field dynamics of general relativity. Following the merger, the newly formed single remnant compact object enters the *ringdown* phase, emitting gravitational waves at characteristic quasi-normal mode frequencies as it settles into a stable state, revealing the remnant's final mass and spin. Understanding these morphological features is essential for constructing approximate parameterised waveform model used for matched-filter templates [53, 54, 55, 56], training machine learning models, and distinguishing genuine gravitational wave signals from transient noise in detector data.

Gravitational waveforms from compact binary coalescences are typically described by a set of fifteen parameters, broadly categorised into *intrinsic* and *extrinsic* parameters.

Definition 3: Intrinsic Gravitational Wave Parameters

Intrinsic parameters describe the internal physical properties of the binary system and directly influence the dynamics of the gravitational waveform. These include:

- m_1, m_2: the component masses of the binary,
- $\vec{S}_1$, $\vec{S}_2$: the spin vectors of the individual compact objects (each described by three components, totalling six spin parameters).

Together, these eight parameters govern the structure and evolution of the waveform, such as its phase and amplitude evolution through inspiral, merger, and ringdown.

Definition 4: Extrinsic Gravitational Wave Parameters

Extrinsic parameters describe how the source is oriented and located relative to the detector network and affect how the waveform is projected onto the detector. These include:

- d_L: the luminosity distance to the source,
- ι: the inclination angle between the orbital angular momentum and the line of sight,
- ϕ_c: the coalescence phase at the time of merger,
- α, δ: the right ascension and declination of the source (sky location),
- ψ: the polarisation angle of the wave,
- t_c: the time of coalescence.

All together, these fifteen parameters fully characterise the signal $h(t;\vec{\theta})$ emitted by a compact binary coalescence (CBC) source, where $\vec{\theta}$ denotes the vector of all parameters.

Having introduced the physical parameters that govern the morphology of gravitational wave signals, it is essential to consider how these signals are observed in practice. The detectability of a gravitational wave depends not only on its intrinsic properties, such as masses and spins, but also on the response of the detectors themselves. Key concepts in this context are the low-frequency cut-off and the sensitive band of a detector, which determine the frequency range over which signals can be reliably measured and thus influence the signal-to-noise ratio and the observable portion of the waveform.

The *sensitive band* of a detector denotes the frequency range over which it can achieve significant sensitivity to gravitational waves. The most sensitive region corresponding to the frequency response of the detector where the detector noise is lowest [25, 26, 27]. The sensitive band is bounded at low frequencies by a low-frequency cut-off and at high frequencies by shot noise and other instrumental limitations [57]. The *low-frequency cut-off* of a gravitational wave detector refers to the lower limit of the frequency range in which the detector can reliably measure strain signals. Frequencies below this cut-off are dominated by seismic noise, suspension resonances, and other environmental disturbances, making the detector effectively insensitive to gravitational waves in that regime.

Several commonly used derived parameters appear throughout the gravitational wave literature. We focus on a subset of these quantities that are particularly relevant to our analysis. A commonly used derived parameter is the *chirp mass*, defined as

$$\mathcal{M}_c = \frac{(m_1 m_2)^{3/5}}{(m_1 + m_2)^{1/5}}, \tag{1.36}$$

which predominantly sets the frequency evolution of the inspiral phase and the rate at which the signal increases in both amplitude and frequency—the characteristic "chirp" of the waveform.

The *total mass* $M = m_1 + m_2$ influences the overall timescale of the signal: more massive systems merge faster and at lower frequencies, making them observable for shorter durations in detectors. Spin

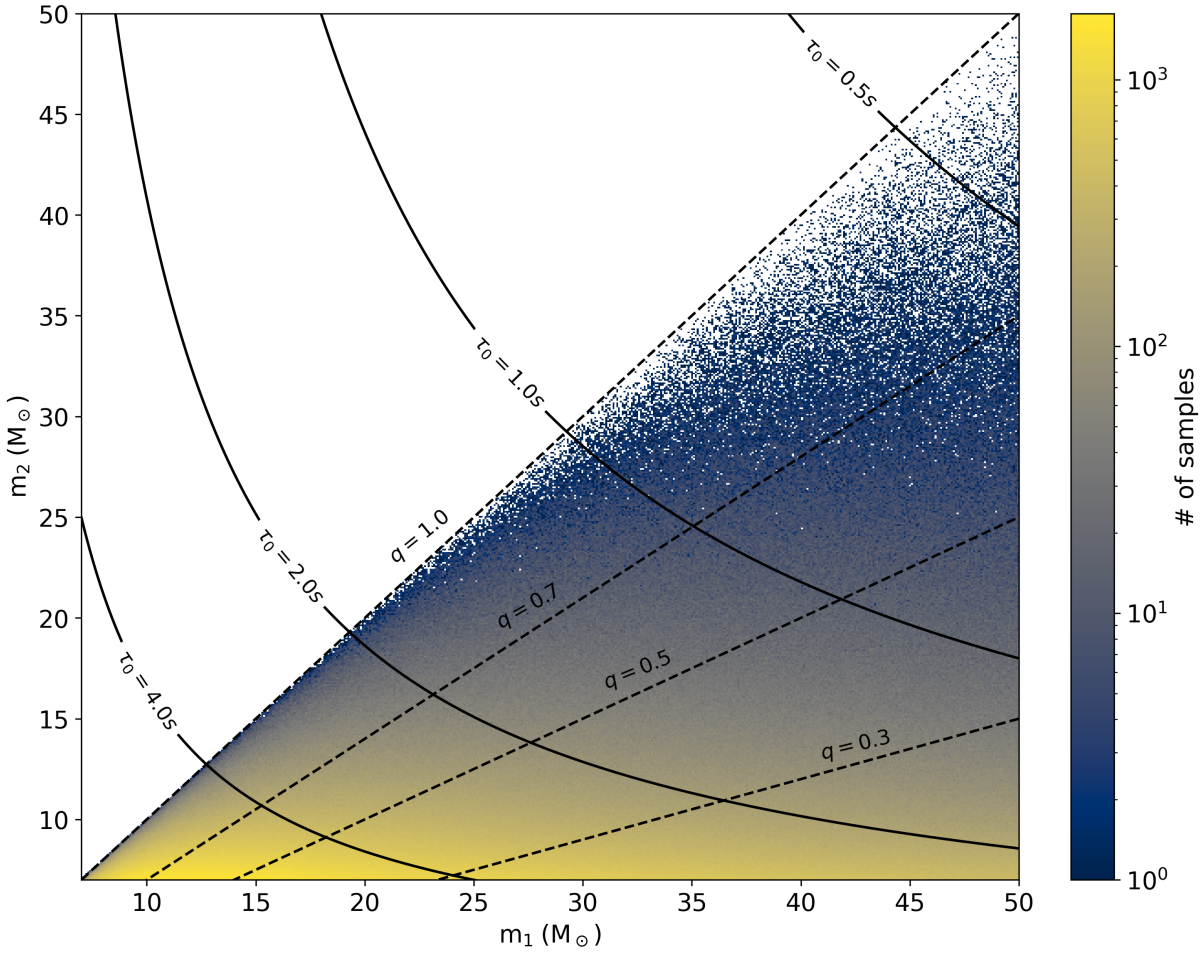


Figure 1.3: 2D histogram of component masses (m_1, m_2) for gravitational wave sources, with contours corresponding to lines of constant chirp mass ($\mathcal{M}_c$) for different values of chirp time (τ_0). Dashed lines represent constant mass ratio (q) contours.

effects, particularly aligned or anti-aligned components with the orbital angular momentum, introduce additional phase evolution, while spin-induced precession adds amplitude and frequency modulations.

The *mass ratio*, defined as $q = m_2/m_1$ with $m_1 \geq m_2$, characterises the asymmetry of the binary system; values closer to unity indicate nearly equal-mass binaries, while smaller values indicate more asymmetric systems. The *symmetric mass ratio*, $\eta = m_1 m_2/(m_1+m_2)^2$, is frequently used in waveform models due to its symmetric behaviour under mass exchange and its bounded range $0 < \eta \leq 0.25$. Another important derived quantity is the *chirp distance*,

$$d_{\text{chirp}} = d_L \left(\frac{\mathcal{M}_{c,0}}{\mathcal{M}_c}\right)^{\frac{5}{6}} \tag{1.37}$$

where $\mathcal{M}_{c,0} = 1.4/2^{1/5}\ M_\odot$ is a fiducial chirp-mass used as a basis for calculation[1]. The reason the

[1]Definition obtained from https://pycbc.org/pycbc/latest/html/pycbc.html#pycbc.

factor $\left(\mathcal{M}_{c,0}/\mathcal{M}_c\right)^{5/6}$ appears is that it corrects the luminosity distance for differences in how signals from binary systems of different chirp masses are observed.

In addition to mass-related parameters, time-domain quantities known as *chirp times* play an important role in characterising the morphology and duration of CBC signals, particularly during the inspiral phase. Two commonly used chirp times are τ_0 and τ_3, which represent the leading-order and next-to-leading-order contributions to the time it takes for the binary system to evolve from a given low-frequency cut-off to merger. These are defined as

$$\tau_0 = \frac{5}{256\pi f_l \eta}(\pi M f_l)^{-5/3} \tag{1.38}$$

$$\tau_3 = \frac{1}{8 f_l \eta}(\pi M f_l)^{-2/3} \tag{1.39}$$

where f_l is the low-frequency cut-off of the detector's sensitive band. These chirp times provide physically intuitive handles on the signal's duration in the detector and are particularly useful for constructing template banks and approximating signal lengths in data analysis pipelines. We also use an equivalent version of τ_0 to study its variation with chirp mass $\mathcal{M}_c$,

$$\tau_0(f_l) = t_c - 5\left(8\pi f_l\right)^{-8/3}\mathcal{M}_c^{-5/3} \tag{1.40}$$

In Fig. 1.3, we use these relations to visualise the relationship between the component masses of binary systems and their associated parameters in gravitational wave analysis.

In general relativity, gravitational waves have two independent polarisation states, commonly referred to as the *plus* (h_+) and *cross* ($h_\times$) modes. These represent quadrupolar distortions of spacetime in orthogonal orientations: the plus polarisation stretches and squeezes space along two perpendicular axes, while the cross polarisation produces the same effect rotated by 45°. The measured strain in a detector is a linear combination of these two modes, weighted by the detector's antenna response functions, which depend on the source sky location and orientation relative to the interferometer arms.

Now, let us visually inspect how varying these parameters changes the waveform morphology. For this purpose, we generate a single baseline gravitational wave signal (only the plus polarisation; redshift is not taken into account) and systematically vary some gravitational wave parameters. We assume a signal low frequency cutoff of 20Hz. The panels in figure 1.4 isolate the effect of a single parameter variation while holding the others fixed (with some minor exceptions) and is compared to the baseline waveform. We use the rest of this section to discuss the differences observed in each panel when compared to the baseline and its implications for the detection problem.

(Panel 0: Baseline Waveform) The baseline waveform corresponds to a binary system with masses of $60M_\odot$ and $20M_\odot$, both with zero spin, inclination $\iota = \pi/3$, coalescence phase $\phi_c = \pi/3$, and $d_L =$

`conversions.chirp_distance` on Sept 2025

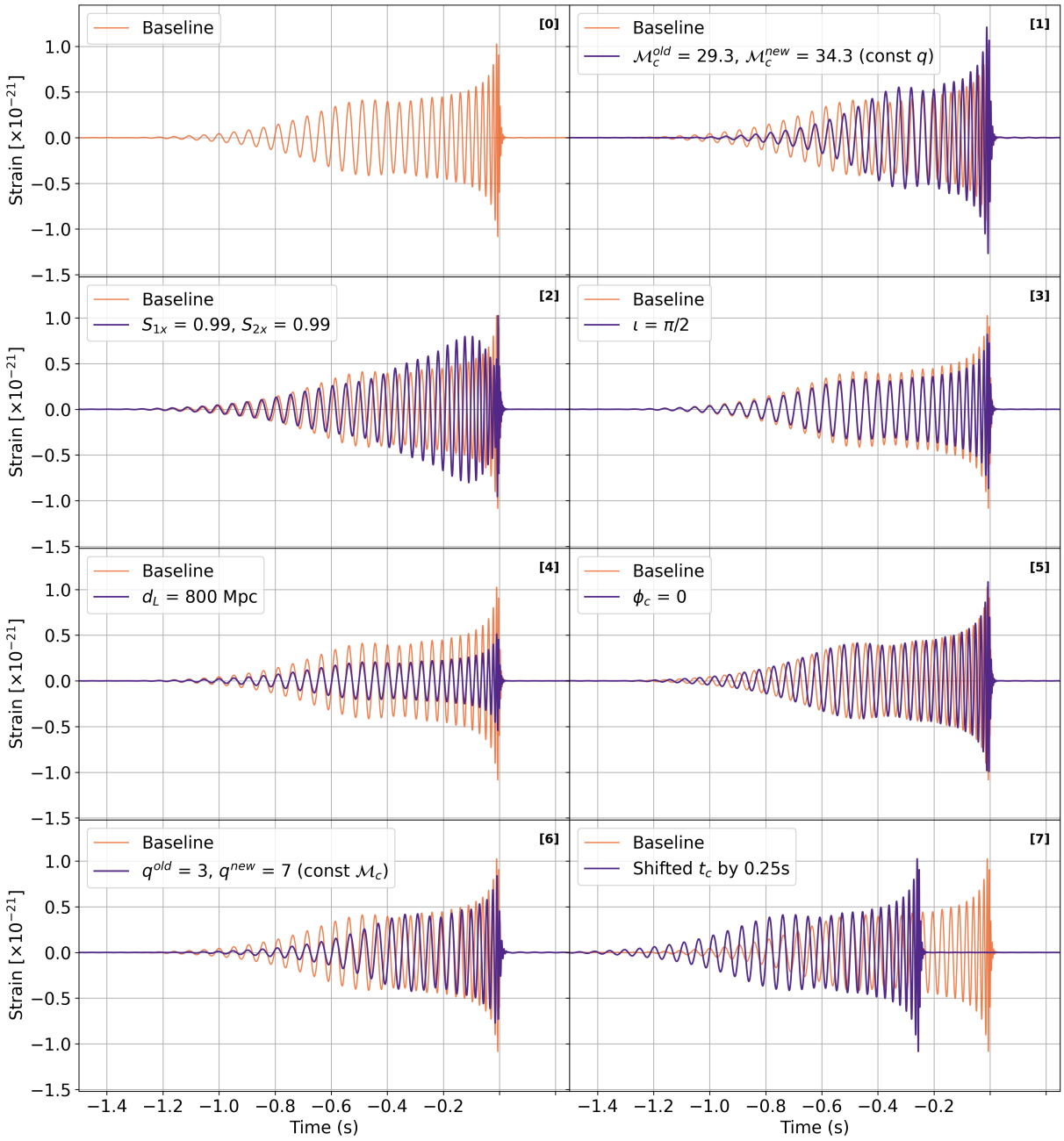


Figure 1.4: Strain amplitude as a function of relative time from merger with $t = 0$ representing the merger time. Comparison of BBH waveforms generated using IMRPhenomPv2 waveform approximant with varying gravitational wave parameters. The baseline waveform is shown in orange, with variations applied to the masses, spin components, inclination, distance, coalescence phase, and time of coalescence. Each subplot corresponds to a single change compared to the baseline. The panel numbers referenced in the text are provided in the top right corner of each subplot in bold.

400Mpc. The signal is produced using the `IMRPhenomPv2` waveform approximant [58, 59, 60].

(Panel 1: Increased $\mathcal{M}_c$ and constant q) In this variation, the component masses of the black holes are increased to 70.33 $M_\odot$ and 23.44 $M_\odot$, while keeping the mass ratio fixed at $q = 3$. As a result, the chirp mass increases slightly, which shifts the overall frequency content of the waveform to lower

values. The amplitude of the waveform also increases slightly, reflecting the change in the total mass of the system. Higher chirp mass lead to smaller chirp time τ_0 and, to the first order, a change in chirp mass while keeping a constant mass ratio can be seen as a time dilation or contraction operation.

Panel 2: Increased spin 1 ($S_{1x} = 0.99$, $S_{2x} = 0.99$) Here, the spins of both black holes are set to $S_{1x} = S_{2x} = 0.99$, significantly increasing the spin components. This introduces strong spin-orbit coupling and precession effects, particularly from the in-plane spin components. These effects cause modulations in both amplitude and phase, especially during the inspiral phase. The orbit becomes misaligned with the total angular momentum vector, causing the waveform to wobble over time.

(Panel 3: Inclination $\iota = \pi/2$) In this variation, the inclination angle is set to $\iota = \pi/2$, meaning that the binary system is viewed edge-on. This change reduces the observed amplitude and slightly modifies the waveform shape, especially if higher-order modes or precession are present. An edge-on view ($\iota = \pi/2$) suppresses the dominant quadrupolar radiation compared to a face-on ($\iota = 0$ or $\iota = \pi$) configuration.

(Panel 4: Distance $d_L = 800$ Mpc) Doubling the distance to the source (from 400 Mpc to 800 Mpc) results in a uniform rescaling of the amplitude of the waveform by a factor of 1/2, as predicted by the inverse-distance scaling of gravitational wave strain. The temporal and spectral properties of the signal does not change directly due to distance, but rather due to an increase in redshift by changing the detector-frame chirp mass. It is straightforward to see that distance and inclination exhibit a degeneracy which impacts the measurement of the luminosity distance; a $\iota = 0$ (face-on) orientation at a farther distance could produce a similar signal amplitude profile as an inclined source at closer distances. This degeneracy can be broken by extracting inclination from precession effects or higher order modes, if these effects are detectable.

(Panel 5: Coalescence Phase $\phi_c = 0$) The coalescence phase is set to $\phi_c = 0$, resulting in a phase shift in the waveform. This adjustment changes the phase of the inspiral, which is reflected as a shift in the starting point of the merger signal. While the amplitude remains the same, this adjustment changes the waveform's phase at coalescence, altering the relative alignment of wave peaks without affecting the timing of the signal.

(Panel 6: Increased mass ratio $q = 7$ and constant $\mathcal{M}_c$) In this variation, the mass ratio is increased to $q = 7$, while the chirp mass is held fixed. The component masses become 96.73 $M_\odot$ and 13.82 $M_\odot$. Although the chirp mass determines the inspiral phase evolution at leading order, increasing the mass ratio leads to a higher total mass, which lowers the merger frequency. As a result, the waveform is shorter given the same low frequency cut-off with the inspiral segment visibly shorter than in the baseline case. The increased asymmetry also alters the waveform shape, particularly during the late inspiral and ringdown, due to a redistribution of angular momentum.

(Panel 7: Time Shift t_c by -0.25 seconds) The coalescence time is shifted by -0.25 seconds, introduc-

ing a small delay in the arrival time of the merger signal without altering its physical content. This demonstrates the expected invariance of the waveform shape under time translation.

So far, our discussion has focused on variations in the *plus* polarisation, h_+, by modifying intrinsic source parameters such as masses and spins. In practice, a detector measures a linear combination of both gravitational wave polarisations, h_+ and $h_\times$, modulated by the detector's orientation relative to the source. This modulation is described by the detector's *antenna pattern* functions, F_+ and $F_\times$, which depend on the source's sky location and polarisation angle. The observed strain in a single detector is therefore

$$h(t) = F_+(\alpha, \delta, \psi)\, h_+(t) + F_\times(\alpha, \delta, \psi)\, h_\times(t), \tag{1.41}$$

where α is the right ascension, δ is the declination of the source, and ψ is the polarisation angle. We now extend our analysis to explore how varying the sky location and polarisation affects the measured strain.

Changing the right ascension α and declination δ alters the relative contributions of h_+ and $h_\times$, leading to differences in the observed amplitude and phase, even for the same intrinsic source parameters. Similarly, varying the polarisation angle ψ rotates the combination of h_+ and $h_\times$ that the detector observes, producing noticeable changes in the waveform morphology. Figure 1.5 illustrate these effects: the first shows how two different sky positions modify the measured strain, and the second demonstrates the impact of changing the polarisation angle for a fixed sky location. These variations highlight the importance of accounting for source orientation and sky location when interpreting observed gravitational wave signals.

1.3.4 Properties of Detector Data

Gravitational wave detector data exhibits a rich structure shaped by both astrophysical signals and instrumental characteristics. Understanding the nature of this data is crucial for effective detection. Mathematically, we can represent detector strain in the time domain as

$$h(t) = s(t) + n(t), \tag{1.42}$$

where s represents an unknown signal and n encompasses the combined noise sources within the detector (if no signal is present $h(t) = n(t)$). In this section, we examine the idealised properties of detector noise, highlighting the key assumptions that facilitate data processing. We then delve into the departures from this ideal behaviour—deviations that must be carefully accounted for in GW data analysis.

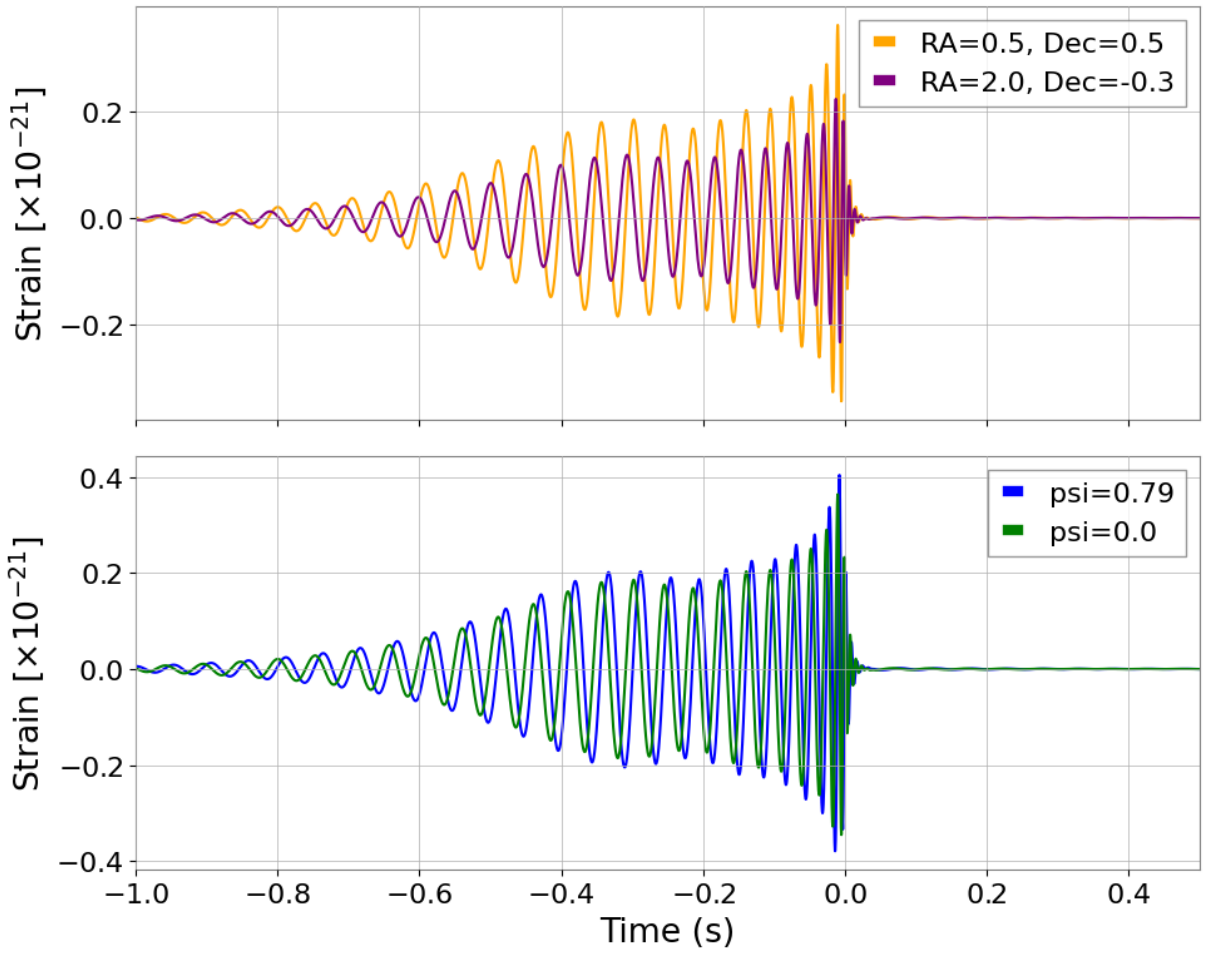


Figure 1.5: Strain amplitude as a function of time: Comparison of gravitational wave strain for two different sky locations (top) and two polarisation angles (bottom) for a fixed binary system.

Signal Processing Primer

Here, we provide a signal processing primer that should aid in understanding the content of this section. We restrict ourselves to concepts relevant to GW data analysis, but provide references to more comprehensive literature wherever appropriate.

The definition of detector strain in equation 1.42 shows continuous time data in the time domain. Although this representation is helpful to understand the mathematical capabilities and limitations that we will encounter in GW data analysis, data acquisition is not continuous, and we often find the need to represent our data in the frequency domain for ease of use. Thus, we provide a brief primer on sampling theory and Fourier analysis within definitions of relevant terminology.

Definition 5: Nyquist Limit

Sampling theory deals with the reduction of some univariate or multivariate continuous-time signal (varies with one or more parameters) into a discrete-time signal by sampling at discrete values of those parameters. A natural question that arises with data reduction is the possibility of perfect reconstruction of the continuous-time signal and the retention of key signal characteristics. The Nyquist-Shannon sampling theorem answers the question: "What is the minimum number of discrete samples required to retain the frequency content of a continuous time periodic signal of a given frequency?".

The **Nyquist frequency**, also known as the **Nyquist limit**, is a fundamental concept in signal processing and data sampling. It is defined as half the sampling rate of a discrete signal and represents the highest frequency component that can be accurately captured without aliasing. Mathematically, if the sampling frequency is f_s, the Nyquist frequency is given by $f_N = f_s/2$. Any frequency components in the input signal that exceed this limit will be undersampled and incorrectly mapped to lower frequencies in the digitised data, a phenomenon known as aliasing [61, 62].

Figure 1.6 illustrates the effects of sampling a sinusoidal signal at different rates relative to the Nyquist frequency. The underlying signal, shown as a black dashed curve, has a frequency of 5 Hz. In the left panel, the signal is sampled at 8 Hz, which is below the Nyquist rate of 10 Hz. As a result, the discrete samples (shown in red) fail to capture the true oscillatory behaviour of the signal, and the apparent waveform is distorted due to aliasing. In contrast, the right panel shows the same signal sampled at 20 Hz, which is above the Nyquist threshold. Here, the discrete samples (shown in green) closely follow the true waveform, allowing accurate reconstruction of the frequency content. This example highlights the importance of satisfying the Nyquist criterion when discretising continuous-time signals for digital processing.

For further information on sampling theory, aliasing, anti-aliasing filters and other relevant signal processing concepts, refer to the books [61, 63, 62].

The concepts introduced in Definition 5, particularly the Nyquist limit, are intimately linked to the frequency content of signals. To precisely characterise and analyse these frequency components, it is essential to move from the time domain (as shown in Equation 1.42) to the frequency domain. This transition is made rigorous through the Fourier transform, which provides a mathematical framework for decomposing a signal into its constituent sinusoidal components. The Fourier transform thereby serves as a foundational tool for understanding both the conditions under which sampling preserves information and the limitations imposed by finite sampling rates.

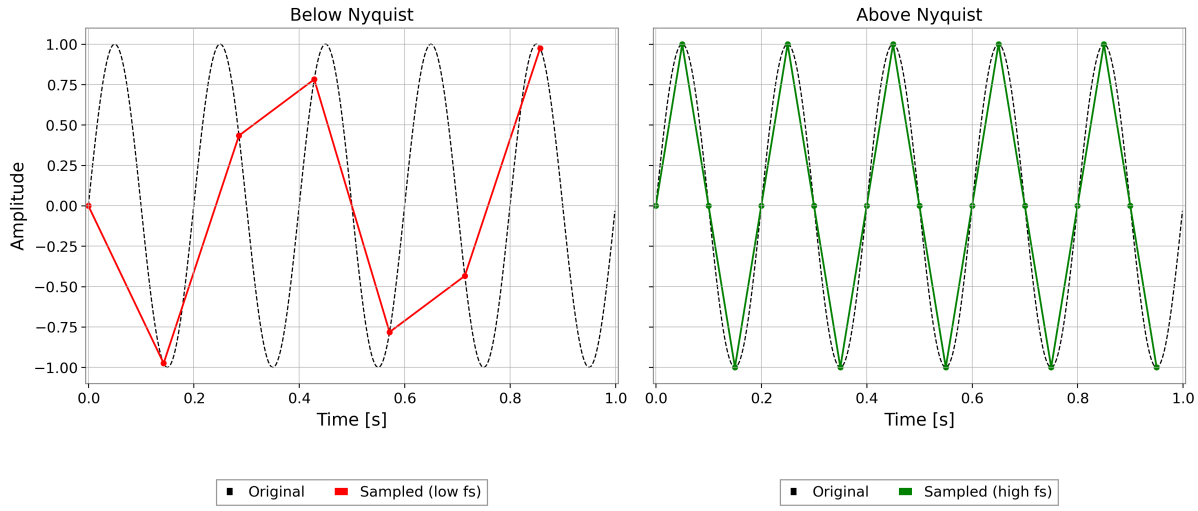


Figure 1.6: Sampling a sinusoidal signal using a sampling rate with a Nyquist limit that is below (left) and above (right) what is required to fully reconstruct the signal frequency. The original waveform is shown as a black dashed line.

Definition 6: Fourier Transform

The **Fourier Transform** $\mathcal{F}$ provides a rigorous framework for analysing signals in terms of their constituent frequency components, rather than their direct temporal evolution. Analogous to the way a complex sound may be decomposed into a superposition of pure musical tones, a continuous time-domain signal $x(t)$ can be represented as a weighted sum of complex exponentials of varying frequencies. This decomposition is realised through the Fourier Transform $\mathcal{F}$, defined as

$$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-i2\pi f t}\, dt, \tag{1.43}$$

where $X(f)$ denotes the spectral representation of $x(t)$, and f represents the frequency in hertz. This transform measures the extent to which the signal correlates with a complex sinusoid at frequency f. When a component of $x(t)$ matches a particular frequency, the integral builds up constructively, while parts that don't match tend to cancel each other out due to their fluctuating nature. As a result, $X(f)$ encodes both the amplitude and phase information of the frequency components present in the original signal.

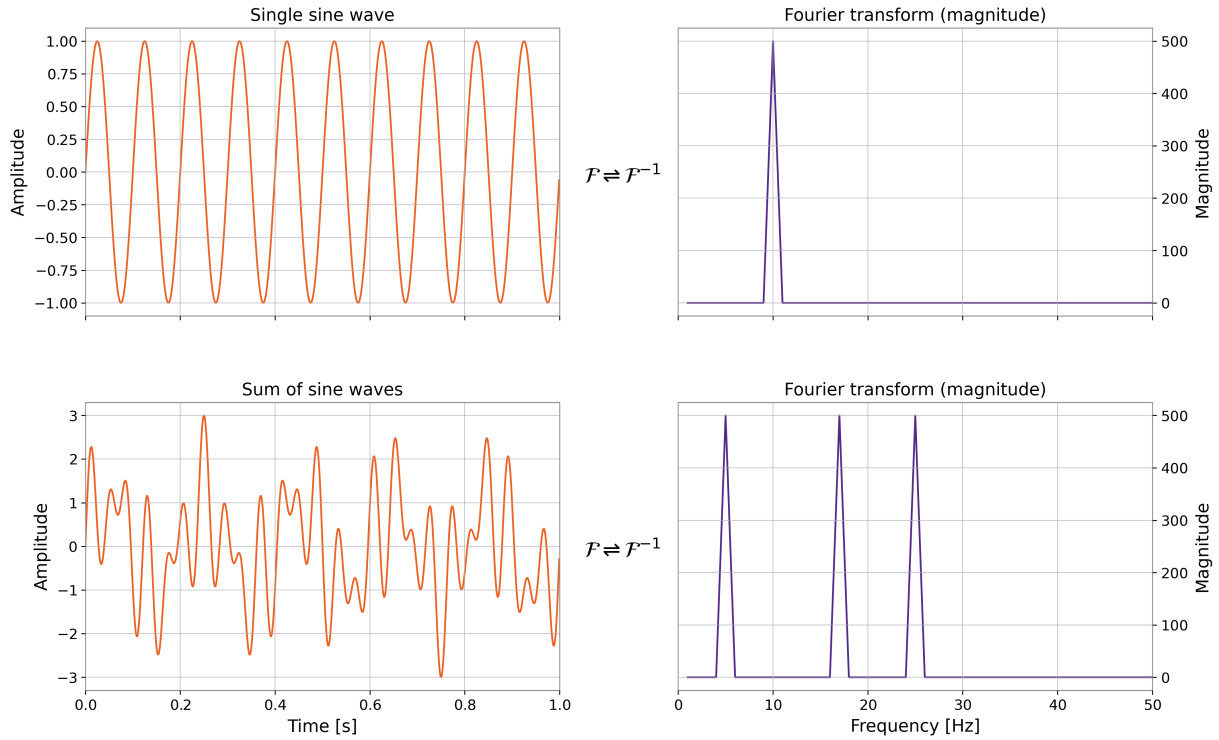


Figure 1.7: Time-domain signals (left) and corresponding Fourier magnitudes (right). Top: single sine wave with a single-frequency peak. Bottom: sum of three sine waves with multiple spectral peaks. Arrows indicate reversibility between the time (orange) and frequency (purple) domains.

Definition 7: Inverse Fourier Transform

The **inverse Fourier Transform** $\mathcal{F}^{-1}$ allows for reconstruction of the time-domain signal from its spectral representation:

$$x(t) = \int_{-\infty}^{\infty} X(f)\, e^{i2\pi f t}\, df. \tag{1.44}$$

This bidirectional mapping between the time and frequency domains affords deep insights into the structure and behaviour of signals, and underpins a wide range of techniques in physics, engineering, and applied mathematics.

Figure 1.7 illustrates the fundamental action of the Fourier transform $\mathcal{F}$ in converting a signal from the time domain to the frequency domain. A monochromatic sine wave, shown in the top-left panel, contains energy at a single frequency, which is reflected in its sharp spectral peak in the top-right panel. In contrast, the bottom-left panel displays a signal composed of multiple sine waves, resulting in a frequency spectrum with multiple peaks (bottom-right panel). While the Fourier transform is mathematically invertible, in practical settings involving discretely sampled signals, the inverse transform $\mathcal{F}^{-1}$ yields an approximate reconstruction due to finite sampling, spectral leakage, and numerical resolution.

Although the Fourier transform offers a complete representation of a signal in the frequency domain, it is often more informative—particularly for noise characterisation and signal detection—to examine how power is distributed across frequencies. This consideration motivates the introduction of the power spectral density, which is formally defined in Definition 8.

Definition 8: Power/Amplitude Spectral Density

The spectral content of detector noise is typically described using the **power spectral density (PSD)**. The power spectrum of continuous time strain $h(t)$ describes the distribution of power for different frequency components that make up $h(t)$. In GW data science, the PSD of background noise is represented as,

$$S_n(f) = \frac{1}{2}\left\langle \tilde{n}(f)\tilde{n}^*(f)\right\rangle, \tag{1.45}$$

where $n(f)$ is the Fourier transform of the noise $n(t)$, $\tilde{n}^*(f)$ is the complex conjugate of $n(f)$, and the angular brackets represents the average of all noise realisations. Since power is proportional to the square of the amplitude of a signal, $S_n^{\frac{1}{2}}(f)$ represents the **amplitude spectral density (ASD)**. In GW data analysis, the PSD quantifies how noise varies with frequency and enables proper normalisation of signal content. By weighting frequency components according to the noise level, analyses emphasise signal-dominated bands while suppressing noise-dominated regions. This makes the PSD essential for both reliable detection and accurate inference.

Ideal Detector Noise

Having established the foundational tools of signal processing—including sampling theory, the Fourier transform, and spectral representations such as the PSD and ASD—we now turn to the statistical characterisation of detector noise. As a starting point, it is useful to consider an idealised model: stationary white Gaussian noise. This simplified framework provides analytical tractability and serves as a reference for understanding more realistic noise environments encountered in gravitational wave detectors.

The three key assumptions about detector noise that aid GW data analysis are as follows:

White Noise: White noise refers to a stochastic process composed of uncorrelated random variables with constant power across all frequencies. It has zero mean, finite variance, and a flat PSD, implying equal noise intensity at each frequency. In contrast, real detector noise is typically *coloured*, exhibiting frequency-dependent structure. Such noise can be approximately whitened by dividing the Fourier-domain data by an estimate of the amplitude spectral density $S_n^{1/2}(f)$, thereby equalising the relative weight of each frequency component.

Gaussian Noise: Gaussian noise refers to a random process whose statistical properties are fully described by a multivariate Gaussian distribution. The probability density function for a single noise

realisation n is given by

$$p(n) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{n-\mu}{\sigma}\right)^2\right), \tag{1.46}$$

where μ is the mean and σ is the standard deviation of the noise. Gaussian noise is completely characterised by its mean and covariance. When the noise is both Gaussian and white, its mean is zero, and the covariance matrix is diagonal with constant variance along the diagonal. It is important to note that Gaussianity and whiteness describe distinct properties of a noise process. Gaussian noise refers to the statistical distribution of the noise amplitudes, whereas white noise is defined by its spectral flatness and lack of temporal correlations. A noise process may be Gaussian but not white, white but not Gaussian, both, or neither.

Stationary Noise: A noise process is said to be *stationary* if its statistical properties are invariant under time translation. This means that quantities such as the mean, variance, and autocorrelation function do not change with time. For stationary noise, the power spectral density is well-defined and stable over the observation period, making it suitable for spectral analysis techniques such as the Fourier transform. Although real gravitational wave detector noise is not strictly stationary, it is often treated as approximately stationary over short time intervals to enable tractable analysis.

While the idealised model of stationary white Gaussian noise provides a useful theoretical baseline, real gravitational wave data exhibits far more complex behaviour. Detector noise is often non-stationary, with statistical properties that evolve over time, and frequently contains transient, non-Gaussian artefacts known as glitches. These features introduce significant challenges for signal extraction and necessitate a range of preprocessing techniques. In the following section, we examine the characteristics of realistic detector noise and discuss their implications for gravitational wave detection.

Realistic Noise Power Spectral Density

Estimating a PSD: The power spectral density of a strain time series characterises the distribution of power across frequency and is a fundamental tool in gravitational-wave data analysis. A simple estimate can be obtained from the squared magnitude of a single Fourier transform of the data, but such an estimate is highly noisy. The Welch method improves upon this by dividing the data into overlapping segments, applying a window function (e.g., Hann or Tukey) to each segment, computing the Fourier transform of each windowed segment, and then averaging the squared magnitudes accounting for proper normalisation. This averaging reduces the variance of the estimate, providing a more precise representation under the assumption that the PSD is stationary within each segment. Importantly, the method does not remove or account for non-stationarity in the underlying noise; it only mitigates the variance inherent to finite-time Fourier estimates.

We emulate and compare four different types of noise: white, coloured, realistic, and real detector noise. Figure 1.8 shows both the PSDs (*left*) and time-domain realisations (*right*) for each model. White

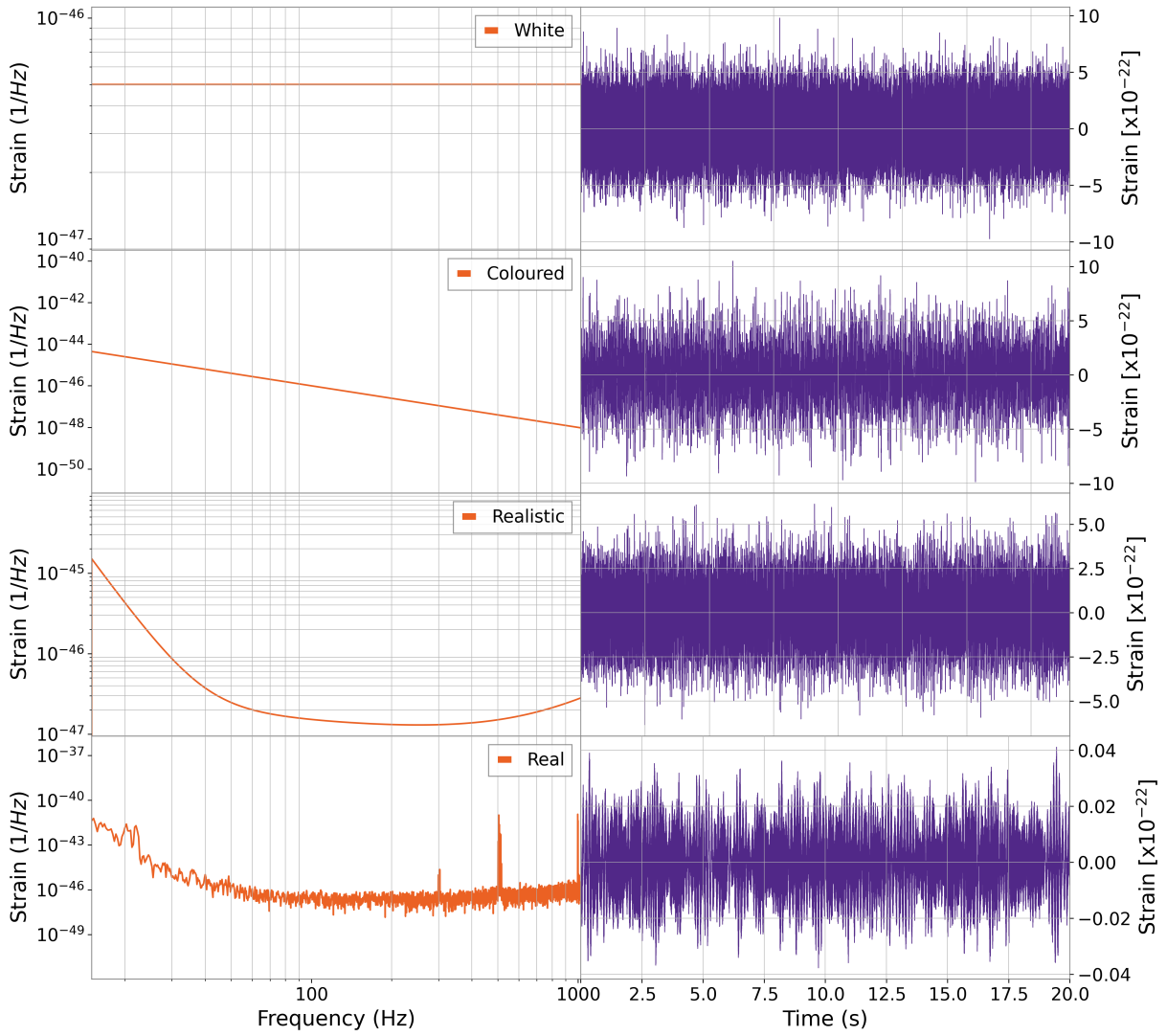


Figure 1.8: Comparison of power spectral densities (left panels) and corresponding time-domain strain realisations (right panels) for different noise models. From top to bottom: white noise, synthetic coloured noise with a power-law PSD, a more realistic noise generated using a smoothed astrophysical PSD template (`aLIGOZeroDetHighPower`), and actual gravitational-wave detector noise from the LIGO Hanford detector (GPS time 1238303637.2-1238303669.2). A low frequency cut-off of 15Hz was applied to all data and the high frequency cut-off is at 1024Hz due to the chosen sampling frequency.

noise exhibits a flat spectrum, while the coloured noise provides a simple example of red noise with power decreasing at higher frequencies. The realistic noise model incorporates spectral features of real GW strain data. Finally, a PSD estimated from real strain data from the LIGO Hanford detector in O3 is included to illustrate the full complexity of measured gravitational-wave noise; the fluctuations seen in the PSD are a consequence of the estimation process rather than the underlying true PSD.

In the real detector noise PSD, several narrow spectral lines are evident—features not present in the synthetic noise models. These line artefacts originate from a variety of instrumental and environmental sources, such as calibration lines, power mains harmonics, and resonances in the suspension

systems for the test mass. Their presence breaks the smoothness of the broadband sensitivity curve and introduces sharp peaks at specific frequencies, which can pose challenges for detection pipelines. Unlike broadband noise, these artefacts are coherent and persistent, necessitating careful treatment during data conditioning and analysis [64, 65].

Estimating noise PSD of data with a signal: Often, for GW data analysis, the segment-wise spectra are combined using the *median* rather than the mean, which reduces the impact of transient excess power in any single segment. Consequently, short-duration gravitational-wave signals contribute minimally to the median PSD, as their energy is confined to only a few segments. However, if a signal is sufficiently long or persistent relative to the window length, it will appear in multiple segments and can bias the PSD estimate upward. The degree to which this occurs depends on the choice of segment length and overlap: longer windows or higher overlap increase the likelihood that the signal contributes to multiple periodograms, while shorter windows localise the signal energy to fewer segments, reducing its influence on the median. Understanding these effects is important for robust noise estimation and for avoiding inadvertent suppression or inflation of signal-to-noise ratios in gravitational-wave searches.

Non-stationarity of Noise

In practice, detector noise is not strictly stationary. The noise power spectral densities evolve over time due to instrumental or environmental changes. For example, slow drifts in the detector alignment, temperature fluctuations, or anthropogenic disturbances can cause the underlying PSD to vary.

Standard estimation techniques, such as the Welch method, rely on the assumption of stationarity within each analysed segment: the data are divided into stretches short enough that the noise properties can be treated as approximately constant. This segmentation is our only means of mitigating non-stationarity when estimating the PSD. However, the estimates obtained from different time segments may not be consistent, reflecting genuine changes in the noise environment rather than statistical fluctuations alone. Thus, while PSD estimation provides a useful characterisation of the noise, it should always be interpreted as a time-local description, not a fixed property of the detector.

Non-Gaussian Transient Noise Artefacts

In addition to stationary background noise, gravitational-wave detectors are affected by short-duration, non-Gaussian transients, commonly referred to as *glitches*. These artefacts arise from a wide range of instrumental and environmental disturbances, including mechanical resonances, electrical discharges, and seismic activity. They are problematic because their morphology can closely resemble that of genuine GW signals, particularly those from compact binary coalescences. Blip glitches, for example, are notoriously hard to distinguish from short duration gravitational waves at high chirp masses. Careful considerations are typically made in matched filtering pipelines are explicitly address these issues [66].

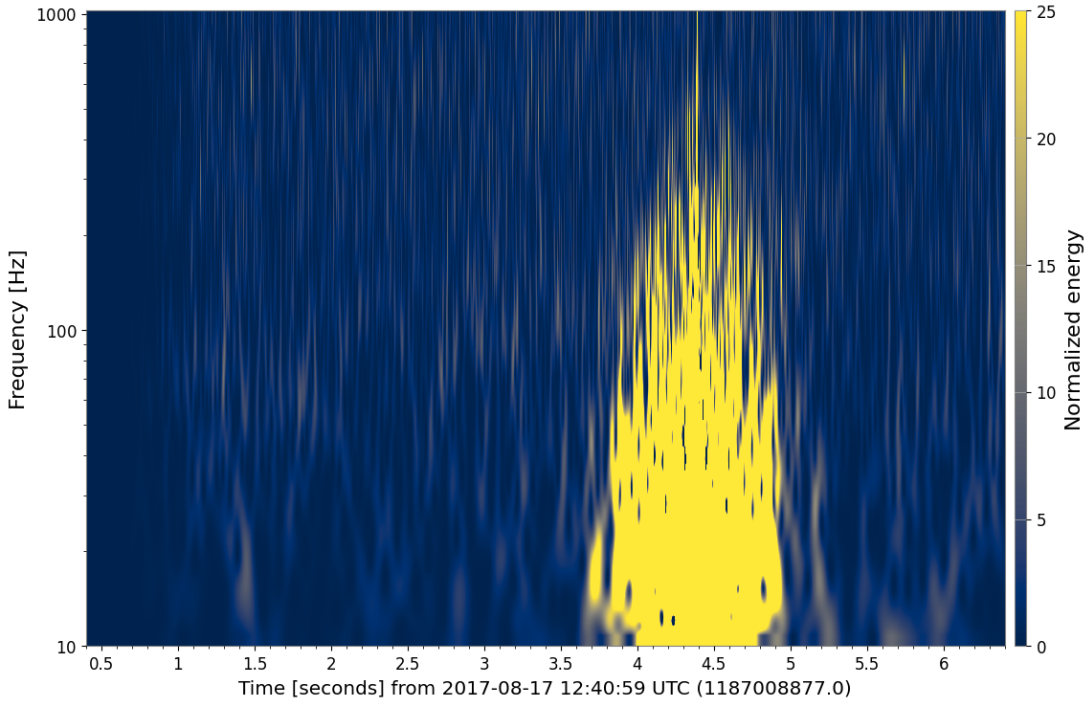


Figure 1.9: Q-transform of strain data around the GPS time of GW170817 showing the presence of a very loud glitch in the L1 detector. Refer [29] for visualisation of GW170817 alongside the glitch.

A well-known example occurred during the observation of GW170817 [29], where a large glitch in the LIGO–Livingston detector overlapped with the true signal. In order to visualise this, we introduce a method called the Q-transform. The Q-transform is a time-frequency analysis technique commonly used in gravitational-wave data analysis for visualisation. It is similar to a short-time Fourier transform, but uses a family of windowed complex exponentials whose duration is adapted to the frequency of interest. Specifically, each basis function has a quality factor Q, defined as the ratio of its central frequency f_0 to its bandwidth Δf,

$$Q = \frac{f_0}{\Delta f}. \tag{1.47}$$

Higher Q corresponds to a narrower frequency band and a longer time duration, which provides better frequency resolution but lower time resolution; lower Q gives better time localisation at the expense of frequency resolution. By varying Q, the Q-transform can produce a time-frequency map that captures signals of different durations and bandwidths efficiently. This makes it particularly useful for identifying short-duration, unmodelled transients in gravitational-wave detectors, such as glitches or bursts, and for visualising their evolution in both time and frequency. See Fig. 1.9 for a Q-transform of strain data around the merger time of GW170817 in the L1 detector. Careful subtraction of the transient was required to recover the astrophysical waveform [67]. This illustrates how glitches can bias detection pipelines, obscure astrophysical signals, or lead to false alarms if not properly identified and mitigated. Glitch characterisation, classification, and removal therefore form an essential component of the data-quality framework in gravitational-wave astronomy.

1.4 Takeaways and Open Questions

This chapter introduced the statistical foundations of signal detection, beginning with the general framework of hypothesis testing and the Neyman–Pearson theorem. The discussion then specialised to the gravitational-wave detection problem, highlighting the challenges posed by detector data, including complex unknown waveform morphology, varying instrumental sensitivity, and the non-stationary, non-Gaussian nature of noise.

Taken together, these elements establish the conceptual bridge between theoretical detection principles and their concrete implementation in gravitational-wave astronomy. The chapter provided the necessary motivation, mathematical tools, and physical context to understand why advanced techniques are required to extract weak astrophysical signals from complex detector noise.

1.4.1 Key Takeaways

- The detection of weak signals in noise can be formally cast as a hypothesis-testing problem, for which the Neyman–Pearson theorem provides an optimal decision rule.
- In gravitational-wave astronomy, detection is complicated by real-world issues such as non-stationary noise, spectral colouring, and transient artefacts.
- Power spectral density estimation underlies the characterisation of detector noise, but its reliability depends on the assumption of stationarity.
- Glitches represent a major challenge because their morphology can mimic astrophysical signals, requiring robust identification and mitigation strategies.

1.4.2 Open Questions

- How can theoretical optimality results, such as the Neyman–Pearson framework, be translated into practical detection algorithms for real detector data?
- Given the limitations of stationarity assumptions, how should one design detection methods that remain robust under non-stationary and non-Gaussian noise conditions?
- What role does prior astrophysical knowledge play in shaping detection strategies, and how can this be balanced against data-driven approaches?
- To what extent can machine learning complement or replace traditional techniques such as matched filtering in addressing these challenges?

2

Matched Filtering for Detection

Matched filtering is a statistically optimal technique for detecting signals of known form in stationary, Gaussian noise [68]. In the context of GW detection, it enables the recovery of weak CBC signals—such as BBH, binary neutron star (BNS) or neutron star black hole (NSBH) mergers—by correlating detector data with modelled waveforms derived from general relativity [48]. It is also used in the search for continuous GW [69, 70], micro- and macro-lensed variants of CBCs [71] and other GW sources where parametric models are possible.

This chapter outlines the theoretical and practical aspects of matched filtering. It begins with a derivation of the matched filter as the Neyman–Pearson optimal detector. Subsequent sections address template bank construction, signal consistency tests, and the limitations of matched filtering in non-Gaussian or non-stationary noise.

To provide a full picture of the simplicity and power of matched filtering while highlighting its assumptions and non-optimality in real noise (which is often not stationary, white Gaussian), we will be processing data around the first known GW event, GW150914 [30], and running a basic form of a matched filter search. For simplicity, we will not be replicating *all* of the steps involved in an actual matched filter search. These will be highlighted in relevant sections and appropriate references will be provided for completeness. We primarily follow [6] for many of the derivations in this chapter.

2.1 Model Comparison

Searches conducted by the LIGO–Virgo–KAGRA (LVK) network of detectors can be formulated as a comparison between two hypotheses. The null hypothesis, $\mathcal{H}_0$, assumes the data consist solely of instrumental noise, whereas the alternative hypothesis, $\mathcal{H}_1$, assumes the presence of both noise and a GW signal. Although in principle all data could contain some level of GW signal, a detection is reported only if the data favour $\mathcal{H}_1$ over $\mathcal{H}_0$ to a sufficient degree.

The likelihood function under each hypothesis is defined by $p(\mathbf{x} \mid \mathcal{H}_0)$ and $p(\mathbf{x} \mid \mathcal{H}_1)$, where $\mathcal{H}_0$ represents the noise-only scenario, while $\mathcal{H}_1$ includes the presence of a signal. Although the likelihood functions are formally defined regardless of the underlying theory of gravity, in practice, their

computation requires explicit models for the expected data under each hypothesis. The noise-only likelihood $p(\mathbf{x} \mid \mathcal{H}_0)$ can be estimated from the statistical properties of the detector noise. However, the signal-plus-noise likelihood $p(\mathbf{x} \mid \mathcal{H}_1)$ depends on a family of signal waveforms parametrised by physical GW parameters. These waveforms are derived under the assumption that general relativity correctly describes the generation and propagation of gravitational waves. Consequently, the likelihood $p(\mathbf{x} \mid \mathcal{H}_1)$ is only an accurate representation of the true signal statistics if general relativity provides a faithful model of the signal morphology. If the true theory of gravity deviates significantly from general relativity, the modelled waveforms would misrepresent the true signals, leading to mismatches in the likelihood and potentially biased or missed detections. Therefore, while the statistical formalism of hypothesis testing remains valid, its effectiveness depends crucially on the accuracy of the underlying physical model assumed for the signal.

The posterior probability for the presence of a signal given the observed data $\mathbf{x}$ is given by Bayes' theorem [72] as

$$p(\mathcal{H}_1 \mid \mathbf{x}) = \frac{p(\mathcal{H}_1)\, p(\mathbf{x} \mid \mathcal{H}_1)}{p(x)} \tag{2.1}$$

$$= \frac{p(\mathcal{H}_1)\, p(\mathbf{x} \mid \mathcal{H}_1)}{p(\mathcal{H}_0)\, p(\mathbf{x} \mid \mathcal{H}_0) + p(\mathcal{H}_1)\, p(\mathbf{x} \mid \mathcal{H}_1)} \tag{2.2}$$

$$= \frac{p(\mathbf{x} \mid \mathcal{H}_1)}{p(\mathbf{x} \mid \mathcal{H}_0)} \left[\frac{p(\mathbf{x} \mid \mathcal{H}_1)}{p(\mathbf{x} \mid \mathcal{H}_0)} + \frac{p(\mathcal{H}_0)}{p(\mathcal{H}_1)} \right]^{-1} \tag{2.3}$$

where $p(\mathcal{H}_1)$ denotes the prior probability for the presence of a signal, the evidence $p(\mathbf{x})$ is the marginal likelihood of the data (the normalising denominator), and $p(\mathcal{H}_0)$ and $p(\mathcal{H}_1)$ are the prior probabilities for the two hypotheses with $0 \leq p(\mathcal{H}_i) \leq 1$ and $p(\mathcal{H}_0) + p(\mathcal{H}_1) = 1$. The prior $p(\mathcal{H}_1)$ expresses the *a priori* belief (before observing $\mathbf{x}$) that a signal is present; $p(\mathcal{H}_0)$ is its complement.

It is convenient to introduce the likelihood ratio (also called the Bayes factor when multiplied by prior odds)

$$\Lambda(\mathbf{x}) \coloneqq \frac{p(\mathbf{x} \mid \mathcal{H}_1)}{p(\mathbf{x} \mid \mathcal{H}_0)}. \tag{2.4}$$

Using $\Lambda(\mathbf{x})$ and the prior odds $p(\mathcal{H}_1)/p(\mathcal{H}_0)$ we can rewrite the posterior (2.1) as

$$p(\mathcal{H}_1 \mid \mathbf{x}) = \frac{\Lambda(\mathbf{x})\; p(\mathcal{H}_1)}{\Lambda(\mathbf{x})\; p(\mathcal{H}_1) + p(\mathcal{H}_0)} = \frac{\Lambda(\mathbf{x})}{\Lambda(\mathbf{x}) + \dfrac{p(\mathcal{H}_0)}{p(\mathcal{H}_1)}}. \tag{2.5}$$

Equivalently, the posterior odds in favour of $\mathcal{H}_1$ are the product of the likelihood ratio and the prior odds:

$$\frac{p(\mathcal{H}_1 \mid \mathbf{x})}{p(\mathcal{H}_0 \mid \mathbf{x})} = \Lambda(\mathbf{x})\; \frac{p(\mathcal{H}_1)}{p(\mathcal{H}_0)}. \tag{2.6}$$

The Neyman–Pearson lemma (see chapter 1) states that for testing a simple null $\mathcal{H}_0$ versus a simple alternative $\mathcal{H}_1$, the most powerful test at a fixed false-alarm probability α is any test that compares the likelihood ratio $\Lambda(\mathbf{x})$ to a threshold λ:

$$\text{reject } \mathcal{H}_0 \quad \Longleftrightarrow \quad \Lambda(\mathbf{x}) > \lambda, \tag{2.7}$$

where λ is chosen so that the probability of false alarm under $\mathcal{H}_0$ equals the prescribed α.

The likelihood ratio $\Lambda(\mathbf{x})$ is the fundamental detection statistic: it determines posterior odds in a Bayesian framework (via multiplication by prior odds) and it is the test statistic of the Neyman–Pearson most-powerful test for simple hypotheses. In the Gaussian-noise case, the matched filter detection statistic is made to be a monotone function of the likelihood ratio, which is why matched filtering is the canonical detection method in GW searches. We will see this more in detail in the following sections.

2.2 Neyman-Pearson Detector and the Matched Filter

Detecting signals obscured by random noise requires a principled statistical framework. Within statistical detection theory, the problem is approached by constructing "receivers" that separate noise from signal in an optimal fashion [61]. This subsection develops the mathematical formulation of such an optimal receiver and examines the qualifications that accompany the term "optimal".

We restate our hypotheses: the null hypothesis, $\mathcal{H}_0$, that only noise is present in the given data, and an alternate hypothesis, $\mathcal{H}_1$, that a gravitational-wave signal is present and contaminated by noise. More specifically, in the context of defining a receiver, the hypothesis test can be written as

$$\mathcal{H}_0 : \mathcal{R}(\mathbf{x}) \leq \lambda, \tag{2.8}$$

$$\mathcal{H}_1 : \mathcal{R}(\mathbf{x}) > \lambda, \tag{2.9}$$

where $\mathcal{R}$ is the receiver acting on a dataset $\mathbf{x} = (x_0, \ldots, x_{N-1})^\top$ with a detection threshold λ. The Neyman–Pearson criterion can be stated as designing a receiver that maximises the probability of detection $P_\mathrm{d} = P(\mathbf{x} \mid \mathcal{H}_1)$ for a given false alarm probability $P_\mathrm{fa} = P(\mathbf{x} \mid \mathcal{H}_0)$. As derived in chapter 1 section 1.2, P_d is maximum for a given P_fa that decides $\mathcal{H}_1$ if

$$L(\mathbf{x}) = \frac{P(\mathbf{x} \mid \mathcal{H}_1)}{P(\mathbf{x} \mid \mathcal{H}_0)} > \lambda, \tag{2.10}$$

where $L(\mathbf{x})$ is the likelihood ratio and for simple hypotheses is the uniformly most powerful test.

Let the observed data be

$$\mathcal{H}_0 : \mathbf{x} = \mathbf{n}, \tag{2.11}$$

$$\mathcal{H}_1 : \mathbf{x} = \mathbf{s} + \mathbf{n}, \tag{2.12}$$

where $\mathbf{s} = (s_0, \dots, s_{N-1})^\top$ is a known deterministic signal and $\mathbf{n}$ is zero-mean Gaussian noise with covariance $C = \mathbb{E}[\mathbf{n}\mathbf{n}^\top]$. The probability densities under each hypothesis are

$$p(\mathbf{x} \mid \mathcal{H}_0) = \frac{1}{\sqrt{(2\pi)^N \det C}} \exp\Big(-\tfrac{1}{2}\mathbf{x}^\top C^{-1}\mathbf{x}\Big), \tag{2.13}$$

$$p(\mathbf{x} \mid \mathcal{H}_1) = \frac{1}{\sqrt{(2\pi)^N \det C}} \exp\Big(-\tfrac{1}{2}(\mathbf{x}-\mathbf{s})^\top C^{-1}(\mathbf{x}-\mathbf{s})\Big). \tag{2.14}$$

The log-likelihood ratio is

$$\ln L(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x}-\mathbf{s})^\top C^{-1}(\mathbf{x}-\mathbf{s}) + \tfrac{1}{2}\mathbf{x}^\top C^{-1}\mathbf{x} \tag{2.15}$$

$$= \mathbf{s}^\top C^{-1}\mathbf{x} - \tfrac{1}{2}\mathbf{s}^\top C^{-1}\mathbf{s}. \tag{2.16}$$

The constant term $\frac{1}{2}\mathbf{s}^\top C^{-1}\mathbf{s}$ can be absorbed into the threshold, so the decision rule is

$$\mathbf{s}^\top C^{-1}\mathbf{x} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma, \tag{2.17}$$

where γ is a threshold fixed by the desired false-alarm rate.

Define the filter vector (the template):

$$\mathbf{h} = C^{-1}\mathbf{s}. \tag{2.18}$$

The test statistic is the linear functional

$$y = \mathbf{h}^\top \mathbf{x} = \mathbf{s}^\top C^{-1}\mathbf{x}. \tag{2.19}$$

This is the matched filter: the inner product of the data with the template weighted by the inverse covariance of the noise.

The matched-filter signal-to-noise ratio is

$$\rho = \sqrt{\mathbf{s}^\top C^{-1}\mathbf{s}}. \tag{2.20}$$

For a general filter $\mathbf{h}$,

$$\mathrm{SNR}(\mathbf{h}) = \frac{\mathbf{h}^\top \mathbf{s}}{\sqrt{\mathbf{h}^\top C \mathbf{h}}}. \tag{2.21}$$

Maximisation by Cauchy–Schwarz shows that $\mathbf{h} \propto C^{-1}\mathbf{s}$, i.e. the matched filter, achieves the maximum $\rho = \sqrt{\mathbf{s}^\top C^{-1}\mathbf{s}}$ [61]. Thus the matched filter is the optimal linear detector for a known signal in Gaussian noise.

Caveats: The matched filter is optimal in the Neyman–Pearson sense only for known deterministic signals in Gaussian noise with known covariance. Realistic noise is non-Gaussian and non-stationary, and the true signals are uncertain in their parameters, so further vetoes, consistency tests, and template banks are needed [61, 68].

2.3 Matched Filtering for GW Detection

In practice, searches aim to compute a detection statistic that ranks candidate events according to their consistency with an astrophysical signal. For compact binary coalescence signals, this is achieved using matched filtering. The matched filter is the optimal linear filter when the exact form of the signal is known, in the sense that the template $h(t)$ is identical to the true waveform $s(t)$ embedded in the data. Here, "known" refers to complete knowledge of the signal morphology, not merely the class of source (e.g., binary black hole or neutron star binary) but the full specification of parameters that determine its waveform.

In realistic searches the parameters are not known, so the data are filtered against a bank of template waveforms, each corresponding to a discrete set of source parameters. The optimal detection statistic would be obtained by marginalising the likelihood ratio $L(\mathbf{x})$ over all unknown parameters by integrating the likelihood ratio over these parameters [73]. However, matched filtering, in practice, *does not* perform this marginalisation; instead, it maximises the likelihood ratio over a discrete set of templates provided by the template bank. The search statistic is then defined as the maximum value of the matched-filter output across the template bank, rather than the analytically optimal but computationally infeasible average over all possible signals.

Let the observed strain data be written as

$$x(t) = n(t) + s(t)\,, \tag{2.22}$$

where $n(t)$ is the detector noise and $s(t)$ is a possible signal. The matched filter was introduced in Sec. 2.2 as the optimal linear filter in Gaussian noise.

In the general derivation of the matched filter, the detection statistic is

$$y = \mathbf{h}^\top \mathbf{x} = \mathbf{s}^\top C^{-1}\mathbf{x}\,, \tag{2.23}$$

where $\mathbf{x}$ is a vector of time-series data, $\mathbf{s}$ is the known signal, and C is the noise covariance matrix. In the discrete derivation, the matched filter uses the noise covariance matrix C, where each element is

$$C_{ij} = \mathbb{E}[n_i n_j] \,, \tag{2.24}$$

with n_i being the noise at the i-th time sample. In continuous time, this generalises naturally to a covariance function $C(t, t')$ that describes the correlation of the noise at two different times t and t':

$$C(t, t') = \mathbb{E}[n(t)n(t')] \,. \tag{2.25}$$

Here, t' is simply another time variable. The covariance function tells us how much the noise at time t is linearly correlated with the noise at time t'. If $t = t'$, this reduces to the variance of the noise at that instant. If $t \neq t'$, it measures how much past or future noise values are linearly related — this is precisely the autocorrelation function.

For stationary noise, statistical properties do not depend on the absolute time, only on the time difference

$$\tau = t - t'. \tag{2.26}$$

In this case, the covariance function depends only on τ:

$$C(t, t') = C(\tau) = \mathbb{E}[n(t)n(t + \tau)] \,. \tag{2.27}$$

This is the standard definition of the autocorrelation function for a stationary process. The significance of this form is that $C(\tau)$ is diagonalised by complex exponentials — that is, the eigenfunctions of the covariance operator are Fourier modes $e^{2\pi i f t}$. This is why the matched filter is conveniently implemented in the frequency domain.

Starting from the autocorrelation function of a stationary noise process $n(t)$,

$$C(\tau) = \mathbb{E}[n(t)n(t + \tau)] \,, \tag{2.28}$$

we can define the Fourier transform of $n(t)$ as

$$\tilde{n}(f) = \int_{-\infty}^{\infty} n(t) e^{-2\pi i f t} dt \,. \tag{2.29}$$

The covariance between Fourier components at frequencies f and f' is

$$\mathbb{E}[\tilde{n}(f)\tilde{n}^*(f')] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbb{E}[n(t)n(t')]\, e^{-2\pi i f t} e^{2\pi i f' t'}\, dt\, dt' \tag{2.30}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} C(t'-t)\, e^{-2\pi i f t} e^{2\pi i f' t'}\, dt\, dt' , \tag{2.31}$$

where we used stationarity: $C(t,t') = C(t'-t) = C(\tau)$ with $\tau = t'-t$. Change variables to $\tau = t'-t$ and t (so $dt' = d\tau$):

$$\mathbb{E}[\tilde{n}(f)\tilde{n}^*(f')] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} C(\tau)\, e^{-2\pi i f t} e^{2\pi i f'(t+\tau)}\, dt\, d\tau \tag{2.32}$$

$$= \int_{-\infty}^{\infty} C(\tau) e^{2\pi i f' \tau}\, d\tau \int_{-\infty}^{\infty} e^{2\pi i (f'-f) t} dt . \tag{2.33}$$

The inner integral over t is a representation of the Dirac delta function:

$$\int_{-\infty}^{\infty} e^{2\pi i (f'-f) t} dt = \delta(f'-f) . \tag{2.34}$$

We then define the one-sided power spectral density $S_n(f)$ as the Fourier transform of the autocorrelation function:

$$S_n(f) = 2\int_{-\infty}^{\infty} C(\tau) e^{2\pi i f \tau} d\tau . \tag{2.35}$$

Substituting this back, we obtain

$$\mathbb{E}[\tilde{n}(f)\tilde{n}^*(f')] = \frac{1}{2} S_n(f)\delta(f-f') . \tag{2.36}$$

Substituting this into the quadratic form $\mathbf{s}^\top C^{-1}\mathbf{x}$ and taking the continuous-time limit yields

$$\mathbf{s}^\top C^{-1}\mathbf{x} \longrightarrow 2\Re\int_0^{\infty} \frac{\tilde{x}^*(f)\tilde{s}(f) + \tilde{x}(f)\tilde{s}^*(f)}{S_n(f)}\, df \equiv (x,s) , \tag{2.37}$$

where (x,s) defines the noise-weighted inner product in the frequency domain. Thus, the discrete vector-matrix form of the matched filter naturally leads to the frequency-domain inner product used in practical gravitational-wave searches, with the inverse covariance operator C^{-1} replaced by division by the power spectral density $S_n(f)$. For a more detailed derivation, please refer to [74].

In the above derivation, $\tilde{s}(f)$ is the Fourier transform of $s(t)$, the template, independent of the coalescence time. We can account for the unknown coalescence time by replacing $\tilde{s}(f)$ with $\tilde{s}(f)e^{2\pi i f t}$ in the inner product shown in Equation 2.37. Equation 2.37 is, more specifically, defined to be the filter for the plus polarisation of the GW, $s_+(t)$, but we do not need a separate filter for the cross polarisation $s_\times(t)$. $s_\times(t)$ is orthogonal to $s_+(t)$ and their Fourier transforms are related as $\tilde{s}_\times(f) = i\tilde{s}_+(f)$, so we can simply get the imaginary counterpart of equation 2.37 for the cross polarisation. This accounts for

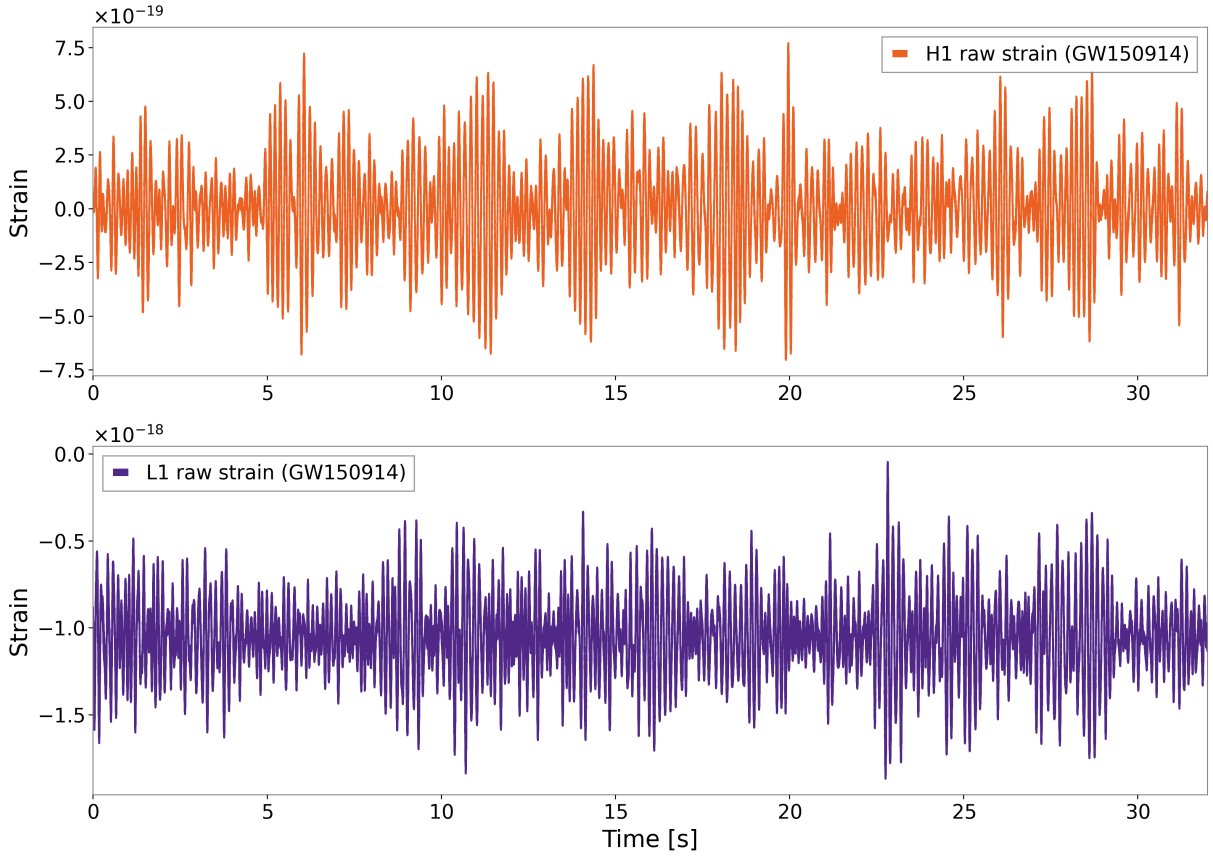


Figure 2.1: Raw strain data from the Hanford (H1) and Livingston (L1) detectors spanning 32 seconds around the time of coalescence of GW150914.

the unknown phase as well [75].

While the matched filter offers an optimal method for known signal recovery under idealised noise assumptions, its practical efficacy depends critically on the quality and conditioning of the input data. Real GW detector data requires careful preprocessing to ensure the validity of the filtering assumptions. The next section discusses some typical preprocessing steps necessary to prepare the strain data for matched filtering.

2.4 Data Preprocessing

Before matched filtering can be applied to the detector strain data, several preprocessing steps are performed to condition the data and reduce the influence of instrumental artefacts. These steps typically include windowing, whitening, and bandpass filtering. Each of these operations plays a crucial role in ensuring that the matched filter operates near-optimally under the assumption of stationary, Gaussian noise. As a practical example of these data preprocessing steps, we show the example of processing strain data around GW150914 shown in Fig. 2.1. We follow the `PyCBC` tutorial for data pre-processing given in this https URL.

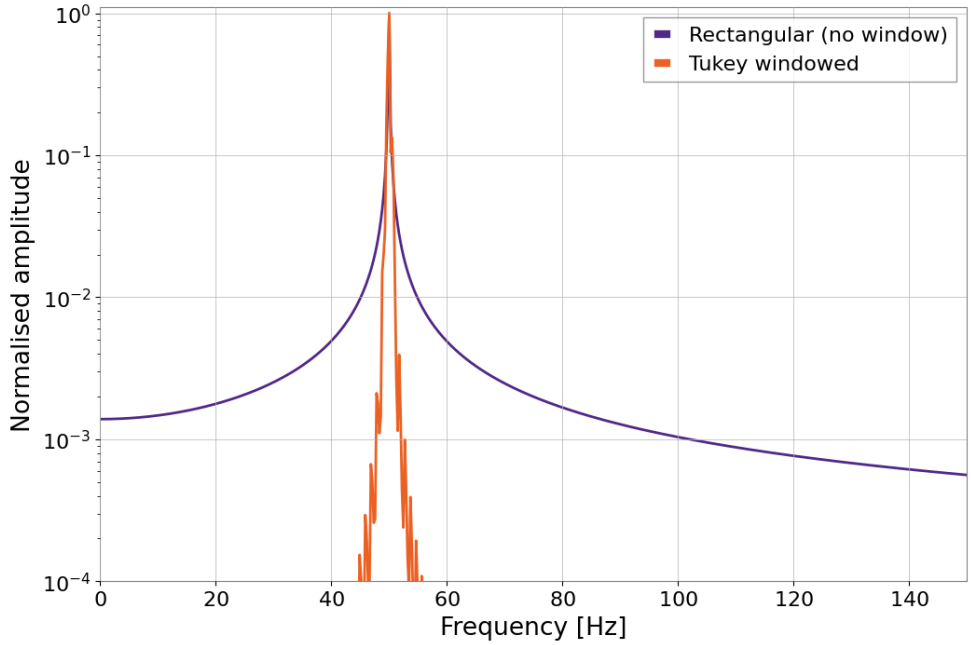


Figure 2.2: Comparison of Fourier spectra for a 50 Hz sine wave using a rectangular window (no tapering) and a Tukey window.

Windowing

Gravitational wave strain data is analysed in finite-length segments. To suppress spectral leakage caused by the implicit periodicity assumption of the discrete Fourier transform, a window function is applied to each segment. This function smoothly tapers the beginning and end of the time series to zero, preventing sharp discontinuities that would otherwise introduce spurious high-frequency components in the frequency domain.

A commonly used window in GW data analysis is the Tukey window, which combines a flat central region with cosine tapers at the ends. The fraction of the segment that is tapered is chosen to balance leakage suppression against loss of effective data duration. The windowing function is typically applied both to the detector strain and to the templates before performing the matched filter operation (using a taper fraction of 0.1). A plot for this operation on the strain data would not hold much meaning; we instead show its effect on spectral leakage.

To illustrate the effect of windowing, consider a simple 50 Hz sine wave sampled at 1 kHz over 4096 points. Its discrete Fourier transform can be computed both with and without a window. Using a rectangular window (no tapering), the fast Fourier transform exhibits spectral leakage, spreading power into neighbouring frequencies as seen in Fig. 2.2. Applying a Tukey window smooths the time-domain edges, which reduces leakage at the cost of slightly broadening the main spectral peak.

For a systematic study of the properties of different windows, we refer the reader to, e.g., [1, 2] for theoretical introductions, and [3] for a specific discussion in the context of gravitational-wave data

analysis.

We refer the Reader to references [76, 77] for a foundational study of the properties of different windowing functions and a rigorous theoretical overview. References [78, 79] continue this discussion in the context of GW data analysis.

Whitening

Matched filtering assumes that the background noise is stationary and Gaussian. However, the raw strain data recorded by GW detectors is coloured by frequency-dependent instrumental noise. Whitening is therefore applied to flatten the PSD of the noise, transforming the data into a form in which all frequency components contribute equal weight to the matched filter.

Let $x(t)$ denote the time-domain strain data, and let $S_n(f)$ be the one-sided PSD (as defined in section 4.1.1) of the noise estimated over a longer stretch of data. In the off-source method of PSD estimation, a long stretch of data excluding the analysis segment (which is assumed to contain the signal) is divided into subsegments, and the PSD is taken as the median of their periodograms. This avoids signal contamination but assumes stationarity over a long interval. The on-source method uses only the analysis segment, containing both noise and any signal. Both assume stationary, Gaussian noise, but the on-source method requires stationarity over a shorter, more local interval. This estimated PSD is then used for whitening.

The whitening process involves transforming $x(t)$ into the frequency domain via a Fourier transform $\mathcal{F}$, dividing by the square root of the PSD, and then transforming back via the inverse Fourier transform $\mathcal{F}^{-1}$:

$$\tilde{x}_{\text{white}}(f) = \frac{\tilde{x}(f)}{\sqrt{S_n(f)}}, \qquad x_{\text{white}}(t) = \mathcal{F}^{-1}\left[\tilde{x}_{\text{white}}(f)\right]. \tag{2.38}$$

The same operation is applied to waveform templates before matched filtering (In matched-filter searches such as PyCBC and MBTA, the detector strain is “over-whitened" by dividing by the PSD, while the template waveforms remain unchanged, an approach chosen for computational efficiency and is mathematically equivalent.). Whitening has the effect of normalising the inner product used in matched filtering, such that it becomes near-equivalent to the optimal detection statistic under the assumption of white Gaussian noise. Whitening removes the frequency-dependent noise weighting and maps the data into a space where the noise is approximately white. The matched-filter inner product from equation 2.37 becomes

$$(x, s) = 2\,\Re \int_0^\infty \left(\tilde{x}_w^*(f)\tilde{s}_w(f) + \tilde{x}_w(f)\tilde{s}_w^*(f)\right)\, df\,, \tag{2.39}$$

removing the explicit $1/S_n(f)$ weighting and reducing to a Euclidean inner product in whitened space. Figure 2.3 shows the PSDs of the Hanford and Livingston detectors estimated using 32 seconds of data around GW150914. This is then used to whiten the strain shown in Fig. 2.1 to obtain Fig. 2.4.

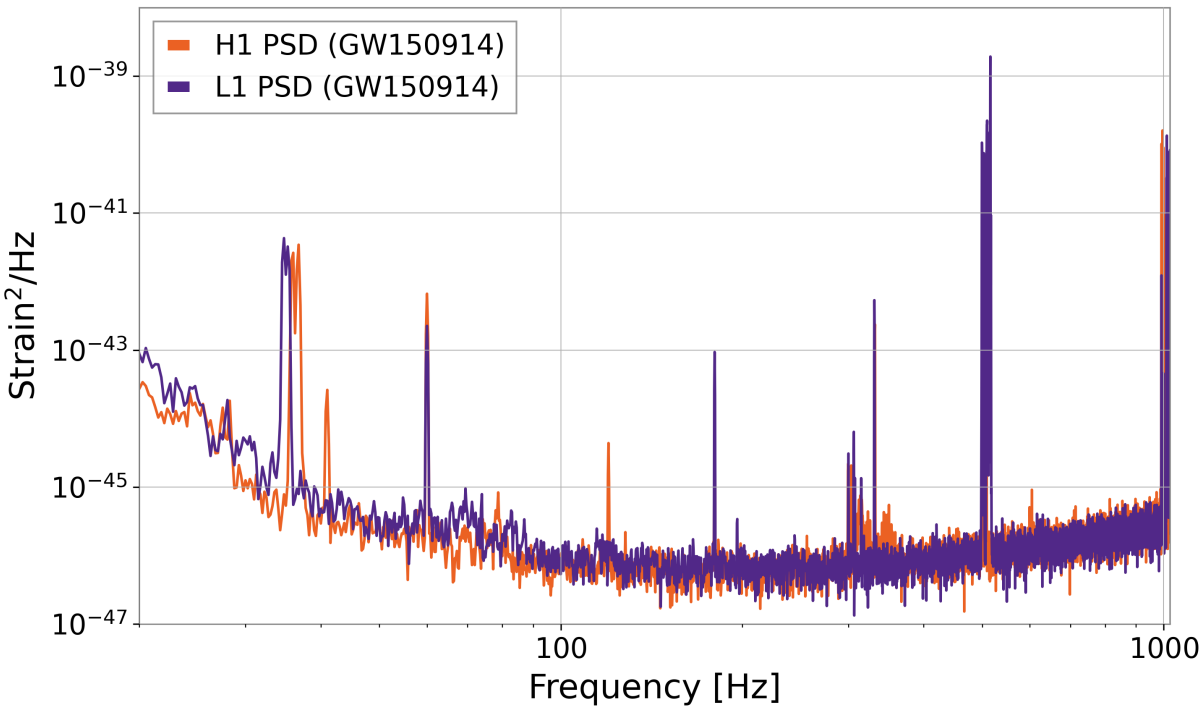


Figure 2.3: Estimated noise PSDs for the Hanford (H1) and Livingston (L1) detectors computed using Welch's method over the 32-second data segment surrounding GW150914

Bandpass Filtering

In addition to whitening, the data is typically subjected to bandpass filtering to restrict the analysis to a frequency range in which the detectors are most sensitive. For the LIGO and Virgo detectors, this range is typically between 15–1024 Hz for binary black hole signals, although the precise bounds may vary depending on the search and the expected source masses.

Narrowband spectral features, such as mechanical resonances (e.g., violin modes) or electrical lines (e.g., 50/60 Hz mains harmonics), can be removed using notch filters. These filters excise or suppress specific frequency bins known to be contaminated by persistent noise lines that are not well-modelled by the PSD and could otherwise lead to spurious high matched filter outputs. This is not typically done in matched filtering searches but can be used for visualisation purposes.

The whitened strain data from Fig. 2.4 are subsequently bandpassed to isolate the frequency range of interest and suppress residual low- and high-frequency noise as shown in Fig. 2.5. We have also time-shifted the data to account for the signal arrival time differences due to the source's sky location and detector location (this is done purely for visualisation purposes and is *not* a pre-requisite for matched filtering).

These preprocessing steps together ensure that the input to the matched filter resembles the idealised assumptions under which the detection statistic is derived, thereby improving both sensitivity and reliability of candidate event ranking.

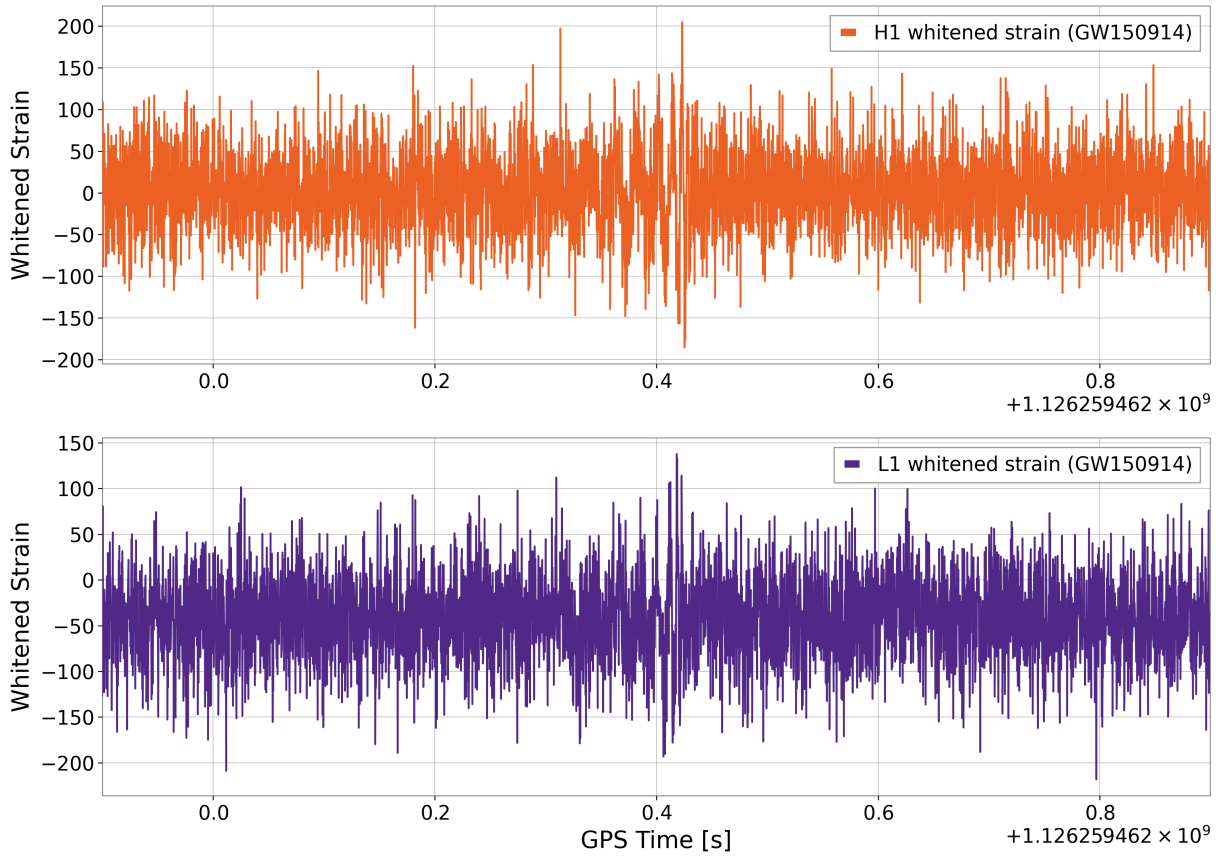


Figure 2.4: Whitened strain data from the Hanford (H1) and Livingston (L1) detectors plotted against GPS time spanning 32 seconds around the time of GW150914. The data has been processed using the estimated PSDs to remove the frequency-dependent noise profile.

2.5 Template Bank Construction

Note: We primarily follow [80] and [81] for this section to briefly describe a simple template placement scheme.

Accurate detection of GW signals from compact binary coalescences requires the comparison of detector data with theoretically modelled waveforms. While general relativity admits exact solutions for isolated, highly symmetric configurations—such as the Schwarzschild or Kerr spacetimes—no closed form solution exists for the full two-body problem. Consequently, approximate methods must be employed to describe the dynamical evolution of binary systems. The post-Newtonian (PN) approximation has enabled the construction of accurate *models* for the late-time dynamics of inspiralling compact binaries. This, in turn, permits the use of matched filtering, under the assumption that the detection problem involves a known signal embedded in stationary, white Gaussian noise. However, the morphology of incoming GW signals depends on several parameters that are not known *a priori* (refer section 1.3.3 for more information). To detect such signals, we generate waveform models, or *templates*, using approximate parametric descriptions that span a representative range of physically

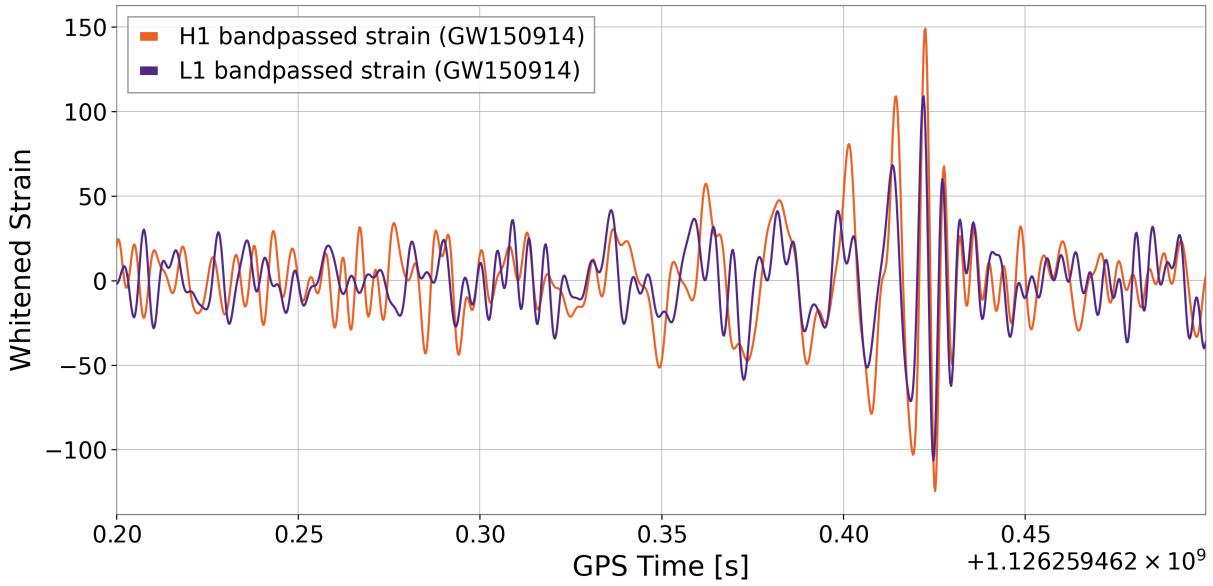


Figure 2.5: Whitened and bandpassed strain data from the Hanford (H1) and Livingston (L1) detectors, time-aligned by correcting for the measured inter-detector time delay.

plausible source parameters. The systematic organisation of these templates into a *template bank* constitutes a foundational step in matched filtering pipelines. Since filtering the data against a continuum of templates is infeasible, the template bank is inherently discrete. Consequently, a mismatch between any real signal and the nearest template in the bank leads to a reduction in the recovered SNR. The spacing of templates must therefore be chosen to balance the loss in SNR with the computational cost of filtering the data against the entire bank. This spacing is determined by imposing a *minimal match* criterion between any potential signal and the templates.

The *minimal match* (MM) sets a lower bound on how closely any signal within the targeted parameter space must align with at least one template in the bank. Commonly chosen values include 0.95 or 0.97, representing a 95% or 97% overlap requirement, respectively (overlap is one particular measure we chose for this explanation. There are other measures like fidelity, faithfulness etc.). This criterion ensures that no signal is missed due to sparse template coverage.

Formally: Let us assume a space of any possible signal $\hat{s}(\theta')$ with some GW parameters θ'. A template bank is constructed with templates $\hat{h}(\theta_i)$ for $i = 0, 1, \ldots, N_{\text{t-1}}$ where N_t is the number of templates. The template bank is considered optimal if N_t is minimal and if for any given signal there exists at least one template in the template bank such that,

$$\min_{\theta'} \max_{i} \left(\hat{s}(\theta'), \hat{h}(\theta_i) \right) \geq \text{MM}. \tag{2.40}$$

In simpler terms: even in the worst case, every signal has at least one template that overlaps with it by at least the minimal match. A higher value of MM leads to finer template spacing and better recovery of signals but increases the computational cost. Refer to [80, 82] for more details. Template banks are

constructed so that the loss in SNR from mismatches between a true signal and its nearest template does not exceed a specified tolerance, the minimal match.

This metric provides a local approximation for the fractional loss in matched-filter SNR arising from a small parameter offset between neighbouring templates. Let $h(\theta)$ denote a normalised template, where θ is the vector of intrinsic parameters with components θ^a. For a small displacement $\Delta\theta^a$, the overlap between templates at θ and $\theta + \Delta\theta$ can be expanded as

$$(h(\theta), h(\theta + \Delta\theta)) \simeq 1 - g_{ab}\,\Delta\theta^a \Delta\theta^b\,, \tag{2.41}$$

where the symmetric tensor

$$g_{ab} = -\frac{1}{2}\frac{\partial^2 (h(\theta), h(\theta'))}{\partial\theta^a\,\partial\theta'^b}\bigg|_{\theta'=\theta} \tag{2.42}$$

defines the *metric on the template manifold*. In practice, template placement algorithms use this quadratic form to approximate mismatches and to ensure that templates are distributed such that the maximal loss in overlap with any signal does not exceed the minimal match threshold.

Simple template placement: We look at a simple non-optimal example of the square placement algorithm (in the 2D case) and we use the stationary phase approximation model waveform model [83, 84]. With a simple non-spinning circular model of CBCs, we assume that the template varies only due to two intrinsic mass parameters. The extrinsic parameters can be analytically maximised or marginalised without including them in the bank. The two mass parameters that the waveform depends on is taken to be the total mass $M = m_1 + m_2$ and symmetric mass ratio $\eta = m_1 m_2 / M^2$. For these mass parameters, the grid spacing varies across the parameter space (as they depend non-linearly on minimal mismatch) and the metric is not constant in (M, η). It is also not constant for any of the mass parameter pairs, and this makes it difficult to construct the simple template placement scheme of an equally spaced grid. A more practical coordinate system is given by the chirp times τ_0 and τ_3, which creates a flat/constant metric and in turn allows us to place the template on a Cartesian grid. The chirp times are re-defined for clarity (first defined in chapter 1 equation 1.38) as follows:

$$\tau_0 = \frac{5}{256\pi f_l \eta}(\pi M f_l)^{-5/3} \tag{2.43}$$

$$\tau_3 = \frac{1}{8 f_l \eta}(\pi M f_l)^{-2/3}, \tag{2.44}$$

where f_l denotes the signal low-frequency cutoff. Although the metric is only strictly constant at 1PN order, we assume that it is globally flat. These chirp time parameters are especially convenient as they can be analytically mapped back to (M, η). Inverting the relations yields direct expressions for the total mass M and the symmetric mass ratio η in terms of the chirp time parameters τ_0 and τ_3:

$$M = \frac{5}{32\pi^2 f_L}\frac{\tau_3}{\tau_0}, \quad \eta = \frac{1}{8 f_L \tau_3}\left(\frac{32\pi\tau_0}{5\tau_3}\right)^{2/3}. \tag{2.45}$$

For convenience, we define two constants:

$$A_0 = \frac{5}{256\left(\pi f_L\right)^{8/3}}, \quad A_3 = \frac{\pi}{8\left(\pi f_L\right)^{5/3}}, \tag{2.46}$$

which simplify the expressions for τ_0 and τ_3 as functions of M and η:

$$\tau_0 = \frac{A_0}{\eta} M^{-5/3}, \quad \tau_3 = \frac{A_3}{\eta} M^{-2/3}. \tag{2.47}$$

The chirp mass $\mathcal{M}$ is related to M and η via

$$\mathcal{M} = \eta^{3/5} M, \tag{2.48}$$

allowing τ_0 to be written purely in terms of $\mathcal{M}$:

$$\tau_0 = A_0 \mathcal{M}^{-5/3}. \tag{2.49}$$

The parameter space is bounded by the minimum and maximum allowed component masses, denoted m_{min} and m_{max}, and $m_1 > m_2$. Additional constraints on the total mass M may also apply. The signal low-frequency cutoff f_l (which is determined by the data quality at low frequencies) and the mass parameter space play a crucial role in determining the length of the waveform, and hence influences the metric, and the number of required templates N_t [83].

How to place templates? In the simple square template placement [83], templates are placed along $\eta = 0.25$ from the minimum to maximum mass considered. Then additional templates are placed along lines of constant chirp time τ_3, with the spacing between each row set based on the minimal mismatch condition and the distance between templates based on a square lattice. The quadratic expression in equation 2.41 is an equation for an ellipse, which defines each template. In simple terms, the metric tells us the shape and orientation of the "mismatch ellipses" around each template — these ellipses show how far you can move in parameter space before the overlap with the template drops below the minimal match. Ideally, templates should be placed along the principal directions of these ellipses (the eigenvectors of the metric) to cover the space efficiently without leaving gaps or adding too much overlap. In practice, using a square grid along the standard axes (τ_0 and τ_3) ignores the actual orientation of the ellipses. This is like trying to cover a tilted ellipse with a square grid: some ellipses overlap more than necessary, while others may leave small gaps. This is why this placement is non-optimal.

This was alleviated in [80] which introduced the optimal template placement method in two dimensions, the hexagonal placement. The templates are made to be perfectly adjacent to one another, reducing both overlap and gaps. However, such placement quickly becomes computationally infeasible in higher dimensions. For this reason, the random template placement method was introduced

in [85]. By giving up complete coverage of the high dimensional parameter space, they make a computationally efficient method with a large reduction of number of required templates. Ref. [86] introduces the stochastic template placement method that achieves a higher coverage for a given number of templates.

Number of templates required: The total number of discrete templates $\mathcal{N}$ is obtained by dividing the proper volume of the intrinsic parameter space by the proper volume associated with a single template, as determined by the chosen minimal match MM. The proper volume element is defined through the metric $g_{ab}(\lambda)$, which is itself constructed from second derivatives of the match, defined in terms of the noise-weighted inner product between templates. For more details please refer [81]. Since these inner products depend explicitly on the statistical properties of the detector noise, the metric—and hence the volume $\int d^N\lambda\sqrt{\det g_{ab}}$—encodes the influence of the noise power spectral density on the template spacing. As a result, the number of templates required to achieve a given minimal match depends not only on the dimensionality and extent of the intrinsic parameter space, but also on the noise characteristics of the detector through their effect on the metric.

$$\mathcal{N} = \frac{\displaystyle\int d^N\lambda\,\sqrt{\det g_{ab}(\lambda)}}{\left(2\sqrt{(1-MM)/N}\right)^N}. \tag{2.50}$$

Since these inner products weight each frequency component by the inverse of the noise power spectral density, a reduction in the PSD increases the contribution of the corresponding frequency bands to the match. As a result, the match becomes more sensitive to small variations in the intrinsic parameters, leading to larger metric components and an increase in $\det g_{ab}$. For a fixed minimal match, this enlarges the proper volume of the parameter space and necessitates a denser placement of templates, thereby increasing the total number of templates required. This effect is most pronounced when the PSD improvement occurs in frequency regions to which the waveform phase evolution is particularly sensitive. Since phase evolution is most sensitive to lower frequencies, the number of templates required by 3G detectors will increase significantly [87, 88].

2.6 Visualising the SNR Structure for GW150914

Conducting a full matched-filter search on sample data, although tempting for the sake of completeness and fun, is outside the scope of this work. In this section, we simply visualise the SNR structure in the (m_1, m_2) space by finding the maximum absolute SNR of complex SNR time series obtained by placing templates randomly in $U(m_1, m_2)$ without using the metric in equation 2.42. We emphasise that this is simply a visualisation exercise and does not follow rigorous template placing methods. This exercise is made to understand how the SNR varies across the (m_1, m_2) space and discuss any structure and behaviour that we see.

To visualise the structure of the matched-filter SNR for GW150914, we performed a study using a large set of simulated BBH waveforms. The analysis began with strain data from the Hanford detector, which was conditioned to remove low-frequency noise and cropped to mitigate edge effects. We then randomly sample component masses uniformly between 7 $M_\odot$ and 50 $M_\odot$, generating 50,000 distinct mass pairs. For each template, a corresponding waveform was generated and aligned in time with the strain data. The maximum absolute SNR, according to equation 2.21, for each template was recorded.

The resulting peak SNR values were then visualised as a two-dimensional colour map over the component mass parameter space as shown in Fig. 2.6. This representation reveals some expected structure: [i] we see bands of similar SNR values corresponding to lines of constant chirp mass as seen in chapter 1 Fig. 1.3, [ii] the peak SNR value reduces as be go down to lower chirp masses, and [iii] the SNR values are relatively high at higher chirp masses compared to lower ones given that the signal morphology does not vary as much with change in masses in this space. This also proves that it is not just a single template that might dominate the parameter space, but a whole variety of them above some detection threshold.

2.7 False Alarm Rate Estimation

The *FAR* is a key metric used in gravitational wave detection to quantify the rate at which noise fluctuations in the detector data produce triggers that exceed a chosen detection threshold. It is defined as the expected number of noise events per unit time at or above a specified value of detection statistic that mimic the morphology and amplitude of an astrophysical signal. If we have a bijective map between the FAR and a detection statistic, we can say that a lower FAR typically indicates a more statistically significant event.

In practice, the FAR is estimated empirically by constructing a background distribution of detection statistics under the null hypothesis, $\mathcal{H}_0$, in which the data contain no astrophysical signals. This is typically achieved by analysing segments of real detector data that are time-shifted or otherwise manipulated to remove any coincident astrophysical events, ensuring that only instrumental noise contributes to the statistics. By computing the detection statistic for a large number of such noise-only trials, one obtains an empirical distribution of the statistic under $\mathcal{H}_0$. The FAR associated with a candidate event is then determined by the fraction of background trials in which the detection statistic exceeds the value observed for the candidate. This approach allows a robust, data-driven estimate of the expected rate of false alarms, accounting for the non-ideal and non-Gaussian features present in real gravitational-wave detector noise.

We look specifically at the time-shifting/sliding method since we will use this procedure in future chapters. For coincident searches involving multiple detectors, the time series from each detector is artificially shifted by unphysical time delays that exceed the maximum light-travel time between sites

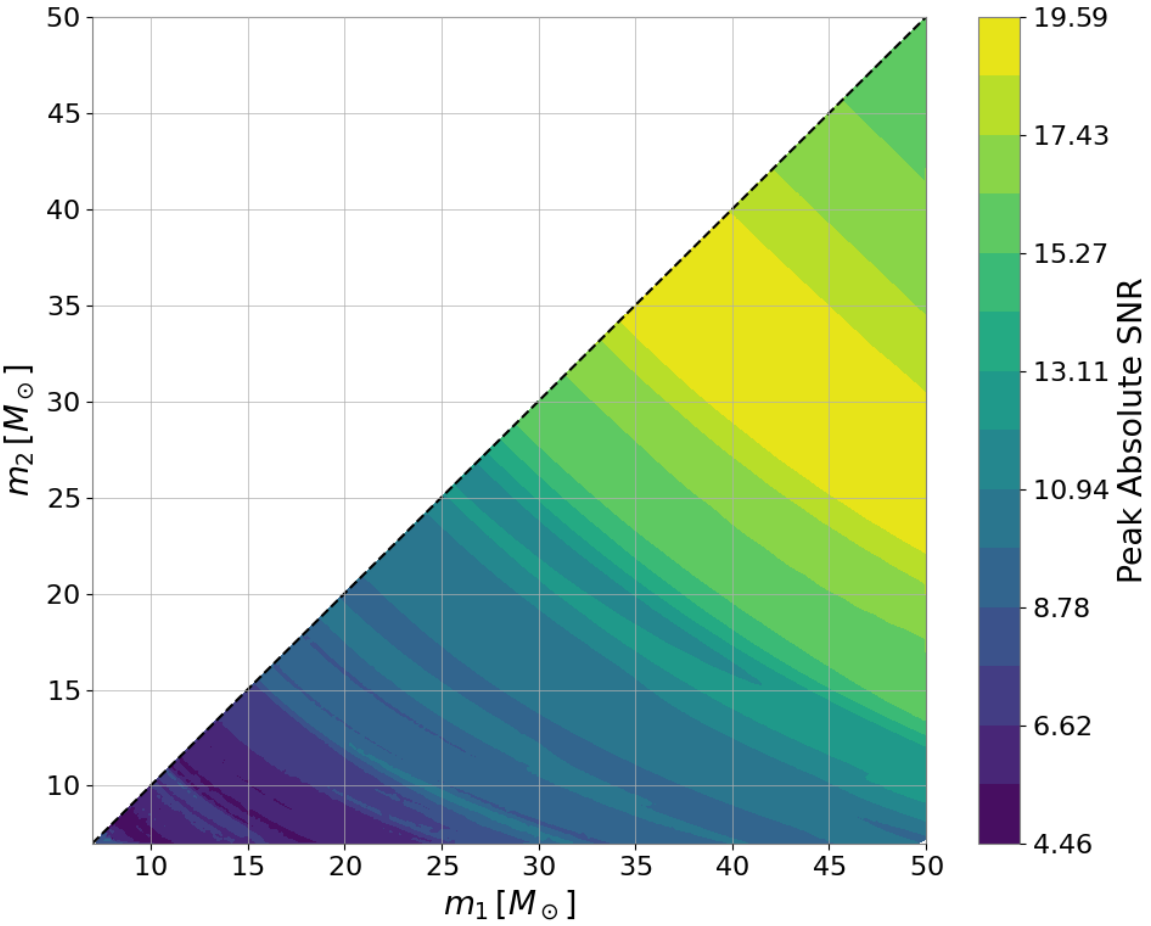


Figure 2.6: Contour plot showing the peak matched filter SNR across the (m_1, m_2) parameter space for the GW event GW150914, using a template bank uniformly placed in component masses. A clear and localised peak in SNR is visible near the true source-frame masses, demonstrating the effectiveness of matched filtering in recovering the signal. The dashed black line denotes the region where $m_1 = m_2$, and axes are limited to the prior bounds of $7\,M_\odot \leq m_2 \leq m_1 \leq 50\,M_\odot$.

(e.g., greater than ≈10 ms for the two LIGO detectors). Since genuine gravitational wave signals will not be simultaneously present in any shifted data streams, any coincident triggers observed in these time-shifted data are attributed to noise coincidences. Repeating this procedure across many independent time-shifts yields an empirical distribution of background triggers. This procedure cannot be applied indefinitely and there is a known saturation effect to its efficacy given limited PSD realisations and glitch data within the background data [89]. Moreover, the time slid data contains astrophysical signals within each detector data stream that will cause contamination to the estimating background distribution [90].

2.8 Signal Consistency Tests and Vetoing

Signal consistency tests address the shortcomings of matched filtering to non-Gaussian noise by checking whether a candidate trigger matches the expected morphology of an astrophysical signal. Inconsistent triggers are vetoed or down-weighted, yielding reweighted detection statistics that improve robustness against glitches while preserving sensitivity. We look at some popular method below.

FINDCHIRP, χ^2 test [91, 92]: This test partitions the frequency-domain template waveform into p bins such that for an authentic signal matching the template, each bin should contribute equally to the total SNR, within statistical fluctuations. The χ^2 statistic is computed by comparing the observed contributions with the expected equal values, yielding a measure of the deviation. Under the null hypothesis of Gaussian noise and a matching signal, the resulting χ^2 statistic follows a chi-squared distribution with $2p-2$ degrees of freedom. Large values indicate inconsistency with the signal model, typically arising from glitches or parameter mismatch. This statistic is commonly used to reweight the SNR in detection pipelines, penalising triggers that exhibit significant deviation from the expected frequency evolution.

Reweighted Ranking Statistic: The reweighted chi^2 test [93], used in current GW search pipelines, modifies the standard matched filter SNR based on the χ^2 value to produce a detection statistic that balances sensitivity and glitch rejection. They use the usual χ^2 statistic to ensure that the frequency evolution is consistent with a binary merger signal and mention that they can describe the *density* of noise triggers using a combination of the SNR and χ^2 although they cannot analytically predict the χ^2 distribution for glitch triggers in each detector. The define this reweighted SNR as follows:

$$\tilde{\rho} = \begin{cases} \dfrac{\rho}{\left[1+\frac{(\chi^2)^{p/2}}{2}\right]^{1/p}} & \text{for } \chi^2 > 1 \\ \rho & \text{for } \chi^2 \leq 1 \end{cases} \tag{2.51}$$

where they define p to be 6 [94].

Blip Glitch Veto: Blip glitches are short noise transients that have high morphological similarity with high mass BBH mergers. Their origin has largely been unknown and they cause severe reduction in detection sensitivity with increasing chirp mass, if left unchecked [95]. A simple veto that has been used by matched filtering pipelines for this purpose is the sine-Gaussian veto [66]. With the assumption that blip glitches can be modelled as sine-Gaussians, sine-Gaussian tiles are placed around the candidate time at frequencies above the template's termination frequency. The discriminator is defined as the sum of the squared SNR values from these tiles, with tile placement (30–120 Hz above the cutoff, spaced by 15 Hz, $Q = 20$) chosen empirically. This measure is then combined with the reweighted SNR to form an improved single-detector statistic [66].

Machine Learning Vetoes: Machine learning-based tests have also been introduced, where trained

classifiers evaluate signal morphology or spectrogram features to distinguish real signals from glitches. For example: [i] the χ^2 signal consistency test has been tuned for maximum efficacy against a large population of noise transients to aid in the search for intermediate mass black hole binaries [96], [ii] to learn the full noise transient morphologies to aid against blip glitches [97], and [iii] machine learning to categorise families of glitches to aid better glitch rejection and potentially identify their source [98, 99]. These methods offer improved performance in highly non-Gaussian conditions, but often require careful training and interpretability analysis.

This section is biased toward the PyCBC way of dealing with signal consistency and glitch vetoes, given that they play a major role in future chapter. However, we refer the reader to other matched filtering pipelines like GstLAL [100] or MBTA [101] for other methods.

2.9 Handling Glitches and Non-Gaussian Noise

To illustrate a critical limitation of matched filtering, we applied the same procedure as in section 2.6 to a segment of data known to contain a "Whistle" transient instrumental glitch. The resulting SNR structure in the (m_1, m_2) space revealed several high-SNR peaks comparable to those observed for GW150914. This underscores the well-known fact that a high matched filter SNR does not in itself constitute evidence for a genuine astrophysical signal. Glitches can produce significant overlap with some templates, particularly those that share morphological features such as rapid frequency evolution or short duration. Consequently, robust detection pipelines must supplement matched filtering with additional signal consistency tests—such as the χ^2 modification to the ranking statistic—and rely on a coincidence framework across multiple detectors to mitigate false positives arising from non-Gaussian noise artefacts. This result emphasises the need for comprehensive background estimation procedures and the incorporation of signal-based vetoes in any realistic search pipeline.

2.10 Limitations and Bottlenecks

The definition of an optimal receiver is often muddied when dealing with real-world data [61], such as for the detection of unknown GWs in detector noise, making the matched-filter SNR a non-optimal detection statistic. We can deconstruct the major deviations from the optimal detection scenario into the following: unknown signal parameters, non-stationary noise, uncertainty in the estimates of noise PSD and non-Gaussian noise artefacts. All of these factors contribute to the departure of matched-filtering from Neyman-Pearson optimality [102, 103, 104, 105, 106] to varying degrees [61, 68].

Say we have the time series data $d = n+h$ collected from a GW detector, where n is the combination of all possible noise sources and h is the sum of the GW response of the detector. In the case of an unknown signal, the optimal detection statistic would be obtained by marginalising the likelihood ratio $\Lambda(d|\theta)$ over all unknown signal parameters, θ, by integrating $\Lambda(d|\theta)$ over θ [68]. The likelihood

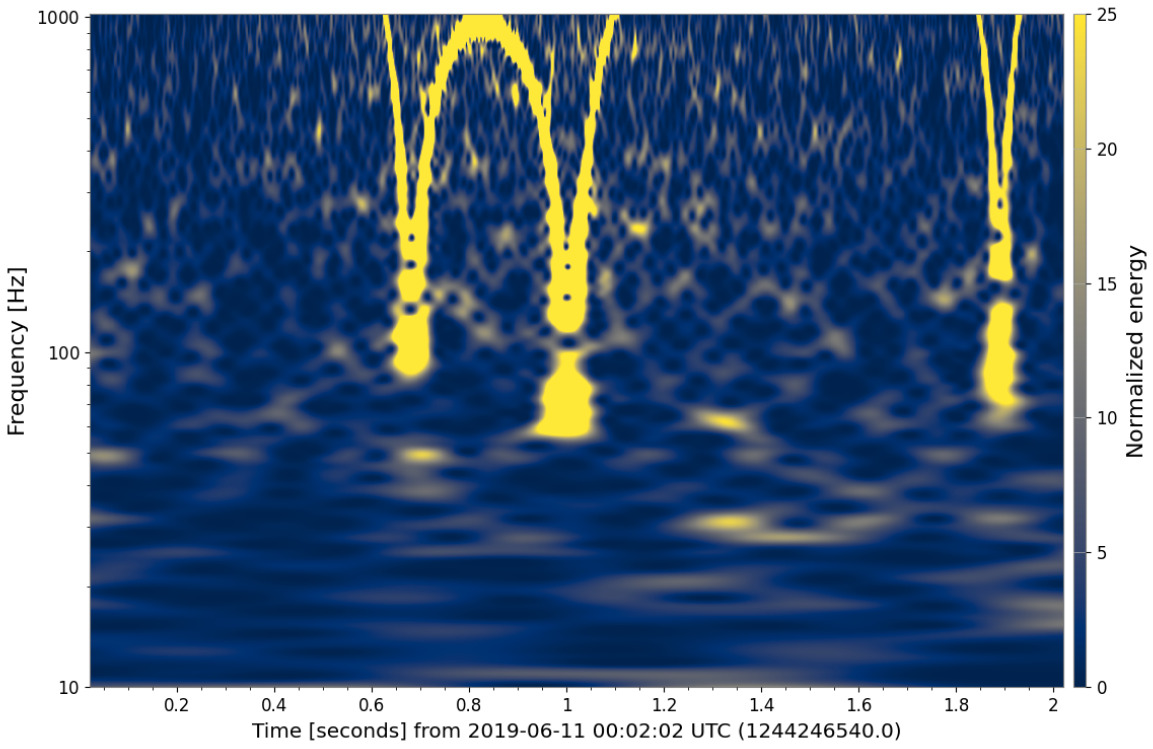


Figure 2.7: Q-transform of a "whistle" glitch occurring near the specified GPS time. The glitch exhibits a clear upward chirping morphology in the 100–1000 Hz range and is visually distinct from typical compact binary coalescence signals. Despite its strong presence in the time-frequency domain, such glitches can produce a high matched-filter SNR when convolved with certain templates, leading to potential false triggers.

ratio will generally be peaked about its maximum, given that the log likelihood ratio is a linear function of the signal model. Thus, maximising $\Lambda(d|\theta)$ over θ is expected to be a good approximation of the marginalised likelihood ratio, up to a constant rescaling factor [68]. However, this approximation is a non-optimal treatment of the parameter space.

The template bank used for matched filtering over the multidimensional GW signal parameter space is constructed such that at least one template in the bank is close enough to a potential astrophysical signal, while not losing more than a predetermined fraction of the SNR under ideal conditions [107, 108, 109]. Although this condition leads to a computationally manageable template bank for CBC in the current detector era, template bank based approaches will become a lot more computationally expensive in the future when longer templates are needed. Although ML will likely be orders of magnitude less expensive, a speed comparison would be near moot since given enough computational resources, matched filtering will likely be faster than real time.

Deviations of the detector noise from independent and identically distributed (IID) Gaussian noise, in the time domain, include coloured noise, non-stationary noise and the presence of non-

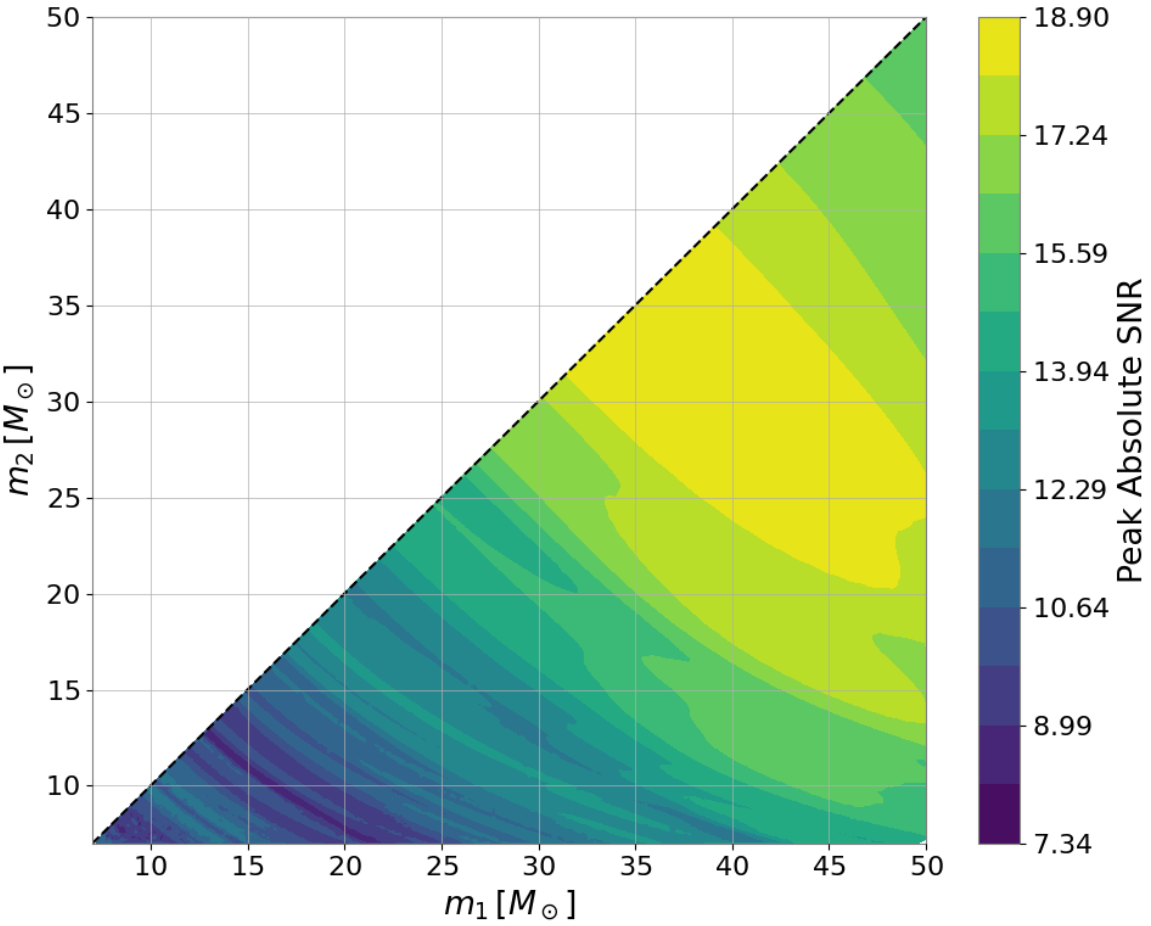


Figure 2.8: Contour plot showing the peak matched filter SNR across the (m_1, m_2) parameter space for the whistle glitch shown in Fig. 2.7, using a template bank uniformly placed in component masses. A localised peak in SNR is visible at multiple masses, demonstrating the limitations of the matched filtering SNR in handling non-Gaussian noise artefacts. The dashed black line denotes the region where $m_1 = m_2$, and axes are limited to the prior bounds of $7\,M_\odot \leq m_2 \leq m_1 \leq 50\,M_\odot$.

Gaussian transient instrumental artefacts arising from known and unknown sources [110, 111, 112]. For a known noise covariance matrix, we can reduce the problem of detecting a signal in coloured noise to the problem of detecting a signal in IID Gaussian noise. In practice, this is typically performed by whitening the signal in the frequency domain by estimating the PSD using the off-source or on-source methods. Both methods assume that the estimated PSD is exact and that the noise is stationary and Gaussian. These assumptions do not hold in either case, but more so for the off-source method, since it requires a longer duration of data than the on-source method [113].

The presence of non-Gaussian transient instrumental artefacts (or glitches) can induce certain templates to produce high SNR outputs, making the matched-filter SNR highly non-optimal as a detection statistic [68]. This is currently mitigated using witness sensors [114], waveform consistency checks [93, 115, 91, 116] and time coincidence tests [2]. However, artefacts of unknown origin that are similar in time and frequency evolution to GW signals are difficult to veto out. This is especially true

in the case of short-duration GW signals, where the background might be contaminated by numerous short-duration transient noise artefacts [30]. Although there exist waveform consistency tests to identify these types of noise, they are based on assumptions that might not always hold true and would thereby only target a small subset of all possible glitches [66]. These assumptions include modelling short-duration noise transients as sine-Gaussian, targeting a particular frequency range for discriminative features and using parameter priors for the assumed noise model that may not completely coincide with the truth [66]. Such assumptions have to be made, since it is impossible to accurately model noise of unknown origin or create a waveform consistency test that is reliable for all future observed glitches. The utmost that we can do to mitigate the effects of unknown noise on detection efficiency, is to incorporate all prior non-Gaussian noise data into a generalised waveform consistency test that is capable of being continually updated. Deep learning is a promising framework that is capable of handling problems of this nature [97, 96].

Aside from the described non-optimality conditions for matched filtering in the context of current gravitational-wave detectors, it is important to acknowledge the adequacy of template bank-based search methods in consideration to future detectors. Future ground-based detectors like Einstein Telescope [49] and Cosmic Explorer [50] alongside space-based detectors like LISA [51] and TianQin [52], with their differences in detectable frequency bands, will explore a wide range of new astrophysical sources. The increased sensitivity that accompanies the third generation of ground-based detectors will ensure the importance of parameters like higher-order modes and eccentricity in searches [88]. Template-based techniques will thus demand a dramatic increase in required computational resources if used in its current state. Increased low-frequency sensitivity also elicits the ability to observe multiple/overlapping signals [117], which leads to detectability issues due to confusion noise [118] and the general non-optimality of matched filtering [119]. One of the exciting new sources that LISA and TianQin will be observing is Extreme Mass-Ratio Inspirals, which according to estimates would require approximately 10^{40} templates or more to perform a fully coherent matched filter search for year long inspiral waveforms [120]. Although methods have been developed to deal with this problem, it is still far from ideal.

2.11 Summary and Open Questions

This chapter has presented a systematic overview of matched filtering as the primary technique for detecting gravitational waves. Starting from the statistical framework of model comparison and the Neyman–Pearson lemma, matched filtering was introduced as the optimal linear method for extracting signals of known form from noisy data. Its application to gravitational-wave searches was then discussed in detail, with emphasis on the need for careful data preprocessing, including windowing, whitening, and the removal of instrumental artefacts through bandpass and notch filters.

A crucial component of the pipeline is the construction of template banks, which discretise the continuous parameter space of potential signals. This ensures coverage of the relevant astrophysical space, but at the cost of increased computational requirements. The behaviour of the matched filter was illustrated through the visualisation of the SNR structure in a real GW event, highlighting both its strengths in detecting weak signals and its vulnerabilities to noise artefacts. FAR estimation, signal consistency tests and vetoing strategies were then discussed as a means of distinguishing genuine astrophysical signals from transient disturbances.

Despite the theoretical optimality of matched filtering under idealised conditions, several challenges remain unresolved:

- **Model Assumptions:** To what extent do deviations from stationarity, non-Gaussianity, and waveform inaccuracies degrade detection confidence?
- **Noise Artefacts:** How can we reliably distinguish between glitches and astrophysical signals, especially when both may yield high SNR under naive matched filtering?
- **Computational Cost:** Is there a more efficient way to cover large parameter spaces while maintaining detection fidelity?

These open questions suggest that while matched filtering provides a principled foundation for GW detection, it may not always be sufficient in isolation—particularly in the presence of real detector noise and large template banks. In chapter 4, we explore how machine learning methods can address some of these challenges. By learning from both signal and noise distributions, these approaches offer a flexible and data-driven framework for enhancing sensitivity, accelerating searches, and improving robustness to out-of-distribution effects. We aim to assess whether such models can supplement or even surpass classical methods in detecting and classifying GW events.

3

Machine Learning for Detection

In this chapter, we explore the possibility of machine learning as a viable alternative to matched filtering for the gravitational wave detection problem. To understand the subtleties of the decision-making process in machine learning models, we build a neural network *ab initio*, starting from a single neuron and incrementally increasing its complexity. We hope this allows the Reader to experience and appreciate the emergence of complex behaviours from simple building blocks, and get a deeper understanding of the choices made in future chapters. We encourage the Reader to follow along using this HTTPS URL.

3.1 The Promise of Deep Learning

Machine learning is a promising alternative to tackle the detection problem in non-ideal settings. This is corroborated by previous machine learning literature, some of which are chosen here: [121, 122] exhibit the capability of ML in detecting low SNR signals, [123] classifies accurately on highly complex and large datasets, and [124, 125] provide theoretical insights into why ML generalises well. It has also been shown that neural networks have the capability to approach Neyman-Pearson optimality for the noisy signal detection problem when trained with a sufficient amount of data on a large enough network under appropriate training strategies [126, 127]. Thus, the ML literature has more than sufficient evidence for its capability to handle GWs in detector noise when used appropriately. However, building a detection model that is capable of approaching the performance provided by the theoretical guarantees of matched filtering requires careful consideration. In this chapter, we will primarily focus on how a neural network works, where and why it fails and what we can do to fix or mitigate these issues.

Before examining the inner workings of neural networks, it is important to clarify what we expect the model to learn and how this learning process differs from earlier approaches to decision-making. Traditional rule-based systems, such as matched filtering, rely on explicitly defined analytical forms that map the input to a ranking statistic, which is then used for classification or detection. Classical machine learning aimed to replace this analytical mapping with a data-driven one, learning from simplified forms of the input data, to account for the many deviations of real data from the idealised settings where rule-based systems performed well. These simplified forms, referred to as *features*, were

hand-crafted by domain experts and the burden of representation design was therefore placed on prior knowledge rather than on the model itself. The performance of classical ML is thus constrained by the quality and completeness of the prior understanding embedded in these hand-crafted features. In contrast, representation learning—which will be the overarching theme of this thesis—aims to learn both the decision boundaries and the feature representations directly from data. This shift from manually defined features to data-driven representations marks a fundamental change in how detection problems are approached, and frames the discussion of the types of decision-making systems that we turn to next. Refer to Fig. 3.1 for a flowchart comparison of the three different approaches to decision-making described here.

3.2 Representation Learning

Representation learning seeks to automatically *discover* useful features or transformations of the input data that facilitate the desired task. This approach enables models to extract hierarchical and abstract representations directly from raw observations, potentially capturing patterns or structures that may be inaccessible to manual feature engineering. Such learned representations can offer improved generalisation to previously unseen data and adaptability to complex or high-dimensional domains, making them particularly suitable for applications where the underlying data distributions are difficult to specify analytically [128].

Example 2: Why Representation Learning Matters?

A simple example illustrating the importance of representation learning can be constructed using two separable datasets: a noisy circular distribution and a compact Gaussian cluster (see Fig. 3.2). The aim is to separate these two datasets using a simple model that assumes that the output can be expressed as a linear combination of the input features. Formally, if the input features are $x_1, x_2, \ldots, x_n$, then the model predicts an output y as

$$y = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b, \tag{3.1}$$

where w_i are the model parameters (weights) and b is an additive bias term. We call this a linear model.

In Cartesian coordinates (see Fig. 3.2 *left*), the two classes overlap in a manner that is non-linearly separable, making classification with a linear model ineffective. However, by transforming the same data into polar coordinates (see Fig. 3.2 *right*), the classes become linearly separable, allowing the same linear model to achieve near-perfect classification performance [129].

This example highlights a core principle of representation learning: the choice of representation directly affects the separability and structure of the data in feature space. In traditional approaches, such transformations must be hand-engineered based on domain knowledge. In contrast, modern

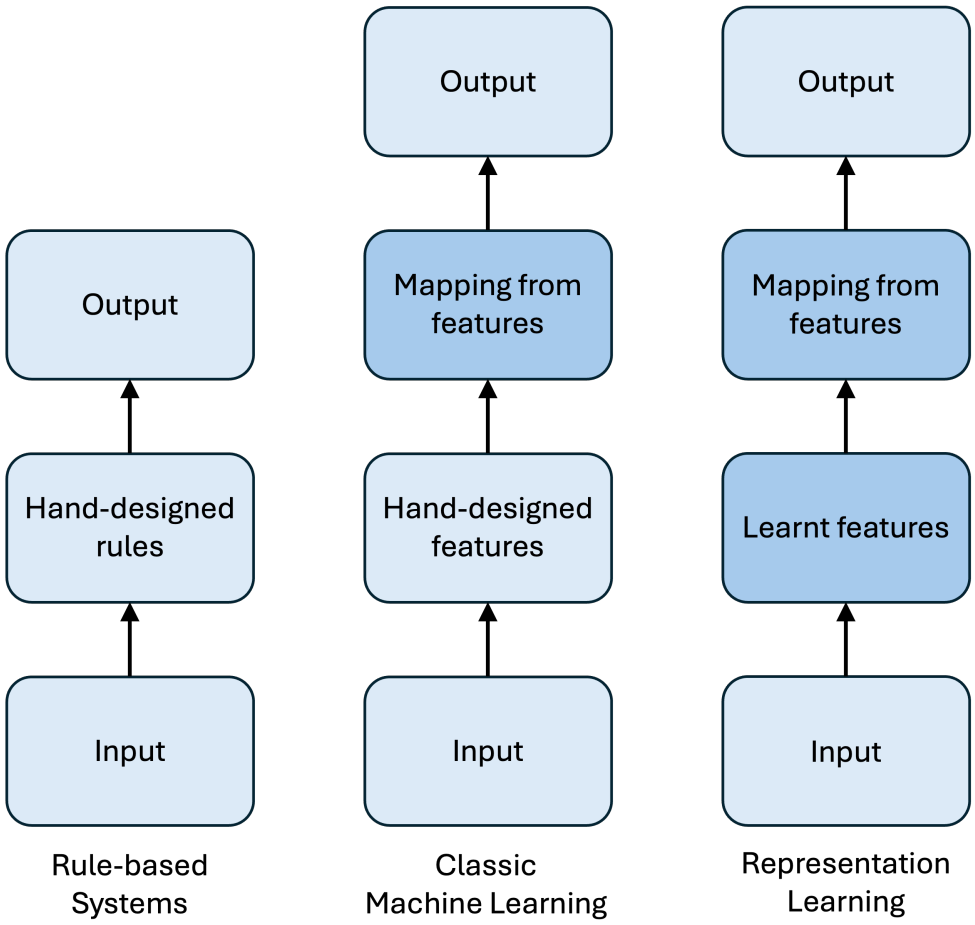


Figure 3.1: Flowcharts comparing different types of systems used for the detection problem. Dark shaded boxes indicate components that are able to learn from data.

representation learning methods aim to learn such transformations automatically, enabling the model to uncover and exploit structures that facilitate accurate and generalisable decision boundaries. This becomes particularly useful in high-dimensional regimes, where analytical approaches often fail.

With the primary goal of learning robust representations of data in mind, let us now move on to building a neural network from scratch.

3.3 Emulating a Neuron

Artificial neural networks are inspired by the structure and function of biological neurons, abstracting their core computational principles into a mathematical form suitable for optimisation. A biological neuron receives signals from other neurons through its dendrites (visualise the neuron as a node in a network and the dendrites as links), aggregates these inputs, and, if the combined signal exceeds a threshold, produces an output along its axon. This process can be represented as a sequence of three operations: weighted summation of inputs, addition of a bias term, and the introduction of non-linearity for decision-making [130, 131]. Let us now concretely define this process for a single artificial

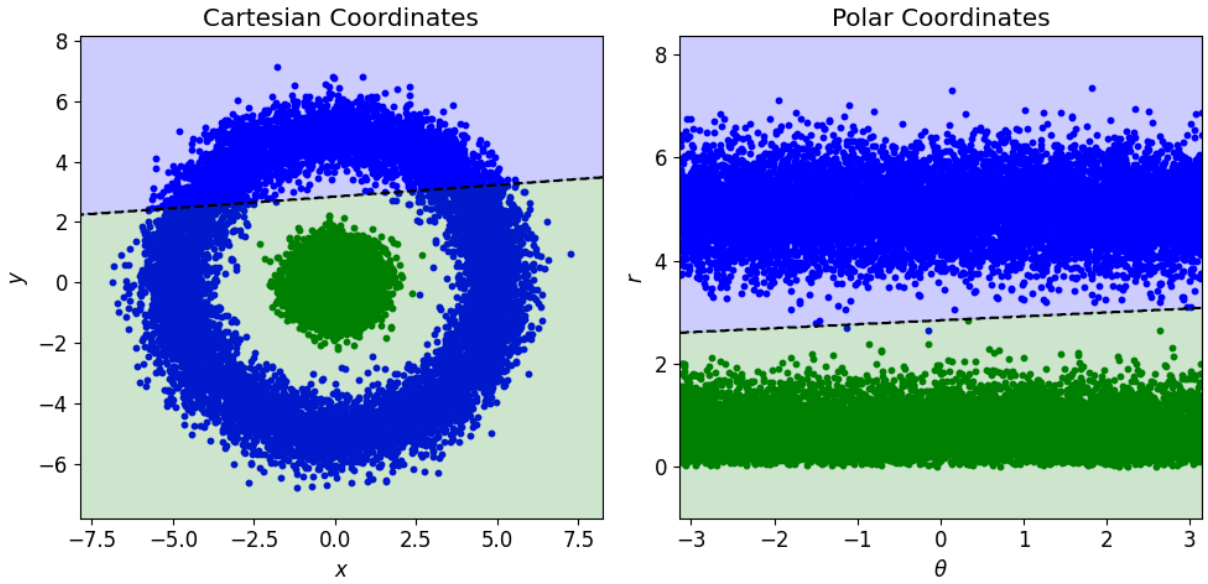


Figure 3.2: Decision boundaries of logistic regression applied to the same set of points in Cartesian and polar coordinates. The dashed black line shows the decision boundary learnt from the training data. The shaded regions represent the class label applied to the 2D parameter space, coloured based on the colour of the cluster.

neuron. Say, in an artificial neuron, we consider an input vector

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top \in \mathbb{R}^n, \tag{3.2}$$

where each component x_i represents the input from the i-th source. Associated with these inputs is a learnable weight vector

$$\mathbf{w} = (w_1, w_2, \ldots, w_n)^\top \in \mathbb{R}^n, \tag{3.3}$$

where w_i denotes the strength of the connection corresponding to x_i. In addition, the neuron has a scalar bias parameter $b \in \mathbb{R}$, which allows the decision boundary to shift away from the origin. The pre-activation z of the neuron is defined as the affine transformation of the input:

$$z = \mathbf{w}^\top \mathbf{x} + b = \sum_{i=1}^{n} w_i x_i + b. \tag{3.4}$$

This scalar pre-activation is then passed through a non-linear activation function,

$$f : \mathbb{R} \to \mathbb{R}, \tag{3.5}$$

yielding the neuron's output

$$y = f(z). \tag{3.6}$$

The role of the activation function $f(\cdot)$ is to introduce non-linearity into the model, which is

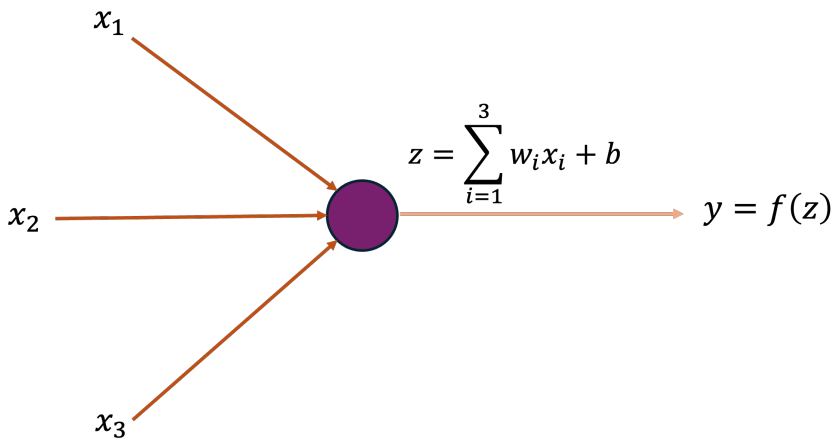


Figure 3.3: Depiction of a single artificial neuron with inputs (x_1, x_2, x_3). The inputs are first put through the affine transformation shown in equation 3.4 to produce z. Finally, non-linearity is introduced in the neuronal decision using the activation function f to produce the output $f(z)$.

essential for the network to approximate complex, non-linear mappings between inputs and outputs. Common activation functions include:

$$\textbf{Sigmoid function:} \quad f(z) = \sigma(z) = \frac{1}{1+e^{-z}}, \quad \sigma(z) \in (0,1). \tag{3.7}$$

$$\textbf{Hyperbolic tangent:} \quad f(z) = \tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}, \quad \tanh(z) \in (-1,1). \tag{3.8}$$

$$\textbf{Rectified Linear Unit:} \quad f(z) = \max(0, z). \tag{3.9}$$

The sigmoid function function squashes real-valued inputs into the range $(0, 1)$, making it suitable for probabilistic interpretations. The tanh function, on the other hand, is zero-centred, meaning negative inputs map to negative outputs and positive inputs to positive outputs, which has been shown to help accelerate convergence during model training in some cases. Finally, the rectified linear unit (usually expressed in the shorthand form as ReLU) preserves positive values while setting negative values to zero. It is a simple yet popular activation function due to its computational efficiency.

Thus, the computation of an artificial neuron can be compactly expressed as

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right), \tag{3.10}$$

which serves as the fundamental building block of neural networks (see Fig. 3.3 for a depiction of a single neuron). Each of these activation functions have their own advantages and disadvantages. We will go through them in more detail after we get a fuller picture of the inner working of a neural network.

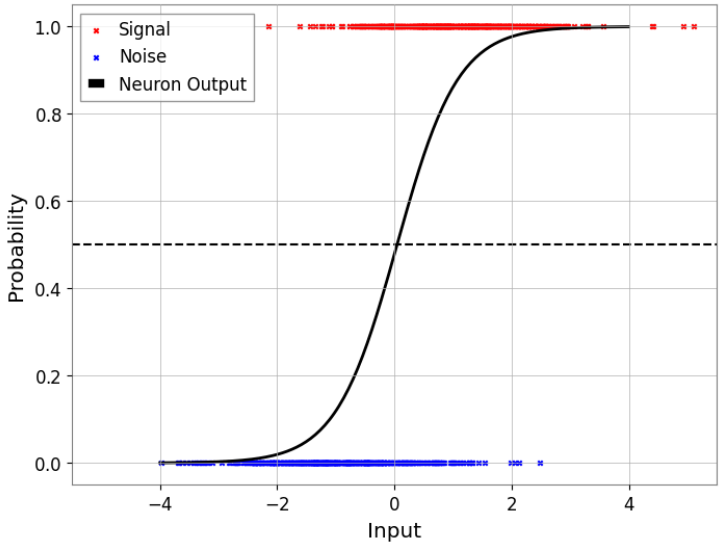


Figure 3.4: Scatter plot of signal and noise samples drawn from Gaussian distributions, together with the output probability curve (showing the probability that a sample lies in the signal class) of a trained single-neuron classifier. The dashed line indicates the decision threshold at probability 0.5.

Example 3: Can a single neuron approximate Neyman-Pearson optimality?

Let us construct a simple binary classification task using synthetic data sampled from two Gaussian distributions: one representing signal with mean +1, and the other representing noise with mean −1. A single artificial neuron is defined, consisting of the linear transformation, followed by a sigmoid activation function, to separate the two classes. The weights and biases of a neural network (or even a single neuron) are optimised iteratively by showing the neuron lots of inputs $\mathbf{x}$ such that it can approximate an unknown function mapping to the outputs y. This is known as *training*.

During training, the model learns optimal weight and bias parameters for the linear mapping. After training, the decision boundary learned by the neuron can be visualised by plotting the input samples alongside the neuron's output probability curve across the input range. Refer to Fig. 3.4 to visualise the outcome of such a training process. We see that the decision threshold that separates the two Gaussian distributions is almost exactly at 0.5. We know from section 1.2 that this is indeed Neyman-Pearson optimal. With this, we prove that a single neuron with sufficient training can approximate simple and optimal decision-making scenarios.

Although the definition of the neuronal process given in equation 3.10 significantly simplifies the biological processes, it preserves the key idea of transforming input signals into a non-linear output based on learned parameters. By arranging such neurons to create a network and training them, artificial neural networks can approximate even complex decision boundaries under suitable conditions. This capability forms the foundation for modern deep learning.

3.4 Emulating a Network of Neurons

Emulating a network of neurons refers to constructing a model that approximates the behaviour of biological or artificial neurons arranged in a network, as it is in reality. Instead of modelling every biophysical detail, the goal is to capture the essential information processing capabilities of interconnected neurons.

In the artificial neural network paradigm, each neuron is represented as a unit that receives inputs $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, computes a weighted sum with parameters $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ and bias b, and applies a non-linear activation function $f(\cdot)$. When multiple neurons are arranged in layers and interconnected with each other as shown in Fig. 3.5, their outputs are passed as inputs to subsequent neurons, forming a directed acyclic graph of computations. A network with L layers can be described recursively as

$$\mathbf{h}^{(l)} = f^{(l)}\left(W^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right), \quad l = 1, 2, \ldots, L, \tag{3.11}$$

where $\mathbf{h}^{(0)} = \mathbf{x}$ is the input, $W^{(l)}$ and $\mathbf{b}^{(l)}$ are the learnable weight matrix and bias vector of layer l, and $f^{(l)}(\cdot)$ is the activation function applied elementwise. The final layer $\mathbf{h}^{(L)}$ constitutes the network output. Without non-linearity, a neural network composed solely of affine transformations would collapse to a single affine transformation. Concretely, if each layer l in an L-layer neural network applies an affine transformation

$$x^{(l+1)} = W_l x^{(l)} + b_l. \tag{3.12}$$

Composing two such layers gives

$$x^{(l+2)} = W_{l+1}(W_l x^{(l)} + b_l) + b_{l+1} = (W_{l+1}W_l)x^{(l)} + (W_{l+1}b_l + b_{l+1}), \tag{3.13}$$

which is again affine. By induction, a network composed solely of affine layers reduces to

$$x^{(L)} = W x + b, \tag{3.14}$$

for some matrix W and vector b. Non-linear activation functions are therefore required to prevent this collapse and to enable the representation of non-linear mappings.

Emulating a network of neurons abstracts away biological complexity to focus on the emergent computation arising from simple units and their connectivity. This emulation allows one to approximate arbitrary non-linear functions, a property formalised by the *universal approximation theorem* [132], which states that sufficiently large neural networks can approximate any continuous function on a compact domain to arbitrary precision. The universal approximation theorem holds because neural

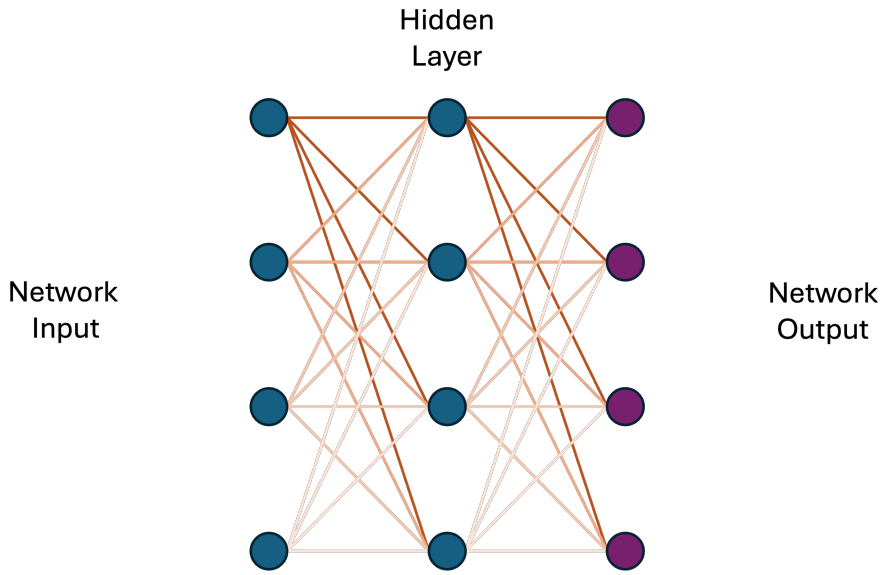


Figure 3.5: A simple network of neurons arranged in equally sized layers. This architecture was historically called the multi-layered perceptron.

networks form linear combinations of non-linear basis functions. For a single hidden layer network,

$$f(x) = \sum_{i=1}^{N} a_i \, \sigma(w_i^\top x + b_i), \tag{3.15}$$

each term $\sigma(w_i^\top x + b_i)$ introduces non-linearity and defines a flexible basis function. By varying the parameters (w_i, b_i), these functions can be positioned and scaled across the input space. The set of such functions is dense in the space of continuous functions on compact domains, meaning that for any continuous target function g and any $\varepsilon > 0$, there exists N such that

$$\sup_{x \in K} |f(x) - g(x)| < \varepsilon. \tag{3.16}$$

Thus, by combining linear superposition with non-linear activations, neural networks can approximate arbitrary continuous functions to arbitrary accuracy. We will see examples of this in future sections.

Example 4: Can multiple neurons learn complex decisions?

This example demonstrates the use of multiple layers of neurons to learn a non-linear decision boundary between two classes of data in two dimensions. Let us consider input data drawn from two Gaussian distributions, one centred at $(1, 1)$ and the other at $(-1, -1)$, making the problem linearly separable only with difficulty in the presence of noise. A small neural network is constructed with an input layer of two neurons (corresponding to the two features, the x and y coordinates of the samples in each Gaussian), a hidden layer of four neurons with sigmoid activation functions, and a single output neuron that produces a probability via a sigmoid transformation.

The hidden layer introduces a non-linear transformation of the input space, enabling the network to learn curved or complex decision surfaces rather than a single straight line. During training, the network weights and biases are optimised to minimise the error between predicted and true class labels. This error is known as the loss in ML literature. As training progresses, the network effectively partitions the input space into regions corresponding to the two Gaussian clusters.

By visualising the trained decision surface over the 2D feature space in Fig. 3.6, one can see how layers of neurons allow the model to approximate flexible decision boundaries, demonstrating the core principle of representation learning: transforming raw inputs into more useful intermediate representations that simplify the classification task.

Network topologies are a fundamental concept in computer science and communication systems, describing how nodes are connected and how information flows between them (see Fig. 3.7 for different types of basic network topologies). What makes them interesting in the context of neural networks is that similar subnetworks can spontaneously emerge within a large, arbitrarily connected architecture. When certain neurons are deactivated (e.g., through sparsity or thresholding), the effective connectivity of the network can resemble classical topologies such as rings, stars, or trees. This perspective suggests that even within a dense neural network, the functional computation at any given point in training may be governed by structured, topology-like subnetworks, potentially influencing generalisation, robustness, and computational efficiency. This is an active and exciting area of research that aims for the explanability of the learning process of neural networks.

3.5 Loss Functions

In this section, we will describe more in detail the training *loss* described earlier. The loss is the crux of the learning process and provides the *motivation* for the network to do what we want it to do during training. Thus, describing this loss requires proper consideration of the problem and sometimes expert domain knowledge.

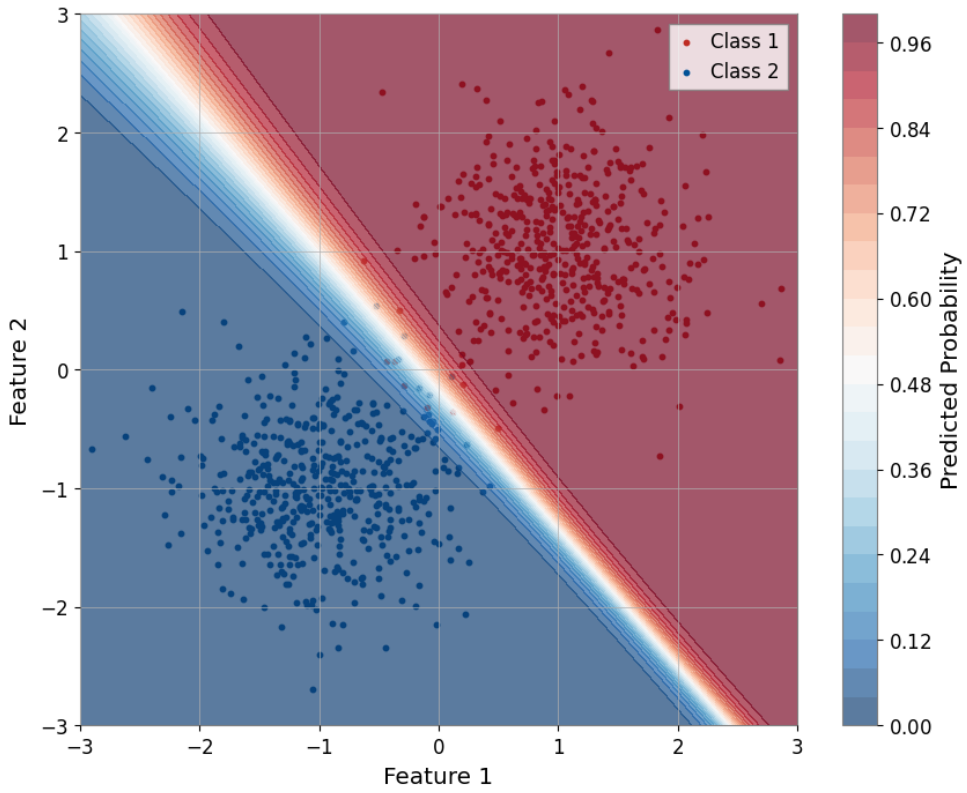


Figure 3.6: Decision surface learned by a two-layer neural network trained to separate two Gaussian-distributed classes in a two-dimensional feature space. The background colour map shows the predicted probability assigned to class 1, with the data points from both classes overlaid.

The loss function, also referred to as the cost or objective function, quantifies the discrepancy between a model's predictions and the true target values. It provides the optimisation signal that drives parameter updates during training. By minimising the loss function with respect to the network's parameters, the learning algorithm encourages the model to produce outputs that more closely align with the desired outcomes. The choice of loss function is therefore critical, as it encodes task-specific assumptions about the data and the form of the optimal solution.

Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ sampled from an underlying distribution $p(\mathbf{x}, y)$, the model $f_\theta(\mathbf{x})$ with parameters θ is trained by minimising a loss function $\mathcal{L}(y, f_\theta(\mathbf{x}))$ that quantifies the discrepancy between the predicted and true outputs. Formally, the optimisation problem is

$$\theta^* = \arg\min_{\theta} \; \mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)} \big[\mathcal{L}(y, f_\theta(\mathbf{x}))\big], \tag{3.17}$$

which in practice is approximated over the entire dataset:

$$\theta^* \approx \arg\min_{\theta} \; \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\big(y_i, f_\theta(\mathbf{x}_i)\big). \tag{3.18}$$

Once trained, the model weights are frozen (cannot be updated when data is passed through the net-

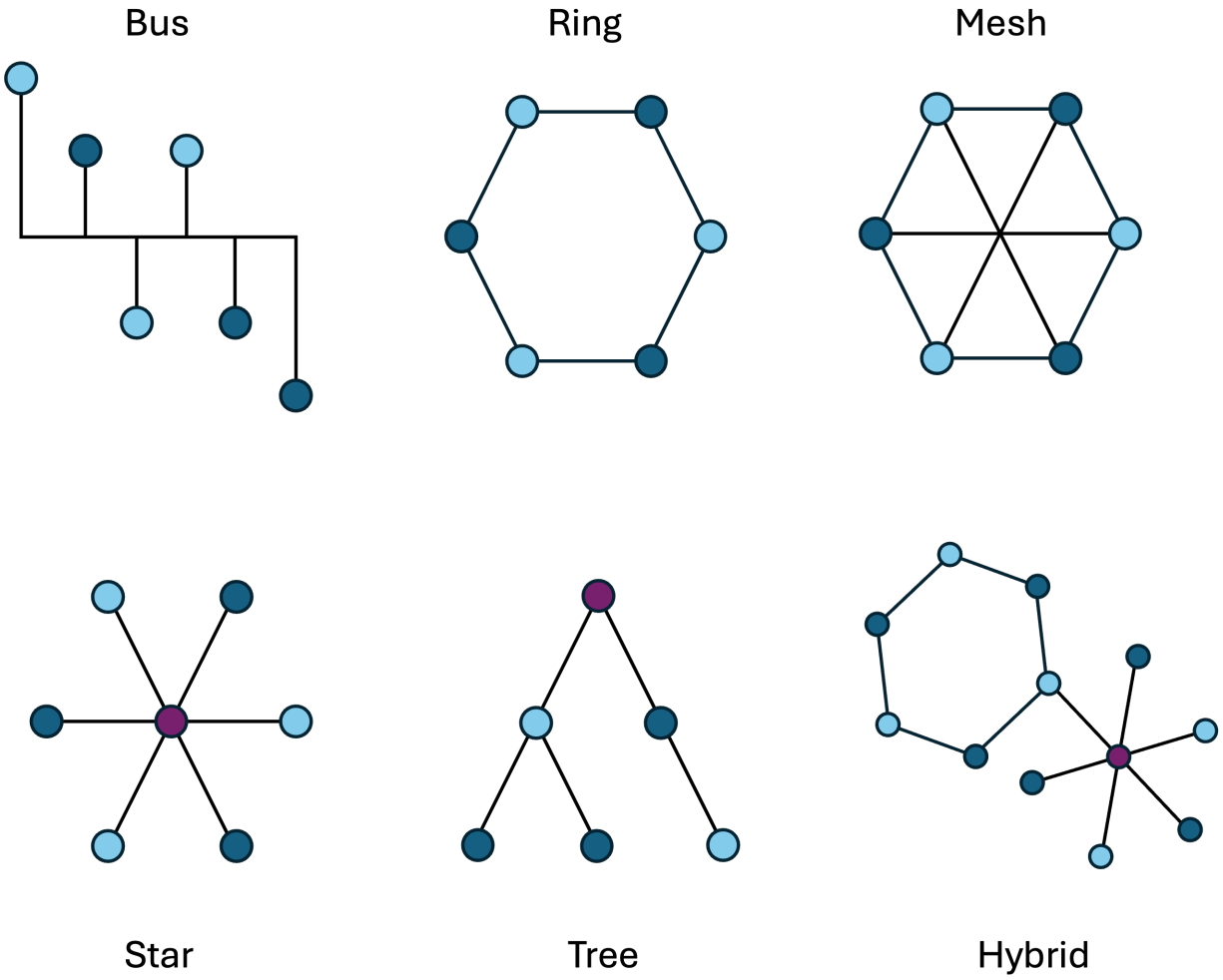


Figure 3.7: Schematic representations of common network topologies: bus topology, ring topology, star topology, tree topology, mesh topology, and hybrid topology. This diagram is a reproduction of that seen in this https url.

work) and the model is run under testing/inference mode on unseen data to test for generalisation capabilities.

In machine learning, tasks are typically categorised as follows: **Classification** assigns inputs to discrete categories (e.g. signal vs. noise [133, 134]) and the learning environment resembles a teacher-student dynamic. **Regression** tasks [135] are about predicting continuous outputs from inputs (e.g. estimating mass from waveform features). **Clustering** [136] involves grouping unlabelled data based on similarity (e.g. discovering sub-populations of events). **Dimensionality reduction** [137] helps with mapping data into a lower-dimensional space while retaining salient structure (e.g. reduced representation of some waveform). Finally, **generative modelling** [138] learns the data distribution to generate synthetic samples (e.g. producing surrogate waveforms).

Each task requires their own kind of loss function [139]. For regression tasks, common choices

include the mean squared error and mean absolute error, which differ in their sensitivity to outliers:

$$\textbf{Mean squared error (MSE):} \quad \mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{3.19}$$

penalises large deviations quadratically and is therefore sensitive to outliers. This property helps if we care about rare outliers.

$$\textbf{Mean absolute error (MAE):} \quad \mathcal{L}_{\text{MAE}} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|, \tag{3.20}$$

penalises deviations linearly and is less sensitive to outliers, which is advantageous if our need is to ignore outliers.

For classification tasks, cross-entropy loss is widely used, as it naturally arises from maximum likelihood estimation under a probabilistic model.

$$\textbf{Cross-entropy loss:} \quad \mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k}\log \hat{p}_{i,k}, \tag{3.21}$$

where $y_{i,k}$ is the one-hot encoded label (simply put: a vector of boolean values) and $\hat{p}_{i,k}$ is the predicted probability for class k.

Specialised loss functions, such as the focal loss can improve performance in settings where we want to compensate for any notion of imbalance that might be present between classes. This imbalance need not necessarily be due to dataset size, and can represent any abstract notion of imbalance between classes (e.g. difference in difficulty of learning given the learning architecture and training procedure).

$$\textbf{Focal loss (class imbalance):} \quad \mathcal{L}_{\text{focal}} = -\frac{1}{N}\sum_{i=1}^{N}(1 - \hat{p}_{i,y_i})^{\gamma}\log \hat{p}_{i,y_i}, \tag{3.22}$$

where $\gamma > 0$ controls the strength of focusing on hard-to-classify examples [140].

In many scientific applications, the loss function can incorporate additional domain-specific constraints or penalties to encourage physically plausible solutions. Examples include regularisation terms, such as the L_1 and L_2 norms:

$$\mathcal{R}_{L_1}(\theta) = \|\theta\|_1 = \sum_j |\theta_j|, \tag{3.23}$$

$$\mathcal{R}_{L_2}(\theta) = \|\theta\|_2^2 = \sum_j \theta_j^2, \tag{3.24}$$

which penalise large parameter values to reduce overfitting. More general penalties can enforce smooth-

ness, sparsity, or even physical conservation laws. In such cases, the total loss is often expressed as a weighted sum of a data fidelity term $\mathcal{L}_{\text{data}}$ and one or more regularisation terms:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda_1 \mathcal{R}_1(\theta) + \lambda_2 \mathcal{R}_2(\theta) + \cdots , \tag{3.25}$$

where the coefficients λ_i control the trade-off between accurately fitting the data and satisfying the imposed constraints.

The design and selection of the loss function must therefore account for both statistical properties and domain requirements. An appropriately chosen loss function not only guides the network towards accurate predictions but also ensures that the learned representations and outputs remain consistent with the problem's theoretical and practical constraints.

Loss Manifold: The loss manifold represents the dependence of the loss function on the parameters of a neural network. In an idealised setting, the loss surface resembles a simple convex bowl. In this case, the optimiser can smoothly descend towards a unique minimum, where the network parameters are optimal for the given task. However, as the cartoon in Fig. 3.8 shows, the landscape encountered may be far more irregular, with multiple valleys and peaks.

Local and Global Minima: The complex landscape depicted in Fig. 3.8 gives rise to the distinction between *local minima* and the *global minimum*. A local minimum is any point in parameter space where the loss is lower than that of all nearby configurations, but not necessarily the lowest possible overall. In contrast, the global minimum is the point with the lowest loss across the entire parameter space. In the figure, we see several valleys where optimisation may become trapped, corresponding to local minima, while only one valley corresponds to the absolute global minimum.

The challenge of deep learning optimisation is therefore not only to descend the manifold, but also to avoid poor local minima and saddle points. Loss optimisation algorithms typically exploit randomness and adaptive step sizes to help escape shallow traps and continue exploration. This makes it possible, in practice, to find minima that generalise well, even if they are not the global minimum in a strict mathematical sense.

Figure 3.8 shows a cartoon of the concept of loss manifold and the presence of local and global minima. The cat in this scenario represents the solver trying to reach the global minimum—its human companion. The horizontal axis corresponds to the parameter space, while the vertical axis indicates the loss value. The red arrows depict the updates performed by some loss minimiser as it attempts to follow the slope of the manifold towards regions of lower loss. The cat may encounter many situations where it may either get distracted or stuck. In these cases, the cat is unable to reach its companion and is said to have been stuck at a local minimum.

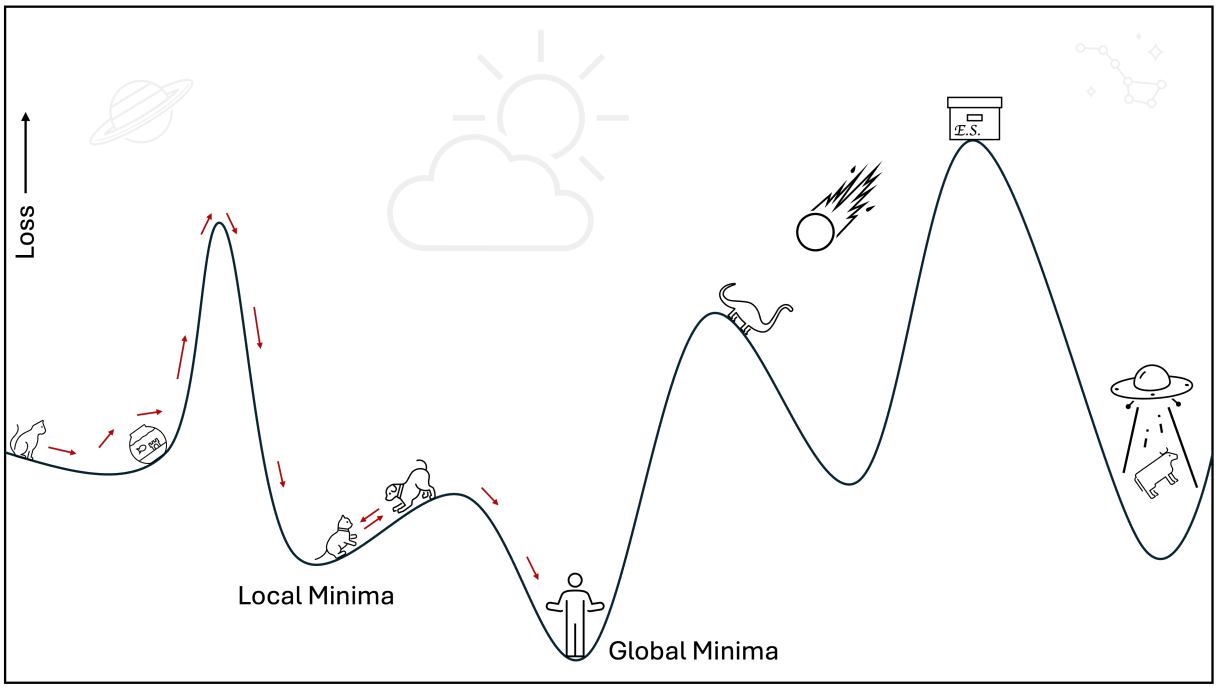


Figure 3.8: Cartoon depicting the loss landscape and the process of optimising the trainable parameters of the machine learning model to reach the global minima while avoiding local minima. The cat represents the machine learning model and the human represents the global minimum of the loss landscape. All other depictions can be seen as distractions that keep the cat from reaching the global minimum.

3.6 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a widely used optimisation algorithm for training neural networks and other machine learning models. Since this is quite a generic algorithm used for many optimisation problems outside ML as well, we will keep the mathematical details as generic as possible [129, 141].

Let us begin by considering some arbitrary real-valued cost function, $C(v)$, that we would like to minimise. Here, v can denote many variables like $v = (v_1, v_2, \ldots, v_n)$ but for the sake of simplicity, let us assume that C depends on two variables $v = (v_1, v_2)$. We assume a simple loss manifold made from a quadratic function, as represented in Fig. 3.9. To make the ball reach the global minimum at the centre of the bowl, we can simply track the derivatives of its movement and find places where $C(v)$ is an extremum.

With a small change Δv_1 and Δv_2 along the two variables considered $v = (v_1, v_2)$, the cost function should change as follows,

$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2 \tag{3.26}$$

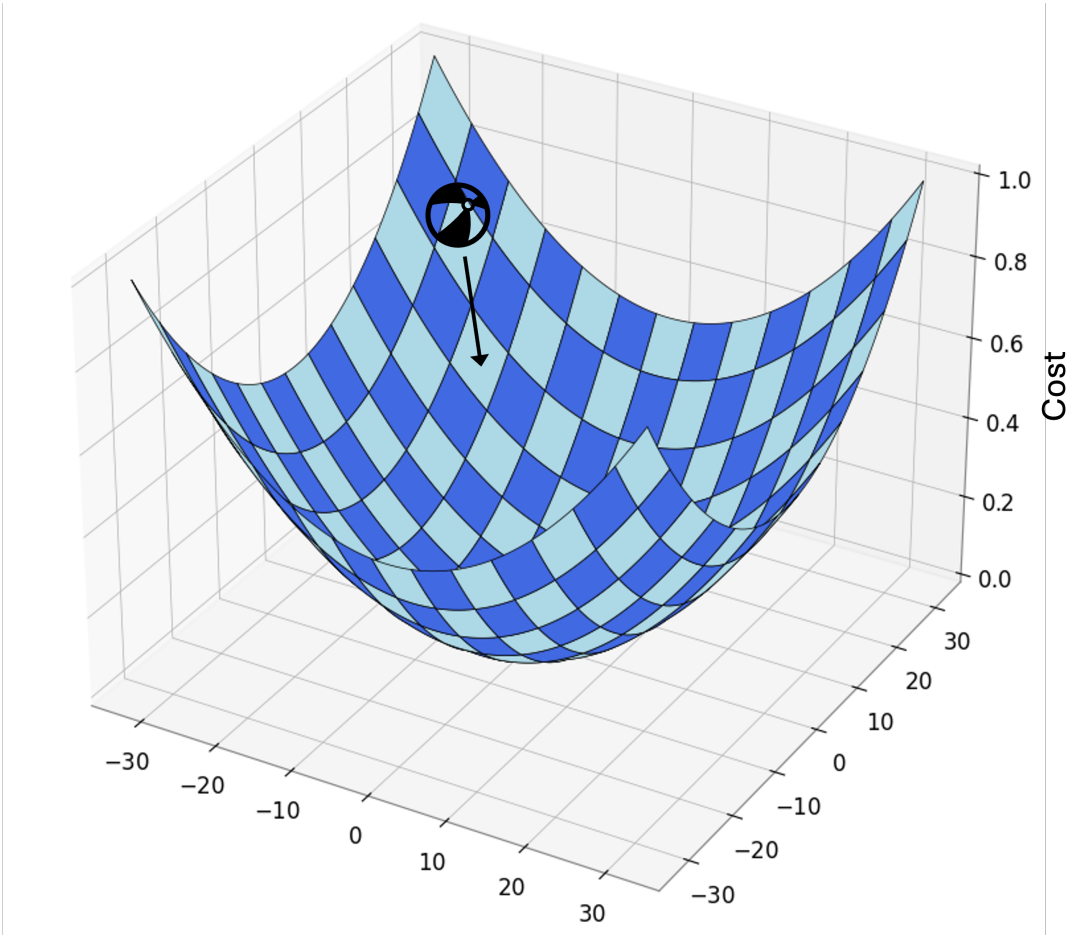


Figure 3.9: A three-dimensional parabolic loss manifold, representing the optimisation landscape of a simple quadratic function. The surface is coloured with a binary chequerboard pattern for visual clarity, while black borders outline the curvature of the bowl. The minimum lies at the centre, illustrating the point in parameter space where the loss is minimised. The ball is the solver trying to reach the global minimum.

where we can define the gradient vector of the cost function to be

$$\nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2} \right)^{\top} \tag{3.27}$$

where $\top$ is the transpose operator. From equations 3.26 and 3.27, we can write

$$\Delta C \approx \nabla C \cdot \Delta v. \tag{3.28}$$

Now, we can choose Δv such that ΔC is negative to move the ball in the direction of reducing cost, as follows

$$\Delta v = -\eta \nabla C \tag{3.29}$$

where η is small, positive real value called the *learning rate*. From equation 3.28, we can write

$$\Delta C \approx -\eta ||\nabla C||^2. \tag{3.30}$$

Since $||\nabla C||^2$ is always positive, ΔC is ensured to always be negative. This effectively says that the learning will proceed in the direction of the negative gradient of the cost function and will decrease the loss as training progresses (hereafter any use of the term *gradient* in the context of machine learning will refer to this definition unless stated otherwise). So, the simple update rule looks as follows:

$$v \rightarrow v' = v - \eta \nabla C \tag{3.31}$$

where v' is the updated value of v after a single step of the technique called *gradient descent*. From the cartoon shown in Fig. 3.8, it is apparent that if the step size for movement is too large, the cat might miss the global minimum and simply jump back and forth within a valley. Similarly, if the step size if too small, it will take too long to reach the global minimum or might lead to mistaking one of the local minima as the global minimum if there are strong positive gradients for the loss on either side.

Calculating ∇C requires the information gleaned from the entirety of the training data. However, in practice, this is typically not computationally feasible. Instead, the value of ∇C is well approximated by working with a small random subset (or mini-batch) of input training samples rather than the entire dataset. This introduces some stochasticity into the updates. Suppose we have a subset of m random samples from the n total samples of the training data $(X_1, X_2, \ldots X_n)$. Our assumption is that:

$$\frac{\sum_{i=0}^{m} \nabla C_i}{m} \approx \frac{\sum_{i=0}^{n} \nabla C_i}{n} = \nabla C \tag{3.32}$$

This assumption, of course, only holds when m is sufficiently large. Beyond efficiency, this stochasticity acts as an implicit regularisation mechanism: by introducing noise into the parameter updates, it prevents the optimiser from settling too quickly into sharp minima or saddle points of the loss landscape, thereby helping to mitigate overfitting and improving generalisation.

In the context of a neural network, given a parameter vector θ (consisting of all trainable parameters of a neural network) and a differentiable cost/loss function $C(\theta)$, the SGD update rule for a mini-batch $\mathcal{B}$ of size m is

$$\theta_{t+1} \rightarrow \theta_t - \eta \frac{1}{m} \sum_{i \in \mathcal{B}} \nabla_{\theta_t} C_i(\theta_t), \tag{3.33}$$

where η is the learning rate and $\nabla_\theta C_i$ denotes the gradient of the loss for the i-th sample. A modification to the update rule introduced for improving training stability is $momentum$. Momentum is an

addition to SGD that introduces an auxiliary velocity term v_t, yielding smoother updates:

$$v_{t+1} = \mu v_t - \eta \frac{1}{m} \sum_{i \in \mathcal{B}} \nabla_{\theta_t} C_i(\theta_t), \tag{3.34}$$

$$\theta_{t+1} = \theta_t + v_{t+1}, \tag{3.35}$$

with $\mu \in [0, 1)$ controlling the contribution of past gradients.

Aside from SGD, there are many adaptive methods of gradient descent that were introduced for the sake of faster convergence to the global minimum. Examples inlcude: RMSprop (an unpublished, adaptive learning rate method introduced by Geoffrey Hinton in Lecture 6e of his Coursera Class [1]), Adam [142], AdamW [143] etc). While adaptive methods often converge faster initially, SGD with momentum frequently yields superior generalisation in large-scale deep learning tasks [144, 145]. Consequently, SGD and its variants form the optimisation backbone for large-scale neural network training across a wide range of domains.

3.7 Back Propagation

Back propagation is the standard algorithm used to compute gradients of the loss function with respect to the parameters of a neural network. It applies the chain rule of calculus in a systematic manner, propagating error signals from the output layer back through the network to earlier layers. This enables efficient computation of all required partial derivatives for parameter updates using optimisation algorithms such as stochastic gradient descent [129, 146].

Consider a neural network composed of L layers. For each layer $l \in \{1, \ldots, L\}$, let the parameters be denoted by weights $W^{[l]}$ and biases $b^{[l]}$. Given an input x, the forward pass is defined recursively as

$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}, \tag{3.36}$$

$$a^{[l]} = \sigma^{[l]}(z^{[l]}), \tag{3.37}$$

where $a^{[0]} = x$, $z^{[l]}$ is the pre-activation vector, $a^{[l]}$ is the activation vector, and $\sigma^{[l]}(\cdot)$ is the activation function for layer l.

Let the cost/loss function be $C(a^{[L]}, y)$, where y denotes the target output. The aim of back-propagation is to compute the gradients

$$\frac{\partial C}{\partial W^{[l]}} \quad \text{and} \quad \frac{\partial C}{\partial b^{[l]}} \qquad \text{for all } l, \tag{3.38}$$

[1] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

efficiently using the chain rule.

Define the error term for layer l as

$$\delta^{[l]} = \frac{\partial C}{\partial z^{[l]}}. \tag{3.39}$$

For the output layer L, this is given by

$$\delta^{[L]} = \nabla_{a^{[L]}} C \odot \sigma'^{[L]}(z^{[L]}), \tag{3.40}$$

where $\odot$ denotes the Hadamard product and $\sigma'^{[L]}(\cdot)$ is the derivative of the activation function. For hidden layers, the recursion is

$$\delta^{[l]} = \left((W^{[l+1]})^\top \delta^{[l+1]}\right) \odot \sigma'^{[l]}(z^{[l]}). \tag{3.41}$$

The gradients with respect to the parameters are then

$$\frac{\partial C}{\partial W^{[l]}} = \delta^{[l]} (a^{[l-1]})^\top, \tag{3.42}$$

$$\frac{\partial C}{\partial b^{[l]}} = \delta^{[l]}. \tag{3.43}$$

This recursive computation allows efficient evaluation of gradients layer by layer, enabling the use of gradient-based optimisation algorithms such as stochastic gradient descent.

3.8 Training Strategies

The performance of machine learning models depends critically on the strategy used to train them. Different paradigms of training are suited to different forms of data and problem settings. Broadly, these strategies can be categorised into supervised learning, unsupervised learning, and reinforcement learning [129].

Supervised Learning: Supervised learning is a paradigm in machine learning where a model is trained on a dataset consisting of input–output pairs. Formally, given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ with inputs $x_i \in \mathcal{X}$ and corresponding labels $y_i \in \mathcal{Y}$, the objective is to learn a function $f_\theta : \mathcal{X} \to \mathcal{Y}$, parameterised by θ, that approximates the underlying mapping between inputs and outputs. Training is achieved by minimising a loss function $\mathcal{L}(f_\theta(x_i), y_i)$, which quantifies the discrepancy between the model prediction and the true label. Once trained, the model is expected to generalise, providing accurate predictions for unseen data.

Unsupervised Learning: Unsupervised learning is concerned with discovering patterns or structures in data without the use of explicit labels. Given a dataset $\mathcal{D} = \{x_i\}_{i=1}^N$ consisting only of inputs,

the objective is to learn representations that capture the underlying structure of the data distribution. Typical tasks include clustering, where the goal is to group similar samples together, and dimensionality reduction, where high-dimensional data are mapped to a lower-dimensional space while preserving salient features. Unlike supervised learning, the absence of labels means the model must rely solely on the statistical regularities of the inputs, making unsupervised methods valuable for exploratory analysis and as a foundation for representation learning.

Reinforcement Learning: Reinforcement learning is a training paradigm that addresses sequential decision-making problems. In this setting, an agent interacts with an environment over discrete time steps, receiving observations, taking actions, and collecting rewards. Formally, the interaction is modelled as a Markov decision process, defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, P the state transition probabilities, R the reward function, and $\gamma \in [0, 1]$ the discount factor. The objective of the agent is to learn a policy $\pi(a|s)$ that maximises the expected cumulative discounted reward

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]. \tag{3.44}$$

Unlike supervised learning, where training signals are provided in the form of labelled data, reinforcement learning relies on delayed and potentially sparse reward signals. This makes the exploration–exploitation trade-off central to the paradigm, as the agent must balance gathering information about the environment with exploiting current knowledge to maximise reward.

Beyond the classical categories of supervised, unsupervised, and reinforcement learning, several alternative training strategies are commonly employed in modern machine learning.

Self-supervised learning: constructs surrogate training objectives directly from the structure of the data, eliminating the need for manual labels. The central idea is to define pretext tasks in which part of the input is withheld, corrupted, or transformed, and the model is trained to predict the missing or altered information. Common examples include masked prediction, temporal ordering, and reconstruction tasks. By solving these surrogate objectives, the model learns representations that capture meaningful statistical regularities in the data. These representations can then be transferred to downstream tasks with limited labelled data, often yielding strong performance [147].

Semi-supervised learning: combines a small labelled dataset $\mathcal{D}_L$ with a larger unlabelled dataset $\mathcal{D}_U$. The objective typically balances a supervised loss with an unsupervised regularisation term,

$$\min_\theta \; \mathcal{L}_{\text{sup}}(\mathcal{D}_L) + \lambda \mathcal{L}_{\text{unsup}}(\mathcal{D}_U). \tag{3.45}$$

Common approaches include consistency regularisation, where the model is encouraged to produce stable predictions under input perturbations, and pseudo-labelling, where confident predictions on

unlabelled data are treated as temporary labels. By leveraging the underlying structure of the unlabelled data, semi-supervised learning can reduce overfitting to limited labels and achieve better generalisation than purely supervised methods.

These strategies differ primarily in how learning signals are constructed and scheduled, rather than in the underlying optimisation algorithms.

Intuition 3: Revisiting Activation Functions

Now that we have a slightly better understanding of how the neural network trains, let us revisit how and why an activation function is constructed. When designing an activation function $f(\cdot)$ for use in neural networks, several desirable properties are typically considered:

1. **Non-linearity**: The function must be non-linear, so that the composition of multiple layers of neurons can approximate complex functions beyond linear mappings.
2. **Monotonicity**: A monotonic activation function ensures stable loss gradient directions during optimisation, avoiding oscillations in parameter updates.
3. **Differentiability**: The function should be differentiable almost everywhere to allow loss gradient-based training (with backpropagation). Piecewise differentiable functions such as ReLU are commonly accepted.
4. **Boundedness**: Bounded activation functions (e.g., sigmoid, tanh) constrain outputs to a finite range, which can improve stability but may suffer from saturation. Unbounded functions (e.g., ReLU) are common, though they may risk exploding activations.
5. **Smoothness**: Smooth functions yield stable gradients and better convergence properties. Sharp discontinuities can cause optimisation difficulties.
6. **Non-zero output range**: Ideally, the function should avoid mapping all negative or positive inputs to a single constant value, as this reduces expressivity and leads to "dead neurons".
7. **Computational efficiency**: Since activations are applied to every neuron across possibly millions/billions of operations in modern deep neural networks, the function should be fast to compute and memory efficient.
8. **Gradient behaviour**: The derivative of the function should avoid vanishing ($\to 0$) or exploding ($\to \infty$) over large input ranges. This ensures stable learning dynamics in deep networks.

However, some of these rules can be broken, and they often are, for the sake of simplicity, computational efficiency or to avoid a particular learning issue that may arise from the training dataset and network architecture.

3.9 Function Space of Neural Networks

Before considering learning dynamics or optimisation, it is instructive to examine what a neural network represents prior to training. Even with randomly initialised weights and no data-dependent adjustment (no training, calculation of errors or update of trainable parameters), a neural network defines a structured class of functions determined entirely by its architecture, activation functions, and parameter distributions. Thus, sampling a network with fixed random weights can be viewed as sampling a function from the set of functions representable by the network.

This perspective reveals that many qualitative properties often attributed to learning—such as smoothness or boundedness—are already present at initialisation. In particular, shallow networks generate smooth, low-complexity functions on compact domains, while exhibiting clear limitations in their ability to represent sharp transitions, high-frequency oscillations, or multi-scale structure.

In this section, we sample functions from shallow fully connected networks with different configurations to examine the types of functions they can and cannot generate. This provides the necessary context for understanding representational capacity and how it varies with architectural design choices, as explored in the next section.

3.9.1 What can a single neuron represent?

Let us consider a network with a single hidden layer containing one neuron and a single linear output neuron. The hidden-layer activation function is the hyperbolic tangent, while no activation is applied at the output. Concretely,

$$h(x) = \tanh(W\,x + b), \tag{3.46}$$

where W is the weight and b is the bias.

Varying the weight: The weight scales the input, controlling the slope or "steepness" of the transition around the activation threshold. Specifically: [i] If $|W|$ is small, the function is approximately linear over $x \in [-2, 2]$, since $\tanh(z) \approx z$ for small z, [ii] If $|W|$ is large, the function saturates quickly to ± 1 outside a narrow region around $x = -b/W$, producing a sharp transition.

Varying the bias: The bias shifts the activation horizontally along the x-axis. Increasing b moves the transition point to the left, while decreasing b moves it to the right. For example:

$$h(0) = \tanh(b), \quad \text{so } b \text{ controls the output at } x = 0. \tag{3.47}$$

Given that we have a linear output neuron as well, the final function that the network represents is:

$$\mathrm{h}_{\mathrm{out}}(x) = W_{\mathrm{out}}\,\tanh(W\,x + b) + b_{\mathrm{out}}, \tag{3.48}$$

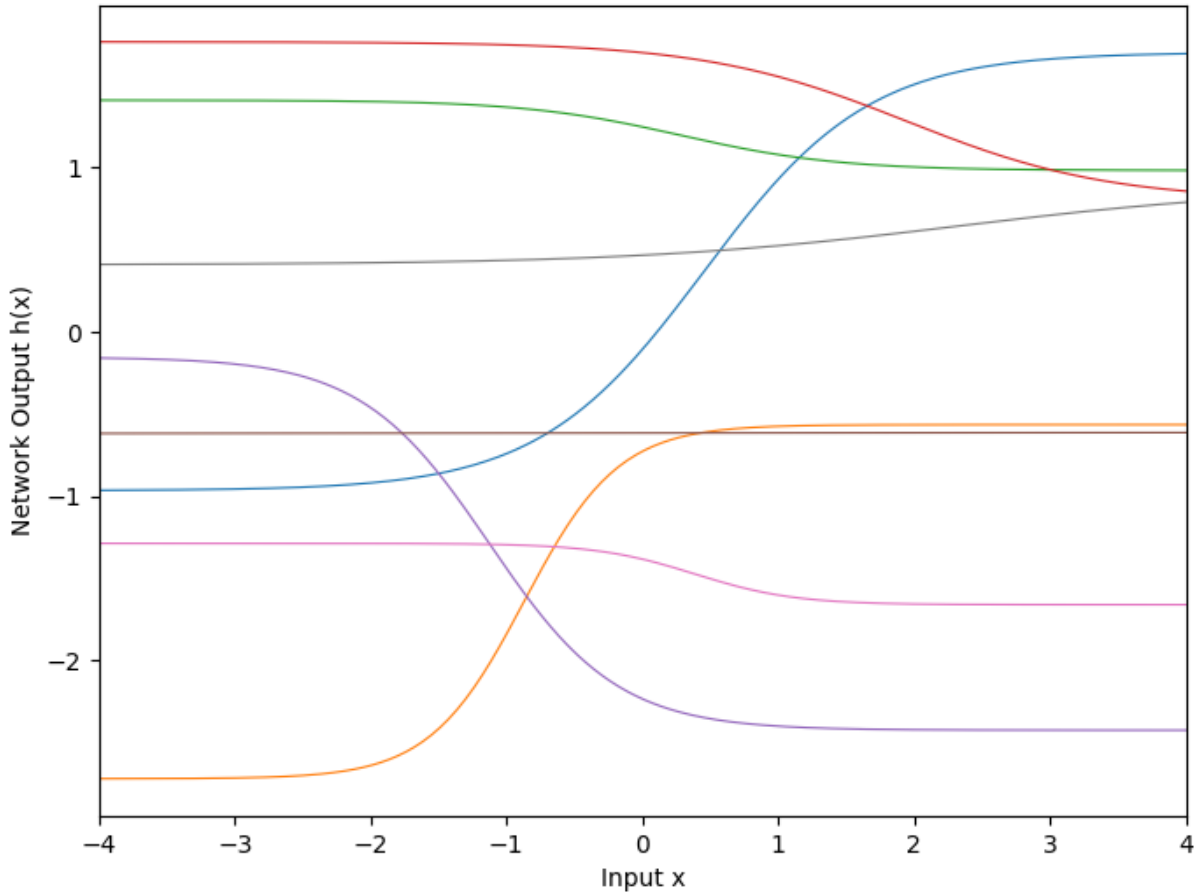


Figure 3.10: Subset of the function space of 1 tanh neuron

where W and b are the weight and bias of the hidden neuron, and W_{out} and b_{out} are the weight and bias of the output neuron.

For this experiment, network weights and biases are independently sampled from a uniform distribution over a bounded latent space $[-2, 2]$. While the latent space is typically allowed to span $[-\infty, +\infty]$, we constrain it here for practicality while maintaining sufficient variability. Each sampled network defines a realisation of $f(x) = \tanh(W x + b)$, evaluated on a fixed one-dimensional input grid of 256 evenly spaced points over the interval $[-2, 2]$. In total, one 8 independent network initialisations are sampled, producing a large empirical distribution over the functions representable by this architecture under random initialisation. The resulting functions are illustrated in Fig. 3.10. The output functions are not bound to ± 1 given that we use a linear output neuron, unconstrained by an activation function.

3.9.2 What can n neurons represent?

For a shallow network with a single hidden layer containing n neurons, the hidden layer outputs are

$$h_j(x) = \tanh(W_j x + b_j), \quad j = 1, \ldots, n, \tag{3.49}$$

where W_j and b_j are the weight and bias of the j-th hidden neuron. The output neuron linearly combines these hidden outputs:

$$\mathrm{h}_{\text{out}}(x) = \sum_{j=1}^{n} W_{\text{out},j} \tanh(W_j x + b_j) + b_{\text{out}}, \tag{3.50}$$

where $W_{\text{out},j}$ is the weight connecting the j-th hidden neuron to the output, and b_{out} is the output bias. Compared to the single-neuron case, the main difference is that multiple hidden neurons allow the network to form a weighted sum of multiple smooth, bounded non-linear functions, enabling more complex function shapes over the input domain. For simplicity, let us look at the two neuron case:

$$\mathrm{h}_{\text{out}}(x) = W_{\text{out},1} \tanh(W_1 x + b_1) + W_{\text{out},2} \tanh(W_2 x + b_2) + b_{\text{out}}. \tag{3.51}$$

This function is deceptively simple but should be capable of approximating a simple function like $f(x) = \sin(x)$ relatively well. Before we sample from this network, let us rationalise why this is even possible. For a single hidden neuron,

$$h(x) = \tanh(wx + b), \tag{3.52}$$

we have

$$h'(x) = w \operatorname{sech}^2(wx + b), \tag{3.53}$$

$$h''(x) = -2w^2 \tanh(wx + b) \operatorname{sech}^2(wx + b). \tag{3.54}$$

From these derivatives, we observe that $h'(x)$ is always positive (or negative) depending on the sign of w, and has a single maximum at $x = -b/w$. Furthermore, $h''(x)$ changes sign exactly once at $x = -b/w$, indicating a single inflection point. Thus, each hidden neuron produces a *single transition region*. For a linear output neuron, with output defined by equation 3.51, we can write the derivative as,

$$g'(x) = W_{\text{out},1}^2 \operatorname{sech}^2(W_1 x + b_1) + W_{\text{out},2}^2 \operatorname{sech}^2(W_2 x + b_2). \tag{3.55}$$

Each term is unimodal and positive (or negative), therefore the sum $g'(x)$ can exhibit at most *two local maxima*. Consequently, $g(x)$ can have at most two regions of rapid change or "transitions". This is why we can rationalise that two neurons can represent a single sinusoidal cycle relatively well.

We repeat the sampling procedure from the previous experiment, replacing the single neuron model

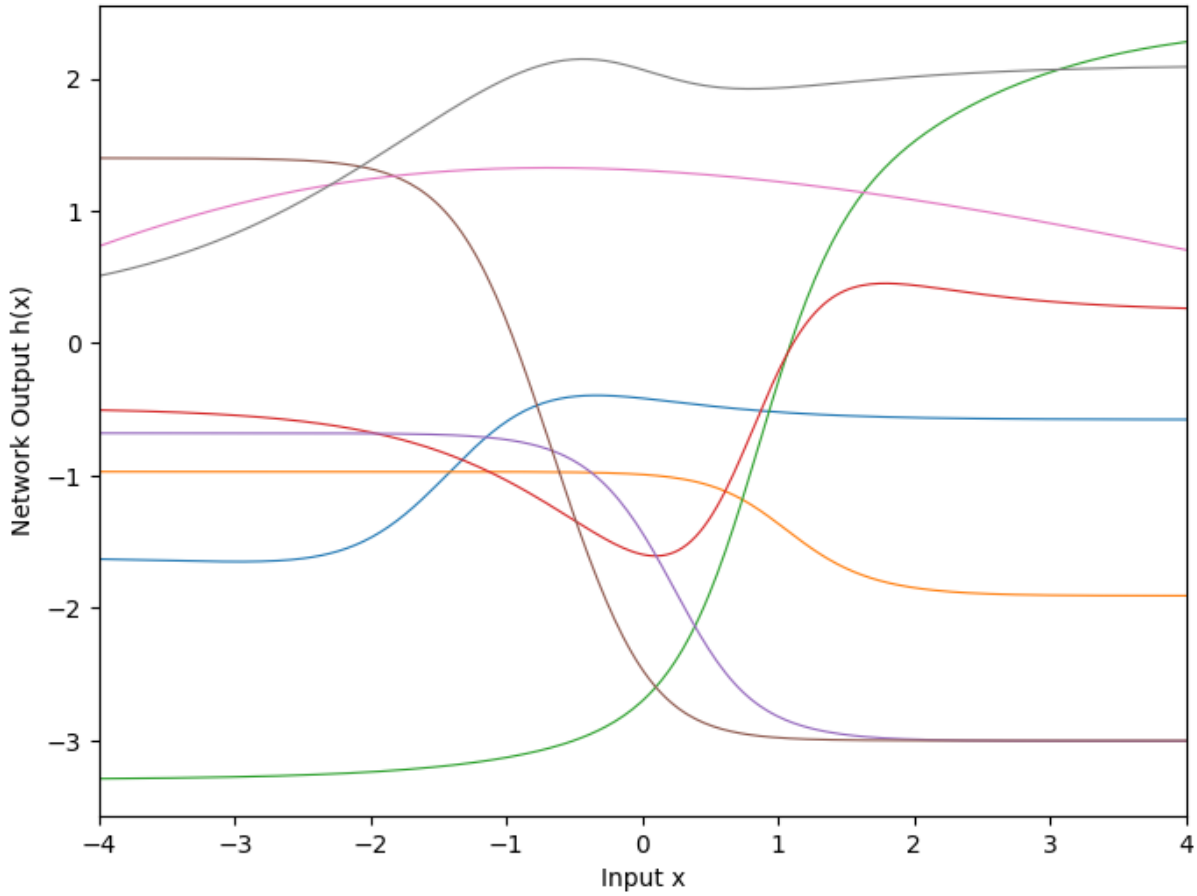


Figure 3.11: Subset of the function space of a 1-layer 2 tanh neuron network

in Eqn. 3.48 with the two-neuron architecture given in Eqn. 3.51. Figure 3.11 shows eight randomly sampled functions from this model. While some increase in variability is immediately apparent, the qualitative effect of adding a second neuron is not fully captured by a small number of samples.

To better expose the change in representational structure, we consider an arbitrary target function $f(x) = \sin(3x) + 0.3x$. We then sample one million random functions from the two-neuron network and examine whether any of these realisations approximate the target. This experiment highlights a qualitative shift in the function space: whereas the single-neuron network is fundamentally limited to monotonic, single-transition shapes, the two-neuron network can generate functions with multiple regions of curvature and non-trivial oscillatory structure. The appearance of functions closely resembling the target serves as evidence that even a minimal increase in width can substantially expand the class of functions the network is capable of representing. In Fig. 3.12, we plot all functions, among the one million sampled realisations, whose ℓ_2 distance from the target function falls below a specified threshold. We can clearly see functions that closely resemble a sinusoid and a good enough fit to our target function even through a brute force search.

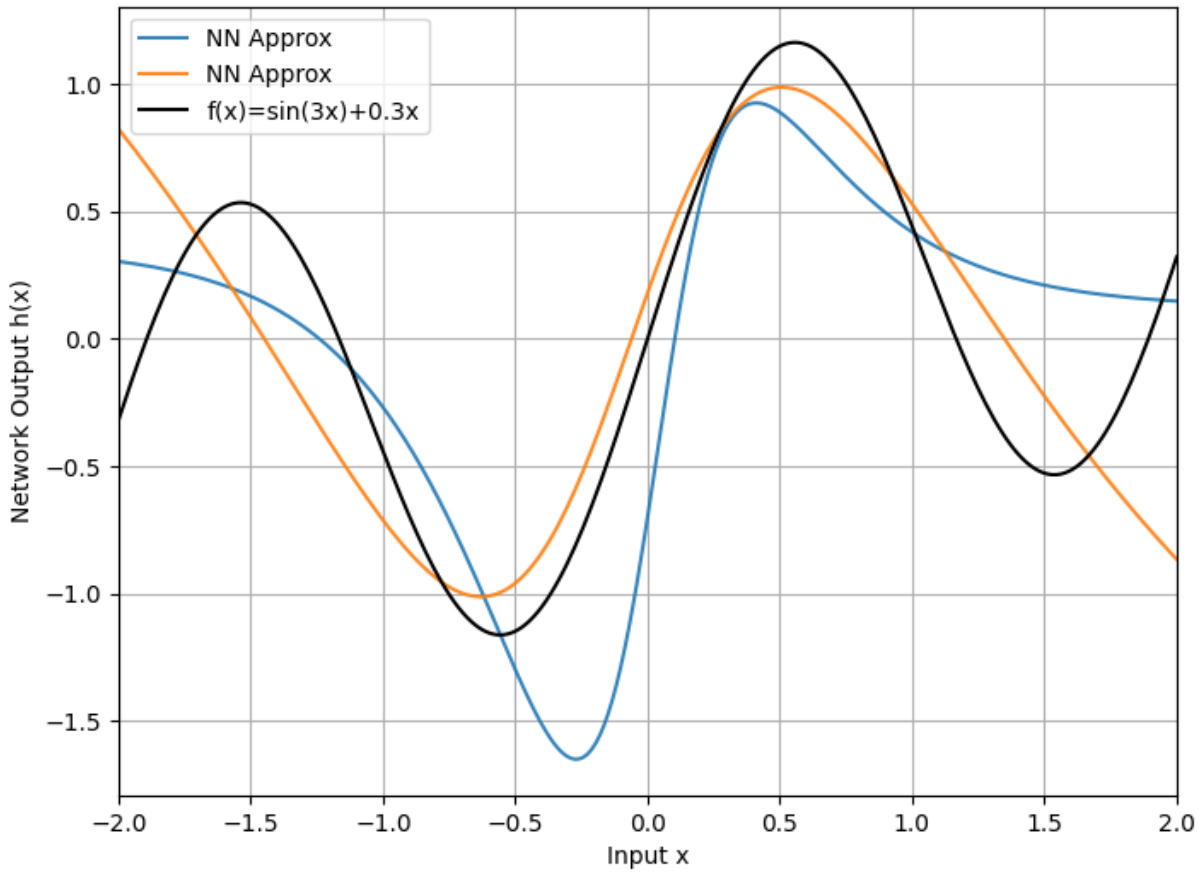


Figure 3.12: Subset of the function space of a 1-layer 2 tanh neuron network that are close in ℓ_2 distance to the target $f(x) = \sin(3x) + 0.3x$ within the range $[-2, +2]$.

3.9.3 What can multiple layers do?

For a network with fixed width n and L hidden layers, the represented function takes the compositional form

$$f(x) = W^{(L+1)}\,\phi\Big(W^{(L)}\,\phi\Big(\cdots\phi\Big(W^{(1)}x + b^{(1)}\Big)\cdots\Big) + b^{(L)}\Big) + b^{(L+1)}, \tag{3.56}$$

where $\phi(\cdot)$ is a non-linear activation function. Increasing depth increases the number of nested non-linear compositions without increasing width. As a result, the number of distinct linear regions (or curvature changes) in $f(x)$ grows exponentially with depth, but only linearly with width. This enables deep networks to represent functions with hierarchical and multiscale structure—such as amplitude-modulated oscillations—that shallow networks of comparable size cannot represent efficiently. We demonstrate this by sampling a set of functions from a 2 hidden layer network with 2 neurons in each layer, followed by a linear output. Below a certain threshold, we are left with 2 functions from the 1 million sampled functions, as shown in Fig. 3.13.

These approximations are very bad fits, but why? This is because the number of possible functions combinatorially explodes with an increase in the number of layers and finding a good enough set of weights and biases to represent our target function is simply much less probable. To obtain a crude

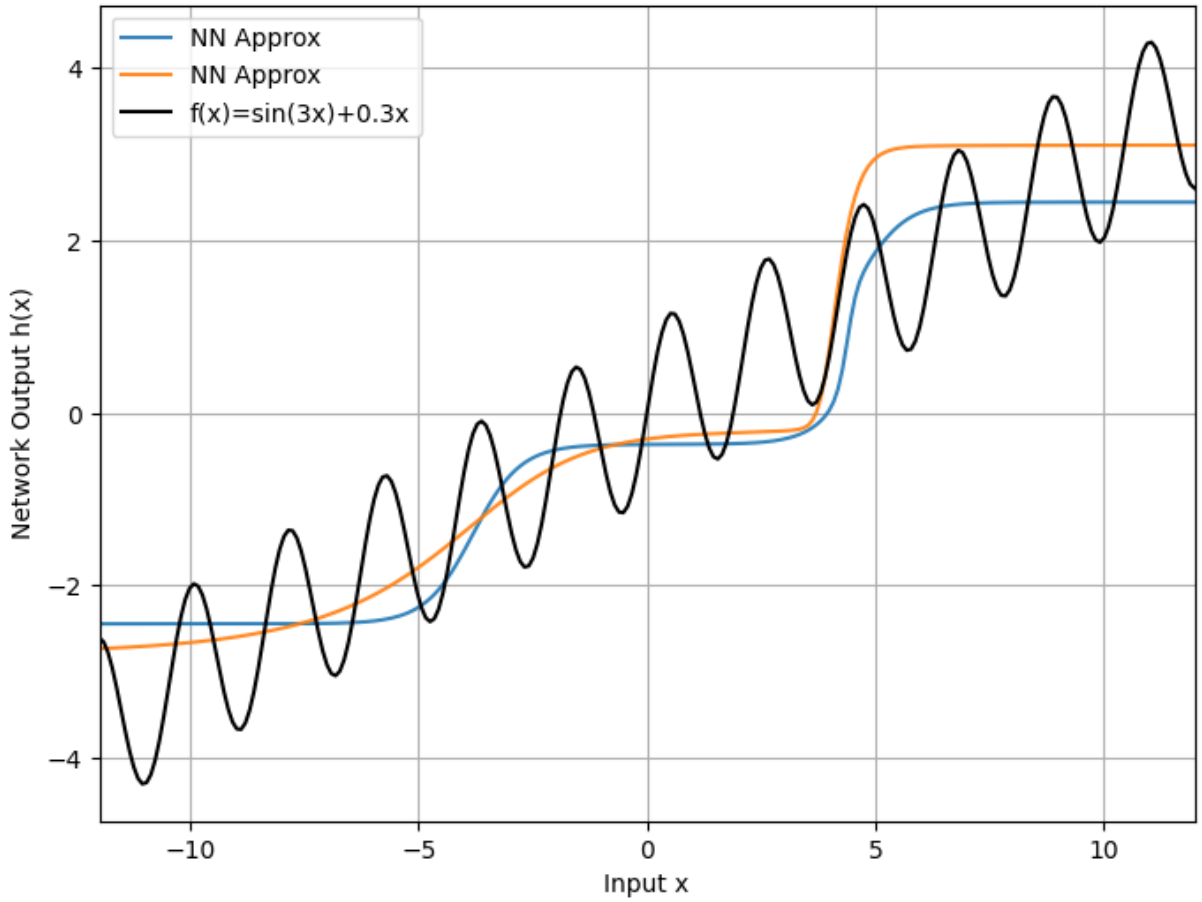


Figure 3.13: Subset of the function space of a 2-layer 2 tanh neuron network that are close in ℓ_2 distance to the target $f(x) = \sin(3x) + 0.3x$ within the range $[-12, +12]$.

estimate of the size of the induced function space, suppose each of the P network parameters is restricted to a bounded interval and discretised into only K possible values. Even under this extremely coarse assumption, the total number of distinct parameter configurations is K^P. For a network with $P = 4 \times 10^7$ parameters (which is a modest network size by today's standards) and $K = 2$, this yields $2^{4\times10^7} \approx 10^{1.2\times10^7}$ distinct configurations, indicating an astronomically large space of representable functions even under severe discretisation. This example allows us to appreciate the power of stochastic gradient descent and backpropagation in finding a near perfect set of weights and biases in a reasonable time given the vastness of the function space, especially in much deeper and larger networks.

I encourage the reader to experiment with the supplementary code provided for this section to get a better intuition of the capabilities and limitations of a neural network. Please feel free to raise any issues that may arise on GitHub and I will try my best to address them.

3.10 Representational Capacity

The representational capacity of a model describes the breadth of functions it can approximate given sufficient parameters and appropriate optimisation. In the context of simple neural networks, this capacity is determined by the total number of neurons, the width/depth of the network, and the choice of activation functions, as seen in the previous section. A network with insufficient capacity may fail to capture essential structures in the data, whereas one with excessive capacity may memorise training examples without learning generalisable patterns. Understanding and managing representational capacity is critical for achieving reliable performance.

We will use the rest of this section to perturb the neural network architecture and its inner working to the point of failure and discuss how we can mitigate some of these issues. In particular, we will gravitate toward examples that will later be useful in the construction of a neural network for the GW detection problem.

3.10.1 Effect of Layer Width

Here, we illustrate the approximation capabilities of a shallow neural network with a single hidden layer for a one-dimensional regression task. The target function that we want the network to learn is defined as

$$f(x) = \sin(3x) + 0.3x. \tag{3.57}$$

This is used to generate input-output pairs for training. The network comprises one input neuron, a hidden layer of variable width with `tanh` activation, and a single linear output neuron. Training is performed using the mean squared error loss and the Adam optimiser. Figure 3.14 compares the actual versus learnt function for a one layer neural network with different width configurations.

The results illustrate the effect of hidden-layer width on the representational capacity of the network. As the number of hidden neurons increases, the network is able to approximate the target function more closely, capturing finer details of its oscillatory structure. This demonstrates how network width governs expressive power, with wider architectures capable of representing increasingly complex functions. These observations are consistent with the findings of [148], which show that increasing layer width can provably improve generalization under appropriate conditions.

3.10.2 Effect of Network Depth

Network depth is often tied to the better generalisation, effective learning of long range dependencies in data and lower losses. In this example, we will try and empirically see whether this is true or not. Here, we will vary the depth of a neural network to affect its ability to approximate a one-dimensional regression function. The target function is again given by Eqn. 3.57 and input-output pairs are generated over a wider interval $x \in [-12, 12]$ compared to the previous example. The network architecture

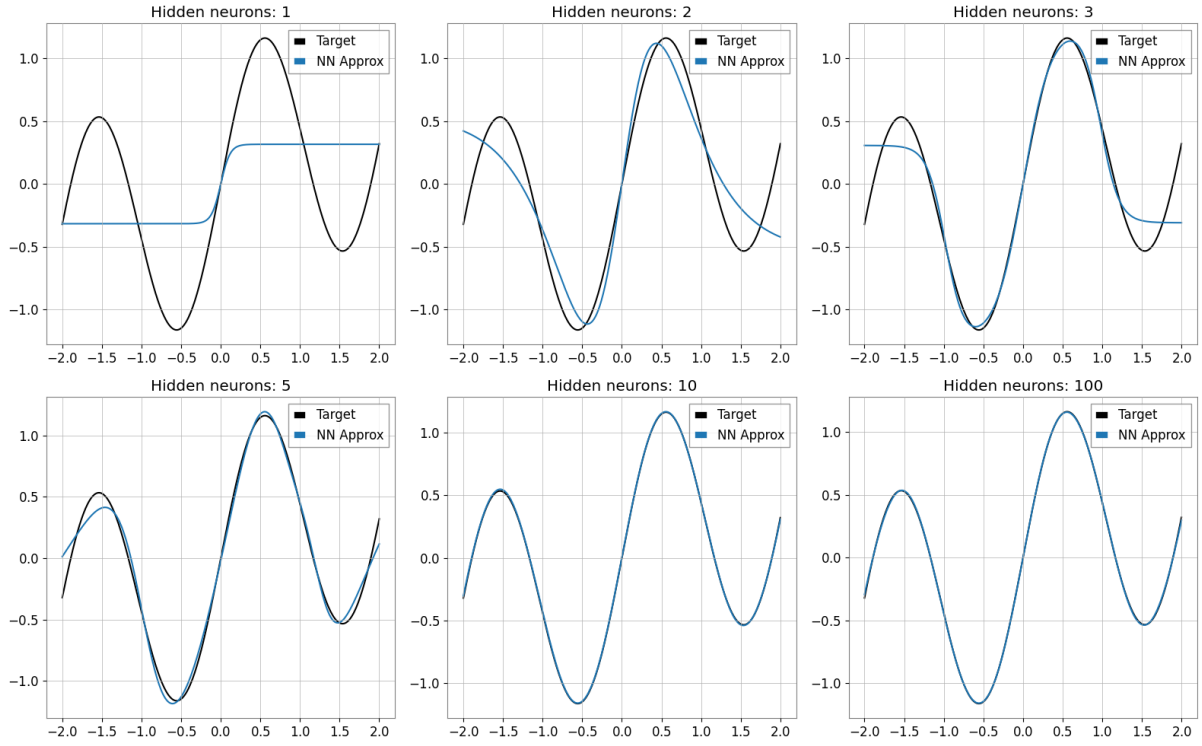


Figure 3.14: Comparison of target function (black) and neural network approximations for varying numbers of hidden neurons. Each subplot shows the effect of hidden layer width on the ability of a single-hidden-layer network to fit a one-dimensional function.

consists of a variable number of hidden layers (depth) with a fixed number of neurons per layer (20 neurons) and `tanh` activation functions.

Training is performed using the mean squared error loss and the Adam optimiser. In Fig. 3.15, each subplot illustrates the approximation achieved by a network with a specific number of hidden layers. As depth increases, the network can capture more complex patterns and global structure of the target function, highlighting the role of network depth in enhancing representational capacity [149, 150].

3.10.3 Effects of Different Activation Functions

This example illustrates the influence of different activation functions on the ability of a small neural network to approximate a highly oscillatory one-dimensional function. The target function is defined as

$$f(x) = x\ \sin(10x), \tag{3.58}$$

and is sampled uniformly distributed over $x \in [-1, 1]$.

We construct two neural networks with the same architecture. The network consists of three fully connected layers with a fixed hidden size of 10 neurons. The first two layers are followed by a non-linear activation function, which are set to `ReLU` in one network and `SiLU` in the other. The final

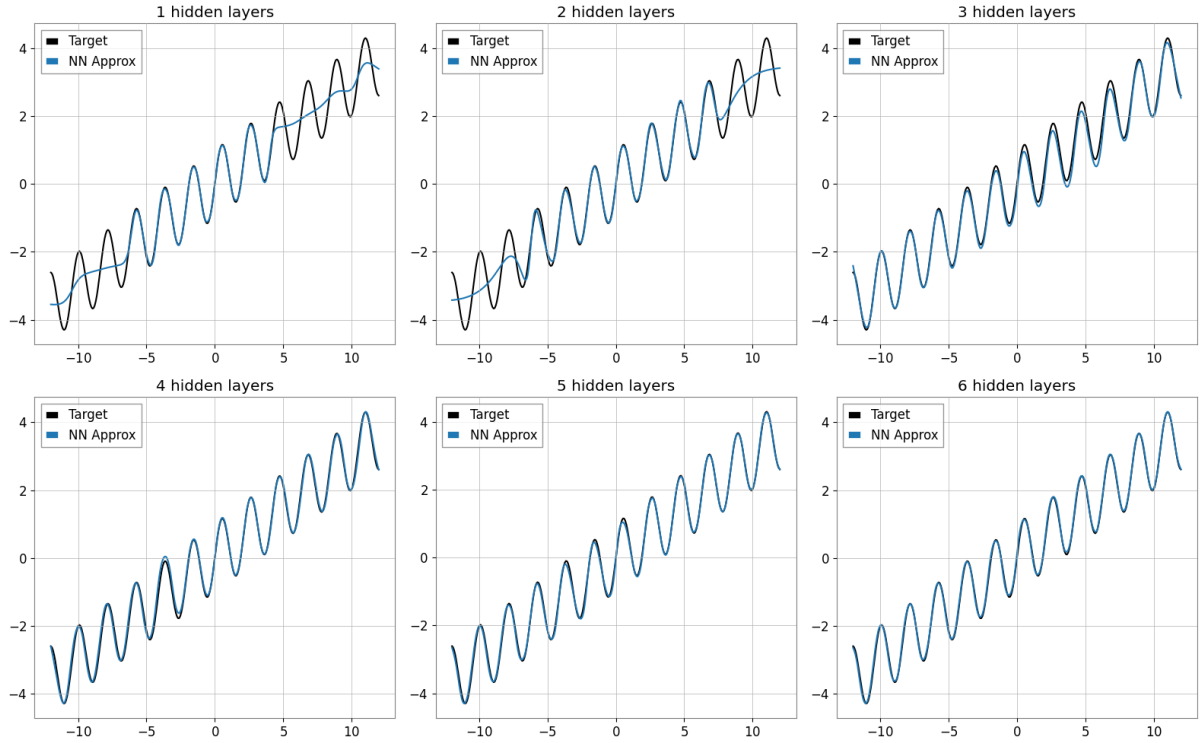


Figure 3.15: Neural network approximations (colored) of a one-dimensional target function (black) for varying numbers of hidden layers. Each subplot shows how increasing network depth affects the network's ability to model complex functional behaviour.

layer outputs a scalar (y for a given x), producing the network's approximation of the target function.

Training is performed using the mean squared error loss and the Adam optimiser. The network is trained for 3000 iterations to minimise the discrepancy between the predicted and true values. Figure 3.16 shows the reconstruction capability of each of these networks. The difference in approximation quality between the ReLU and SiLU activation functions arises from their distinct mathematical properties, particularly in how they transform input signals and propagate gradients through the network.

The ReLU function is defined as

$$\mathrm{ReLU}(z) = \max(0, z), \tag{3.59}$$

which is piecewise linear and strictly zero for all negative inputs. This introduces sparsity in the activations: neurons that receive negative pre-activations do not contribute to the output or to the gradient update. While ReLU networks are effective at representing piecewise linear functions, their ability to model smooth oscillatory functions is limited, especially when the input frequently crosses zero, as in $f(x) = x\sin(10x)$. The network must rely on linear combinations of multiple ReLU units to approximate each oscillation, leading to sharp transitions and a piecewise linear approximation. Moreover, the zero-gradient regions for negative inputs can slow learning in parts of the input domain, potentially

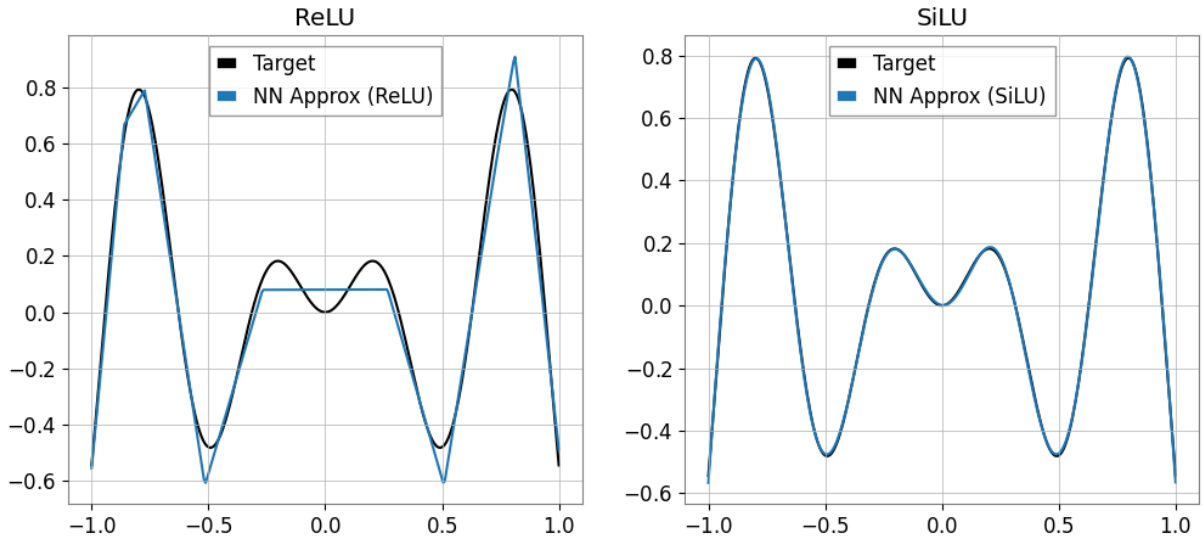


Figure 3.16: Neural network approximations of the target function $f(x) = x\sin(10x)$ using two different activation functions: ReLU (left) and SiLU (right). The black curve represents the target function, and the coloured curves represent the network outputs. Each subplot demonstrates the effect of activation choice on approximation quality.

causing underfitting in those regions.

In contrast, the SiLU (Sigmoid Linear Unit, also known as the swish function) is defined as

$$\text{SiLU}(z) = z \cdot \sigma(z) = \frac{z}{1 + e^{-z}}, \tag{3.60}$$

where $\sigma(z)$ is the sigmoid function. SiLU is smooth, differentiable everywhere, and non-monotonic for $z < 0$, which allows small negative pre-activations to contribute to the output with non-zero gradients. This smoothness enables the network to better capture oscillatory patterns, as each neuron can represent both positive and negative slopes without the abrupt zeroing behaviour of ReLU. The continuous derivative of the SiLU function without abrupt changes also improves gradient flow during backpropagation (gradient flow is the movement of error signals backward through a network during training, determining how effectively earlier layers learn), allowing more stable and effective learning of high-frequency components in the target function.

In this example, the target function $f(x) = x\sin(10x)$ has rapid oscillations and changes sign frequently across the domain. The ReLU network struggles to approximate the smooth sinusoidal peaks and troughs, producing a piecewise linear approximation with noticeable kinks. The SiLU network, however, leverages its smooth, non-zero negative activations to generate a continuous and differentiable approximation, resulting in a closer fit to both the amplitude and phase of the oscillations. Consequently, the choice of activation function significantly affects the representational capacity of small networks when modelling highly oscillatory functions [151].

3.10.4 Parameter Count vs Effective Capacity

So far, more network parameters tends to provide better learning. Is that always true?

We investigate how the architecture of a neural network—specifically, the combination of width of each hidden layer and the depth of the network (a shallow-wide and a narrow-deep network with different parameter counts)—affects its ability to approximate a target function. The target function chosen is again 3.57 which exhibits both oscillatory and linear components, providing a moderate level of complexity for function approximation. Two classes of network architectures are examined. First, *shallow-wide networks* with a single hidden layer of varying width (number of neurons) ranging from 1 to 128. Increasing width allows a single-layer network to represent more linear combinations of activation functions, thereby improving approximation capacity. Second, *narrow-deep networks* with a fixed small width (8 neurons per hidden layer) and varying depth from 1 to 6 layers. Adding layers increases the compositional depth, allowing the network to capture more complex interactions and hierarchical structures in the input-output mapping.

For each network configuration, we train using the Adam optimiser with a mean squared error loss for a fixed number of epochs (where epochs are typically one complete pass of the entire training dataset through a model during the learning process). After training, the final average loss is recorded, along with the total number of learnable parameters in the network. The two classes of networks are compared by plotting the MSE loss versus the number of parameters. This allows a direct assessment of how architectural choices impact representational efficiency: whether adding neurons to a shallow network or adding layers to a narrow network provides better approximation for a given parameter budget.

The expected behaviour is that increasing either width or depth generally reduces training loss. Shallow-wide networks excel at representing functions that can be decomposed into many linear combinations of simple activation functions, whereas narrow-deep networks leverage hierarchical compositions to capture complex patterns more efficiently per parameter. In this case, as we can expect given the target function, the narrow-deep network learns better, as seen in Fig. 3.17.

3.10.5 Memorisation and Overfitting

This example illustrates how a neural network with more parameters than are needed for the problem, otherwise known as an over-parameterised network, can overfit when trained on an extremely limited dataset. The target function is a simple sinusoid:

$$f(x) = \sin(2x), \tag{3.61}$$

evaluated over the interval $[-2, 2]$. To induce the network to memorise the input data instead of generalise over the problem, also known as *overfitting*, only a very small subset of 5 points is used for

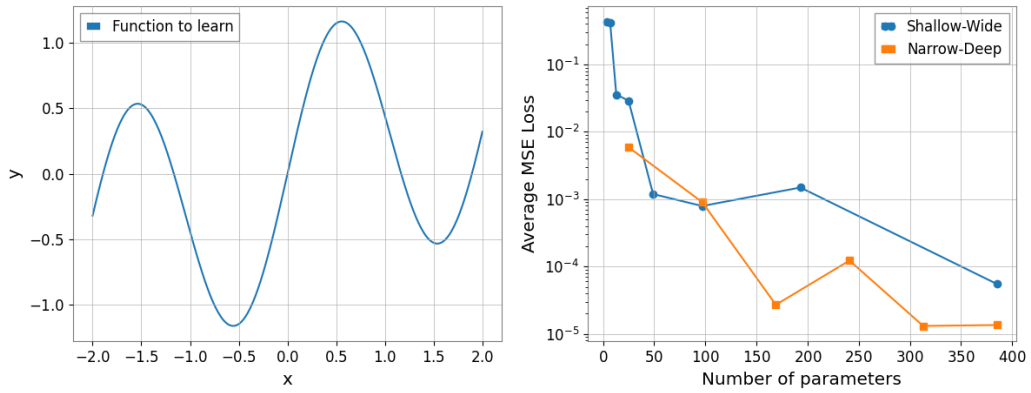


Figure 3.17: Left: target function $f(x) = \sin(3x) + 0.3x$. Right: comparison of mean squared error loss as a function of total number of network parameters for shallow-wide (single layer, varying width) and narrow-deep (fixed width, varying depth) networks.

training, while the remaining points serve as a validation set. The network architecture is deliberately wide, consisting of two hidden layers with 64 neurons each and `tanh` activations, which provides sufficient capacity to exactly interpolate the training points.

During training, the network minimises the mean squared error loss on the training set using the Adam optimiser. The evolution of both training and validation loss is recorded. The intended behaviour is that the training loss decreases rapidly as the network fits the few available points, while the validation loss initially decreases but soon increases, signalling overfitting (as seen in Fig. 3.18). This demonstrates the sensitivity of large-capacity models to the size of the training dataset and highlights the importance of regularisation and sufficient data coverage.

To visualise the overfitting behaviour better, we can plot the learnt function against the intended target function alongside the training samples that the network is meant to have overfit on. The wide neural network trained on only 5 points is evaluated across the full input domain $x \in [-2, 2]$. Figure 3.19 shows that despite interpolating the training points exactly, the network exhibits large deviations from the true function between these points, illustrating overfitting and poor generalisation.

Several interesting works delve deeper into this topic: on what the network chooses to memorise and why [152], why does a neural network learn anything at all even though they are capable of memorising [153] and the trade-off between generalisation and memorisation in neural networks [154].

3.10.6 Exploring the vanishing and exploding gradients problem

Learning is not just about how good the network architecture is or how much data we have at hand. It depends significantly on how the gradients vary during training. This is heavily influenced by the weight initialisation in a neural network, which is the process of setting the starting values of the model's parameters before training. This strongly governs convergence speed and stability.

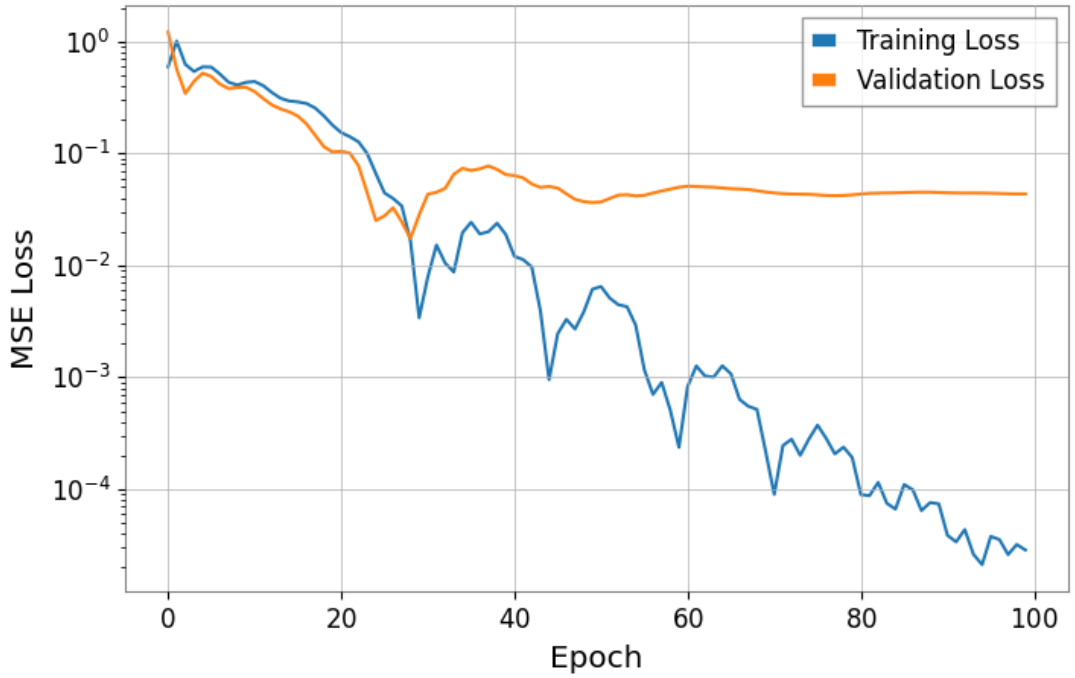


Figure 3.18: Training and validation mean squared error loss versus epoch for a wide neural network trained on only 5 points sampled from $f(x) = \sin(2x)$.

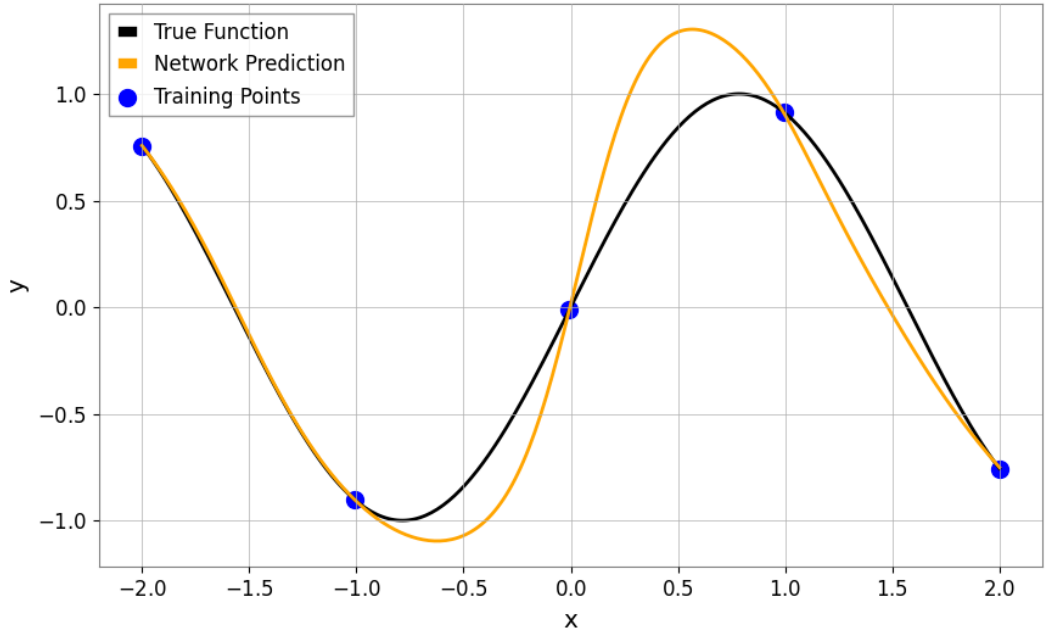


Figure 3.19: Network predictions (orange) over the full domain compared to the true function (black). Training points (blue) are highlighted.

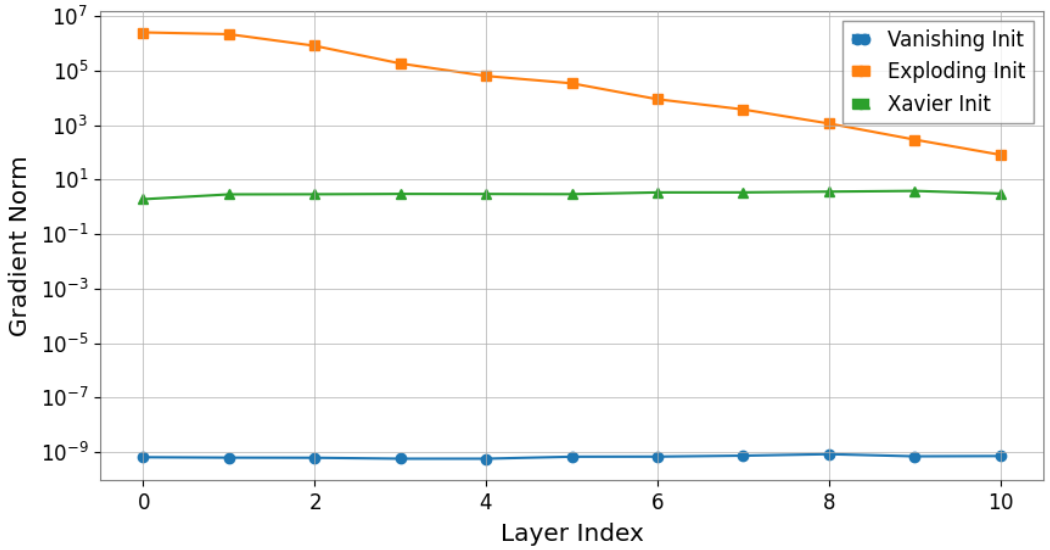


Figure 3.20: Layer-wise gradient norms for vanishing (circles), exploding (squares), and Xavier (triangles) initialisations in a 10-layer network.

We consider a deep neural network with 10 hidden layers of width 128 trained to approximate the simple function $f(x) = \sin(x)$ over $x \in [-2, 2]$. Three different weight initialisation strategies are evaluated. *Vanishing initialisation* weights drawn from a normal distribution with very small standard deviation ($\sigma = 0.01$). *Exploding initialisation* weights drawn from a normal distribution with large standard deviation ($\sigma = 2.0$). Finally, *Xavier initialisation* weights initialised using the Xavier uniform scheme [155].

For each network, we compute the L_2 norm of the gradients of all weight matrices with respect to the mean squared error loss after a single forward-backward pass. We see in Fig. 3.20 that the vanishing initialisation produces extremely small gradients in all layers, leading to slow or stalled learning (vanishing gradients), while the exploding initialisation produces excessively large gradients (where parameter updates become unmanageably large, causing unstable training–loss oscillations or NaN outputs), potentially causing instability. The Xavier initialisation maintains well-scaled gradients across layers, supporting stable training.

As discussed earlier in section 3.3 and intuition box 3, activation functions introduce non-linearities in neural networks, enabling the network to model complex functions. Here we will see in a bit more mathematical detail why the smoothness and effective nullification of all negative input by the ReLU function leads to undesirable behaviours. To do that, let us consider the comparison between the ReLU and SiLU [151] activation functions (see equations 3.59 and 3.60). The derivatives of the two

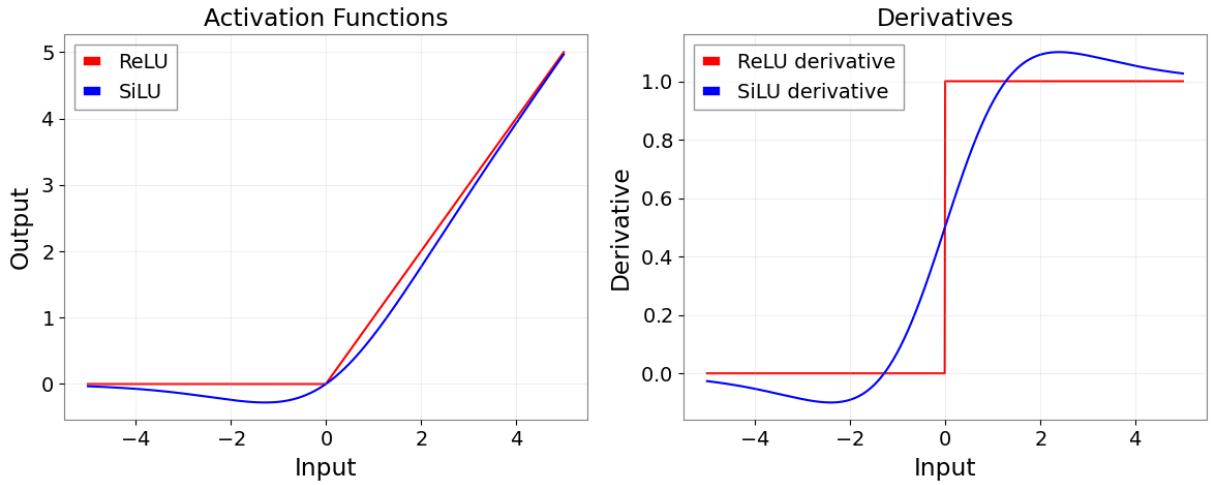


Figure 3.21: Comparison of ReLU and SiLU activation functions (left) and their derivatives (right) as functions of input x.

activation functions with respect to x are given by

$$\frac{d}{dx}\text{ReLU}(x) = \begin{cases} 1, & x > 0, \\ 0, & x \leq 0, \end{cases} \tag{3.62}$$

$$\frac{d}{dx}\text{SiLU}(x) = \sigma(x) + x\,\sigma(x)(1 - \sigma(x)). \tag{3.63}$$

Figure 3.21 illustrates the functional forms of both activations and their derivatives across the input domain $x \in [-5, 5]$. ReLU is piecewise linear and can suffer from "dead neurons" for $x \leq 0$, where the gradient of the neuron is zero for *any* input [156]. This essentially means that the total representative capacity of the network is reduced and we are operating on effectively a smaller network with potentially a non-optimal learning architecture. In contrast, SiLU is smooth and non-monotonic, with non-zero gradient everywhere, which can help the smooth flow of gradients during training.

We investigate the prevalence of inactive or "dead" neurons in deep networks under different activation functions. The dataset consists of $n_{\text{samples}} = 1000$ input vectors, each of dimension 5. The input vectors $X \in \mathbb{R}^{1000\times 5}$ are drawn from a standard normal distribution. The corresponding output values $y \in \mathbb{R}^{1000\times 1}$ are generated as the sum of the five input features of each vector, with additive Gaussian noise of standard deviation 0.1. More concretely,

$$y = \sum_{i=1}^{5} X_i + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.1^2). \tag{3.64}$$

During training, the activations of each hidden layer are stored. After training, neurons are considered "dead" if their activations remain identically zero across the dataset (for ReLU) or below a small

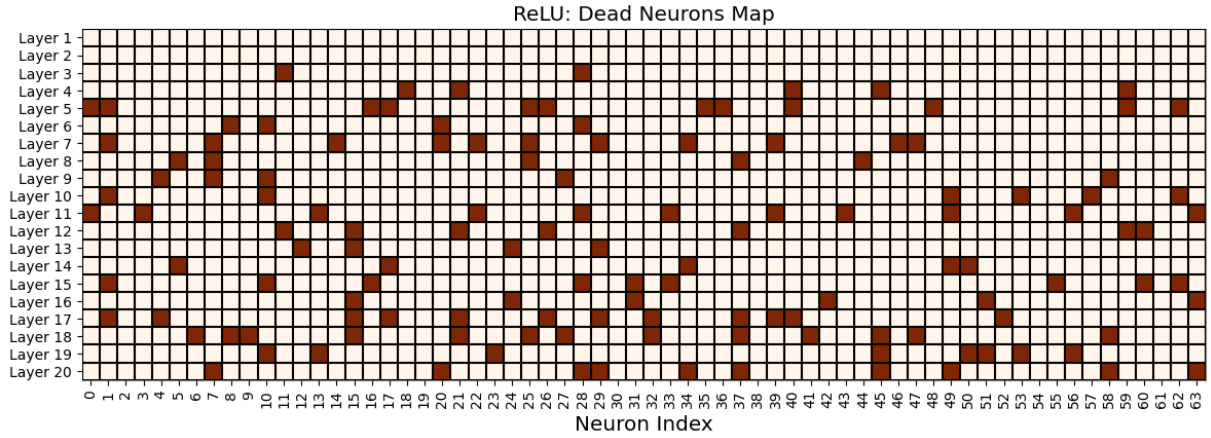


Figure 3.22: Layer-wise map of dead neurons for a ReLU-activated network with 20 hidden layers and 64 neurons per layer. Red cells indicate inactive neurons across the dataset, highlighting the occurrence of dead units in deep networks.

threshold around zero (for SiLU). This provides a layer-wise map of inactive neurons, quantifying the proportion of units that fail to contribute to the network's representation. Such dead neurons can impair model expressivity and indicate suboptimal activation function choice or poor initialisation. We see in Fig. 3.22 that the ReLU network produces many dead neurons and reduces effective learning capacity of the network. We don't present the equivalent for the SiLU network here, since it did not produce any dead neurons when a threshold of 10^{-3} was used to determine the presence of dead neurons. This threshold is somewhat arbitrary, and a higher threshold will lead to a non-zero count, but this count might not be valid since low activation neurons work to de-emphasise certain input features during training and are essential to training. For the SiLU network, one can evaluate whether removing neurons that appear "dead"–for example, by freezing their trainable parameters—actually degrades performance, as measured by an increase in loss. Additionally, because the derivative of the SiLU activation is nonzero everywhere, neurons that temporarily exhibit near-zero activations can gradually recover during training, meaning that apparent inactivity does not necessarily imply permanent loss of representational capacity.

3.10.7 Data-Limited vs Model-Limited Regimes

This example demonstrates the interplay between data availability and model capacity in neural network training. The target function is defined using out usual moderately complex function in equation 3.57 and datasets of varying sizes are generated to probe different learning regimes.

Data-Limited Regime. A large-capacity network with multiple hidden layers and many neurons is trained on a very small dataset ($n_{\text{train}} = 5$). Validation is performed on a denser dataset. In this regime, the network can perfectly interpolate the few training points, leading to *overfitting*, where the training loss decreases rapidly, but the validation loss remains high. See Fig. 3.23 (*top*) for a visualisation; the overfitting example is similar to that seen in section 3.10.5.

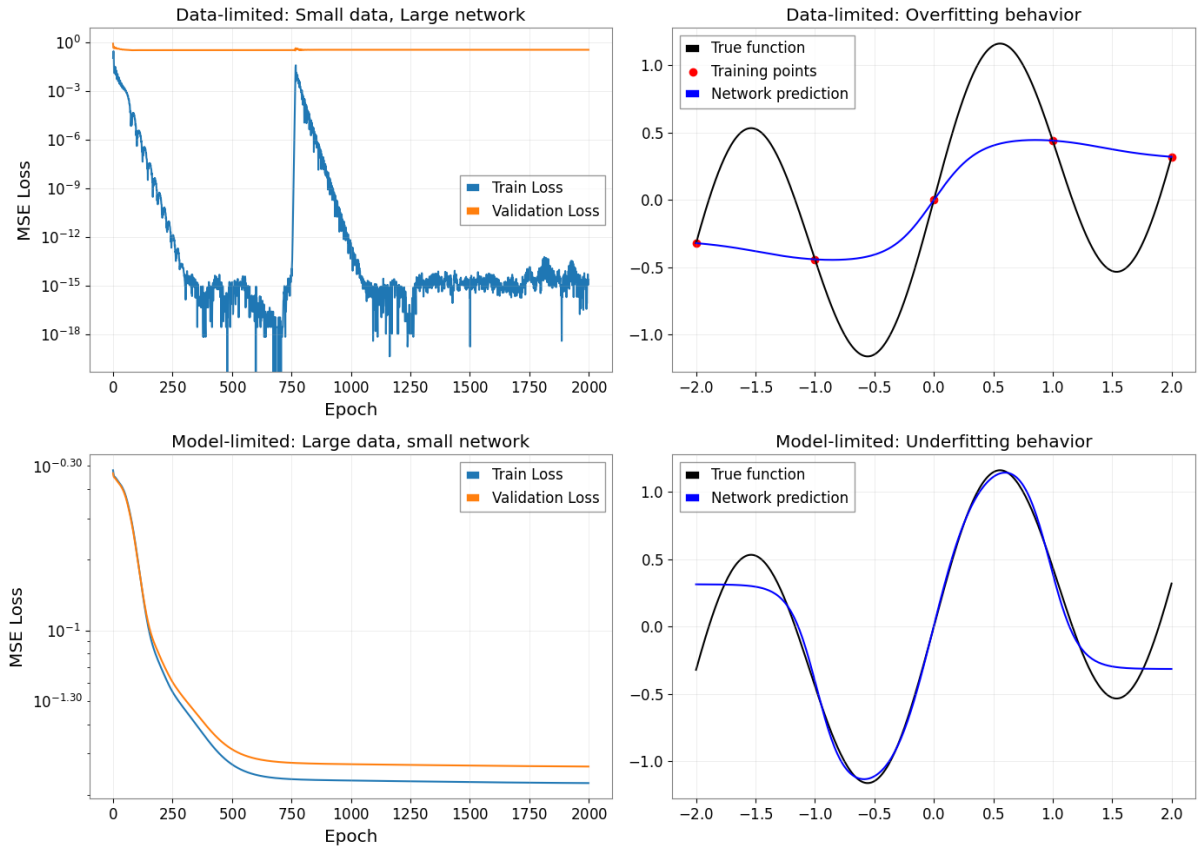


Figure 3.23: Comparison of data-limited and model-limited training regimes. Top row: small dataset with a large network, showing overfitting. Bottom row: large dataset with a small network, showing underfitting. Left column: training and validation losses; right column: network predictions versus true function.

Model-Limited Regime. A small-capacity network with only a single hidden layer and few neurons is trained on a large dataset ($n_{\text{train}} = 3000$). Here, the network is unable to capture the full complexity of the target function, leading to *underfitting*. Both training and validation losses remain high, and the predicted function exhibits a smoothed approximation that misses important variations in the true function. See Fig. 3.23 (*bottom*) for a visualisation. Underfitting arises when the model lacks sufficient capacity to capture the relevant structures in the data, resulting in high training and test error. Moreover, we can also typically see a constant gap between the test and training errors without signs of overfitting. This is known as a *generalisation gap* and is indicative of an underfitting network or a difference between the training and test dataset.

3.10.8 Computational Cost of Representational Capacity

This experiment explores how increasing the width of a neural network affects its ability to approximate a target function. The target function is again defined using equation 3.57 and a dataset is generated uniformly over $x \in [-2, 2]$.

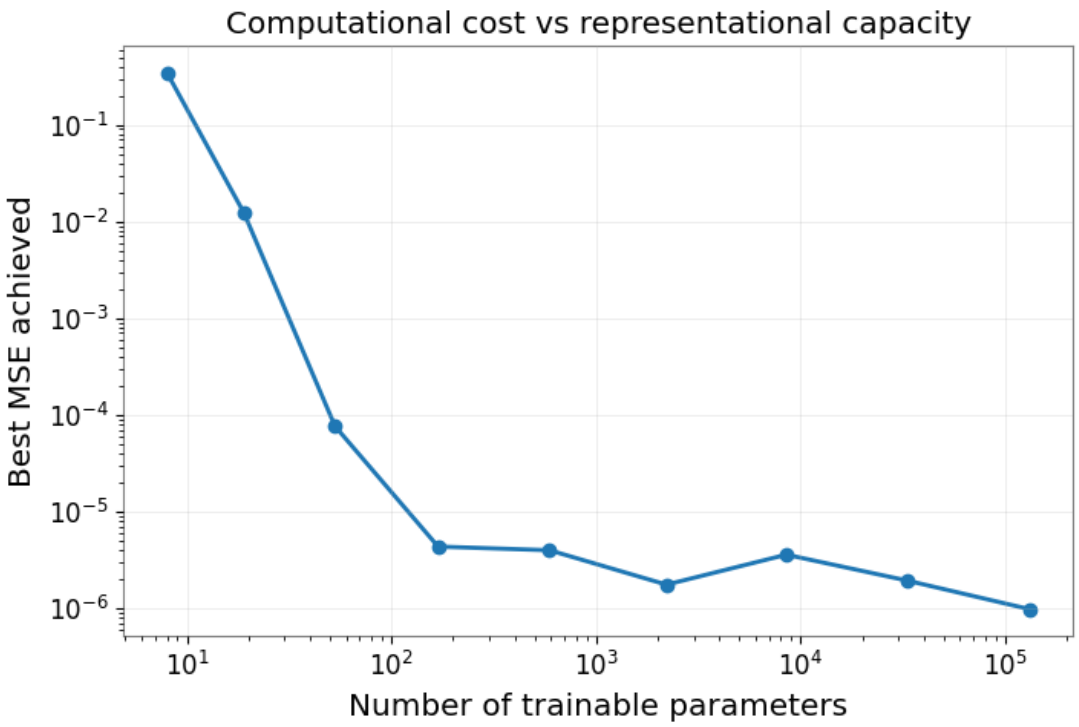


Figure 3.24: Best mean squared error achieved as a function of the total number of trainable parameters for networks of varying width.

The experiment systematically increases the hidden layer width $w \in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$ while keeping the number of layers fixed. For each configuration, the number of trainable parameters is recorded. Increasing width increases the network's representational capacity, allowing it to achieve lower MSE on the training data. However, larger capacity without overfitting does not always mean significantly lower losses and better generalisation. The extra learnable information from a given input dataset quickly reduces as model capacity increases without taking any domain knowledge into account. The resulting relationship between parameter count and best achievable loss, shown in Fig. 3.24, highlights the trade-off between computational cost and representational power. Using more trainable parameters increasing computational cost but the resultant representational capacity gains is marginal as seen in the saturating best mean loss achieved in learning a function.

3.11 Inductive Bias

In machine learning, an *inductive bias* refers to any assumption that restricts or guides a model's search through its hypothesis space. Rather than allowing an algorithm to consider all possible mappings between inputs and outputs, inductive biases narrow the set of candidate functions to those deemed more plausible given prior knowledge or architectural constraints. This is essential because, without such biases, learning from finite data would be intractable: infinitely many functions could explain the observed examples equally well but generalise poorly. Inductive biases arise in many forms, includ-

ing choices of network architecture, activation functions, regularisation schemes, optimisation procedures, and data representations. Each of these elements shapes the space of functions the model can learn, thereby influencing both performance and generalisation. Understanding and deliberately introducing inductive biases allows practitioners to encode domain knowledge, improve data efficiency, and obtain models that are better aligned with the underlying structure of the problem at hand.

In this section, we choose a handful of different mechanisms and examples that serve as a primer to the use of inductive bias in forthcoming chapters. We first introduce the *attention mechanism*, which illustrates how models can dynamically weight features to capture relationships across varying scales. We then discuss *equivariance*, which describes a method to preserve transformations of the input data consistently through the network architecture, and contrast this with *invariance*, where representations are explicitly or implicitly made insensitive to certain transformations. Next, we explore the *implicit inductive bias of gradient descent*, highlighting how the optimisation process itself can favour particular classes of solutions even without explicit constraints. Finally, we address the role of *data augmentation*, re-framing it in terms of learned invariances rather than fixed architectural constraints. Taken together, these mechanisms provide a foundation for understanding how inductive biases emerge and can be deliberately employed to improve model performance, guide learning, and align representations with the structure of the underlying data.

Inductive bias is fundamental to learning, as it constrains the hypothesis space and enables generalisation beyond observed data. As [157] emphasised, without some form of bias, a learning algorithm cannot prefer one generalisation over another, making learning from finite data impossible. In modern deep learning, inductive biases arise not only from architectural choices but also from the optimisation process itself. For instance, [158] show that gradient descent on linearly separable data implicitly biases the solution towards maximum-margin classifiers, even in the absence of explicit regularisation. Together, these works illustrate that both explicit design and implicit optimisation dynamics contribute to the inductive biases that shape model behaviour and generalisation.

We use this section to also describe a couple of commonly used advanced neural network architectures, their natural inductive biases and their applications. We will discuss the Convolutional Neural Network (CNN) architecture using the concepts of equivariance and invariance, and explore why this architecture popularised ML as a field to the general populace although ML has existed since the 1940s. Alongside the attention mechanism, we will explore the transformer architecture responsible for initiating a new era of natural language processors (eg. ChatGPT, Gemini, Claude, Copilot etc.) for wide ranging applications.

3.11.1 Equivariance

Equivariance describes a property of a function whereby certain transformations applied to the input lead to predictable transformations of the output. More concretely, let f denote a mapping from

inputs x to outputs $f(x)$. We say that f is *equivariant* to a transformation T if there exists a corresponding transformation S on the output such that

$$f(T(x)) = S(f(x)). \tag{3.65}$$

That is, applying T to the input and then evaluating f produces the same result as first evaluating f and then applying S to the output. Many natural operators in mathematics and physics are equivariant. For example, the laws of physics are typically equivariant under translations and rotations. Equivariance in physics formalises the idea that applying a transformation to a system before or after evaluating its governing equations yields equivalent results. Maxwell's equations are a fundamental example that are Lorentz equivariant: applying a Lorentz transformation before or after evaluating them yields the same form.

A familiar and classic example in machine learning arises in CNNs. Before we delve into the mathematics of equivariance, it is useful to first formalise the CNN architecture. CNNs are a class of parametric function approximators designed to exploit symmetries present in structured data, most notably translation symmetry in signals such as images and time series. From a mathematical perspective, these architectures can be understood as compositions of structured linear operators (convolutions), pointwise non-linear maps, and pooling operators, arranged to produce representations that are equivariant or invariant under group actions.

Let X denote an input space of signals and Y an output space. A CNN defines a parameterised function

$$f_\theta : X \to Y, \tag{3.66}$$

where θ denotes the collection of all learnable parameters. In the case of spatial data (e.g. images),

$$X = \mathbb{R}^{H \times W \times C}, \tag{3.67}$$

where H and W are spatial dimensions and C is the number of channels. For time series data, this reduces naturally to a one-dimensional domain. A CNN is constructed as a composition

$$f_\theta = \phi_L \circ \mathcal{L}_L \circ \cdots \circ \phi_1 \circ \mathcal{L}_1, \tag{3.68}$$

where each $\mathcal{L}_\ell$ is a structured linear operator (typically a convolution or pooling operator) and each ϕ_ℓ is a pointwise nonlinearity. Each convolutional layer applies a discrete convolution to its input. Let

$$x : \mathbb{Z}^d \to \mathbb{R}^{C_{\text{in}}}, \quad k : \mathbb{Z}^d \to \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}, \tag{3.69}$$

where $d = 2$ for images and $d = 1$ for time series. The convolution operator C_k is defined by

$$(C_k x)_c(u) = \sum_{c'=1}^{C_{\text{in}}} \sum_{v \in \mathbb{Z}^d} k_{c,c'}(v)\, x_{c'}(u - v). \tag{3.70}$$

This operator is linear. When combined with a bias term $b \in \mathbb{R}^{C_{\text{out}}}$, the output of a convolutional layer is

$$x_\ell = C_{k_\ell}(x_{\ell-1}) + b_\ell. \tag{3.71}$$

To achieve expressive power beyond linear models, each convolutional layer is followed by a pointwise nonlinearity

$$(\phi(x))_c(u) = \sigma(x_c(u)), \tag{3.72}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function such as the rectified linear unit. The composition of convolution and pointwise nonlinearity preserves the translation operation, ensuring that intermediate representations transform consistently under translations of the input. The output of each layer is referred to as a *feature map*, and successive layers compute increasingly abstract features by composing local operations.

We can make the notion of translation equivariance rigorous for the basic continuous convolution operation and discuss how this is useful in ML in its discretised form.

Let $f : \mathbb{R}^n \to \mathbb{R}$ denote an input signal (or image in 2D) and $k : \mathbb{R}^n \to \mathbb{R}$ denote a kernel (also known as filter). The continuous convolution $k * f : \mathbb{R}^n \to \mathbb{R}$ is defined by

$$(k * f)(x) \;=\; \int_{\mathbb{R}^n} k(x - y)\, f(y)\, dy, \qquad x \in \mathbb{R}^n. \tag{3.73}$$

Equivalently one can use the alternative form,

$$(k * f)(x) = \int k(y) f(x - y)\, dy; \tag{3.74}$$

both are the same by a change of variable.

For a translation vector $t \in \mathbb{R}^n$, we define the translation (shift) operator $\mathcal{L}_t$ acting on functions by

$$(\mathcal{L}_t f)(x) \;=\; f(x - t), \qquad x \in \mathbb{R}^n. \tag{3.75}$$

This convention means that $\mathcal{L}_t$ shifts the function to the right by t (so a feature originally at location $x = x_0$ moves to $x_0 + t$). Intuition suggests that a kernel operating on a location translated input should behave exactly the same as it did with the original input, but in the new location. Let us make this intuition more concrete with the intuition box 4.

Intuition 4: Translation Equivariance of the Convolution Operation

We now show that convolution commutes with translation in the sense that convolving a translated input is the same as translating the convolved output:

$$k * (\mathcal{L}_t f) \;=\; \mathcal{L}_t (k * f). \tag{3.76}$$

Proof. For any $x \in \mathbb{R}^n$,

$$\big(k * (\mathcal{L}_t f)\big)(x) = \int_{\mathbb{R}^n} k(x-y)\,(\mathcal{L}_t f)(y)\,dy = \int_{\mathbb{R}^n} k(x-y)\,f(y-t)\,dy.$$

Make the change of variable $u = y - t$ (hence $y = u + t$, $dy = du$). Then

$$\begin{aligned}\big(k * (\mathcal{L}_t f)\big)(x) &= \int_{\mathbb{R}^n} k\big(x-(u+t)\big)\,f(u)\,du = \int_{\mathbb{R}^n} k\big((x-t)-u\big)\,f(u)\,du \\ &= (k * f)(x-t) = \big(\mathcal{L}_t (k * f)\big)(x).\end{aligned}$$

This establishes the desired equality for all x, i.e.

$$k * (\mathcal{L}_t f) = \mathcal{L}_t (k * f). \tag{3.77}$$

We have thus proved that the convolution operator in translation equivariant: shifting the input simply shifts the output, without altering the underlying responses.

Condition 1. The equalities above hold under the usual regularity/integrability conditions allowing the change of variables inside the integrals. For discrete signals the same identities hold by replacing integrals with sums.

Condition 2. The proofs rely crucially on using the *same fixed kernel* k for the translated input. If the kernel itself depends on absolute coordinates of the features, equivariance is lost.

Condition 3. In practical implementations of CNNs, strict translation equivariance is not always preserved. This deviation arises primarily from two factors: [i] *Boundary handling*: Finite input domains require padding (e.g., zero-padding or reflection padding). Since the choice of padding does not perfectly mimic a true translation of the signal in an infinite domain, equivariance can be broken near the edges. [ii] *Discrete sampling*: Continuous convolutions are approximated on a discrete grid. Translations smaller than the grid resolution cannot be represented exactly, and translations that do not align with the grid may be interpolated or truncated. These practical considerations mean that convolutional layers are not perfectly translation-equivariant, but they are typically *approximately equivariant* for most translations within the domain. This approximate property is sufficient in most applications, as read signals are inherently discretized and finite.

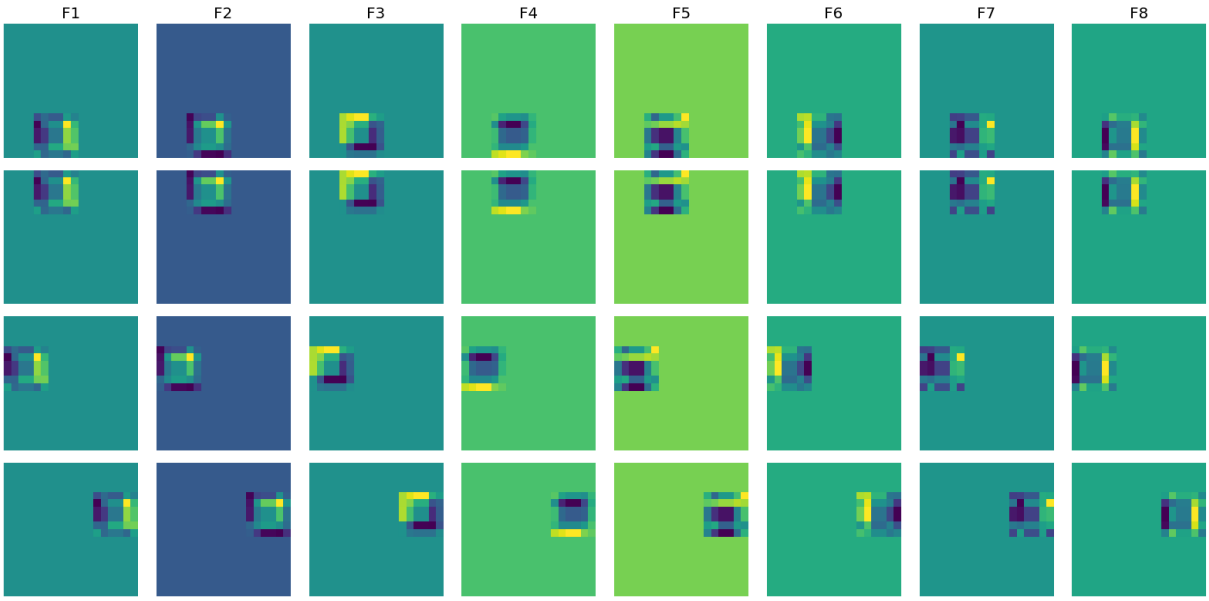


Figure 3.25: Feature maps of the first convolutional layer for 16 × 16 square images at four different positions, visualized after per-feature normalisation.

To solidify this concept even further, we demonstrate translation equivariance in a simple CNN. We generate a set of simple 16×16 binary images, each containing a small 4×4 white square positioned at one of four locations: bottom, top, left, or right. These images are stacked into a batch and passed through a simple two-layer CNN with padding to preserve edge information. The first convolutional layer has 8 filters, and the second has 16. After computing the activations of the first convolutional layer, we visualise the resulting feature maps along each column of Fig. 3.25. Each feature map uses different fixed kernels, and thus exact equivariance will exist only along each column of the figure. Equivariance is known to provide provably better generalisation compared to non-equivariant models [159] and has been shown previously to be more efficient in training data and scale [160].

Why does equivariance increase training data efficiency? Say we have a picture of a cat. We want to built a system such that if the cat moves to a different position and sits down the same way, the model still identifies the new picture as a cat without additional learning. To a typical fully connected neural network, these two positions would produce different internal feature representations and thus are technically two different cats. So, for such a network we must move the cat to every possible position and train it to still identify it as the same cat. However, a CNN can do the same thing with exactly one training image.

Optional Flavour 1

History and Significance of Convolution to Machine Learning. This beautiful concept suddenly made training a neural network computationally feasible with a reasonably finite amount of training data. The introduction of the convolution operator into machine learning is an inflection point in the field and marked the birth of modern day machine learning.

3.11.2 Invariance

An important form of inductive bias in neural networks is invariance, where the architecture is designed to produce identical or similar representations despite transformations of the input. A common example is translation invariance, where a feature should be recognised regardless of its position in the input space. CNNs naturally capture local translation equivariance as seen in section 3.11.1, and with training, different images can be made to produce approximately invariant features by placing them under the same class label. But, complete invariance is not guaranteed even with sufficient training.

In this section, we construct a simple CNN that employs something called *global average pooling* to enforce and guarantee translation invariance.

Pooling operators reduce spatial or temporal resolution by aggregating values over local neighbourhoods. A typical example is average pooling (continuing the notation convention from section 3.11.1):

$$(\mathcal{P}x)(u) = \frac{1}{|B|} \sum_{v \in B(u)} x(v), \tag{3.78}$$

where $B(u)$ is a local neighbourhood around u. Pooling operators are not strictly equivariant, but they introduce stability and approximate invariance to small translations. This trade-off is central to the design of CNN architectures intended for classification tasks.

For tasks such as classification, equivariant feature maps must ultimately be mapped to invariant outputs. This is commonly achieved via global pooling:

$$f(x) = \psi\left(\sum_{u \in \Omega} x_L(u)\right), \tag{3.79}$$

where ψ is a classifier acting on aggregated features. This construction ensures

$$f(T_\tau x) = f(x), \tag{3.80}$$

thereby producing translation-invariant predictions.

We make this concrete using a simple example. Synthetic images containing a square positioned

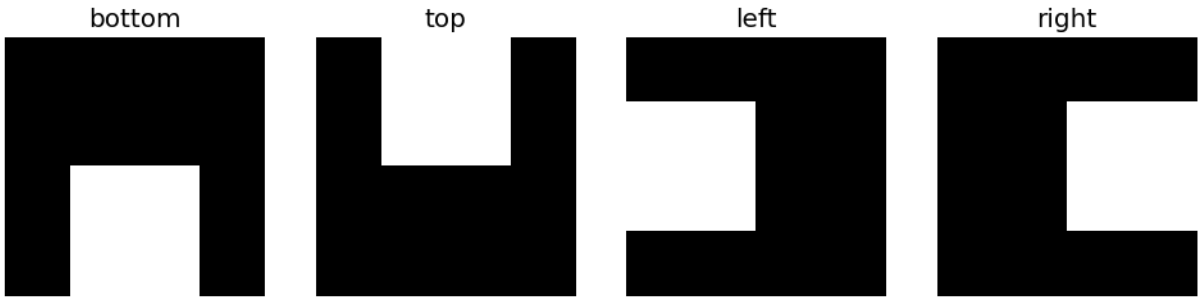


Figure 3.26: Input images with a square placed at four different positions.

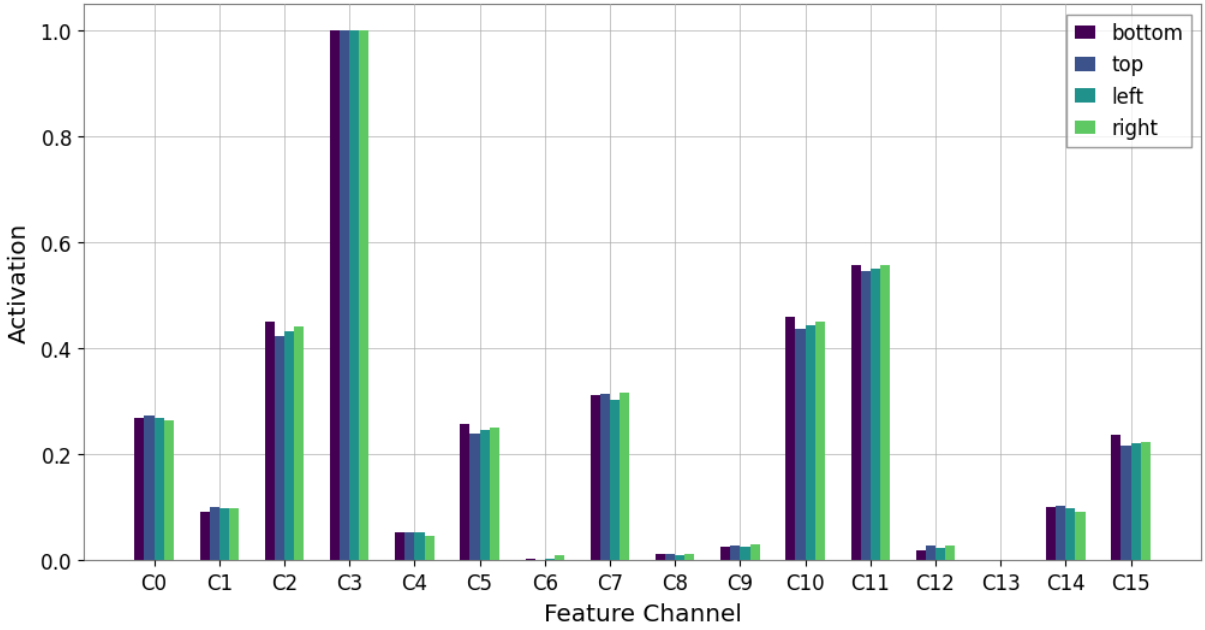


Figure 3.27: Illustration of translation invariance through global average pooling. The grid cells show normalised feature activations across channels after passing the images through a CNN with GAP. Despite the varying input positions, the feature representations remain consistent, highlighting the invariance property.

at the top, bottom, left, or right of the canvas (see Fig. 3.26 for reference) are passed through a CNN model with a global pooling operation applied on the last layer. Without this pooling operation, the resulting feature representations would differ significantly depending on the position of the square, as the spatial location of activations would be preserved. By applying global average pooling, however, the spatial dimension is collapsed, producing feature vectors that are largely independent of position. The plotted activations in Fig. 3.27 show that the model extracts consistent feature patterns across all square positions, demonstrating how architectural choices can encode inductive bias towards invariance.

What does a CNN learn? So far we have discussed the CNN architecture without explicitly defining how the kernel function is defined and how it influences the learning outcome. The kernel function is fully fixed in the previous examples to visualise the notion of equivariance and invariance. In a practical learning scenario, the kernel *is* what the network primarily learns in each layer.

The kernels $\{k_\ell\}$ are not fixed but learned from data by solving an optimisation problem:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L}(f_\theta(x), y)\right]. \tag{3.81}$$

From a mathematical standpoint, learning kernels corresponds to learning a set of basis functions that optimally extract task-relevant local patterns. In early layers, learned kernels often resemble local edge detectors (similar to that seen in section 3.11.1) or frequency-selective filters for time series data; in deeper layers, kernels encode increasingly abstract features. The restriction to convolutional kernels significantly reduces the dimensionality of the hypothesis space relative to fully connected networks, improving statistical efficiency while enforcing symmetry constraints.

How does a CNN aggregate information? So each layer in a CNN learns kernels that are to be convolved over the input signal. This operation sounds pretty similar to matched filtering because, locally, it very literally is. However, each kernel is typically smaller than the input and its values quickly lose physical meaning with layer depth and non-linearity. So what does each kernel see and learn?

The *receptive field* of a unit in layer ℓ is the subset of the input influencing its value. If the kernel size of layer ℓ is K_ℓ, then the receptive field size satisfies

$$r_\ell = r_{\ell-1} + (K_\ell - 1). \tag{3.82}$$

Consequently, deeper layers integrate information over progressively larger regions. CNNs therefore implement a hierarchical local-to-global computation, which aligns well with the structure of natural signals. In a typical CNN, stacking layers increases the temporal receptive field, allowing CNNs to model long-range dependencies while maintaining computational efficiency. Pooling introduces robustness to small temporal misalignments, which is often desirable in real-world time series.

Intuition 5: Arguably the best explanation of convolution for ML

As supplementary reading (or in this case viewing) material, I provide the following: The best explanation that I have come across for the use of convolution in a neural network is by 3Blue1Brown in the following video: https://www.youtube.com/watch?v=KuXjwB4LzSA. I encourage the reader to watch the video in its entirety (not sponsored, of course). The entire channel is a treasure trove for anyone who finds mathematics to be too abstract, but wants to find the motivation to learn more of it.

3.11.3 Attention Mechanism

In its simplest form, the *attention mechanism* [161] assigns a set of non-negative weights to input features or elements and then forms a weighted combination of them. This mechanism biases the model to focus on the most relevant components of the input while still retaining information from the entire

context, and it generalises naturally to sequences, images, or sets. This experiment demonstrates how a simple attention mechanism can identify relevant features in a dataset with heterogeneous scales. A synthetic dataset with $n_{\text{samples}} = 2000$ was generated using a uniform random distribution for each of the $n_{\text{features}} = 5$ independent features. The input features have varying scales: for instance, X_0 is shifted by +5, X_3 by +4, and X_4 by −5, while X_1 and X_2 remain unaltered. The target y is defined as

$$y = 2X_1 - X_3 + 0.1\epsilon, \quad \epsilon \sim \mathcal{N}(0,1), \tag{3.83}$$

so that features X_1 and X_3 are most predictive, while the other features contribute only noise.

The network is a single-layer attention model:

$$a_i = \frac{\exp(w_i)}{\sum_j \exp(w_j)}, \tag{3.84}$$

$$\tilde{x}_i = a_i x_i, \tag{3.85}$$

$$\hat{y} = W\tilde{x} + b, \tag{3.86}$$

where w_i are learnable attention logits, a_i are the attention coefficients (sigmoid-normalised), $\tilde{x}_i$ are the weighted inputs, and W, b parameterise the output linear layer.

Attention mechanisms are inherently scale-invariant in this setup because the attention weights a_i are computed via a sigmoid function over learnable parameters, not directly from feature magnitude. This allows the network to properly focus on informative features, regardless of their absolute scale. For example, even though X_0 and X_4 have larger magnitudes due to added shifts, the attention correctly assigns higher weight to X_1 and X_3 (as shown in Fig. 3.28), which truly affect the target. Moreover, features with similar scales, such as X_1 and X_2, do not confuse the attention mechanism. The learning process adjusts w_i such that only the predictive features are weighted strongly, while irrelevant or redundant features receive low attention.

The left panel of Fig. 3.28 shows the distribution of input features with boxplots, highlighting the differing scales. The right panel shows the learned attention weights, with important features X_1 and X_3 clearly emphasised (red stars). This confirms that the attention mechanism can discriminate relevant features despite differences in scale and similarity among other inputs.

A more complex form of attention is often seen in the transformer architecture [162]. The transformer architecture departs fundamentally from convolutional sequence models by relying exclusively on attention mechanisms [163]. This architectural choice induces a markedly different set of inductive biases, with important consequences for generalisation, sample efficiency, and representational capacity [164].

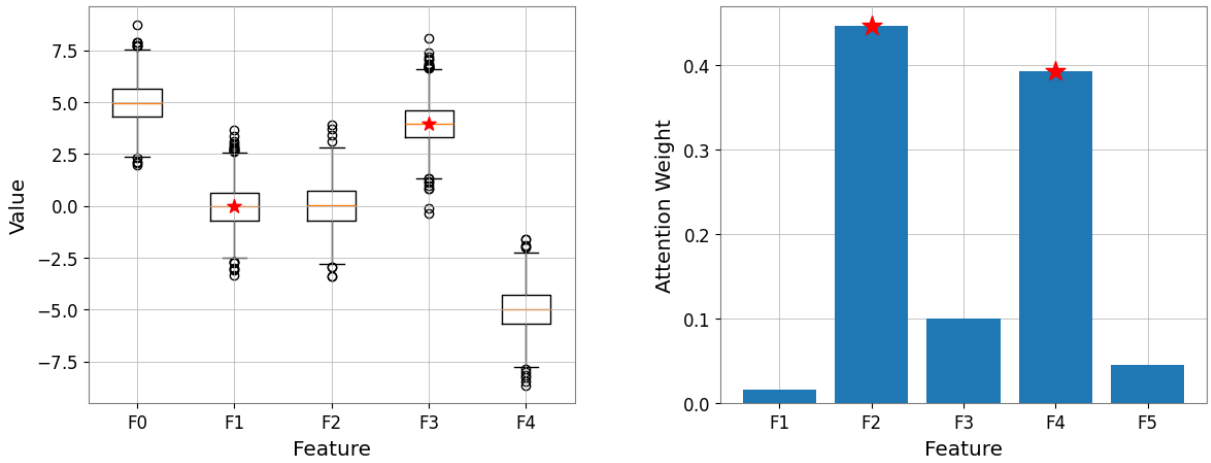


Figure 3.28: Left: Boxplots of input feature distributions showing heterogeneous scales. Right: Learned attention weights from a simple attention network. Red stars indicate truly important features X_1 and X_3. Attention correctly identifies relevant features, unaffected by large or similar-scale irrelevant features.

Let an input sequence of length n be represented as a matrix

$$X = [x_1, x_2, \ldots, x_n]^\top \in \mathbb{R}^{n \times d}, \tag{3.87}$$

where $x_i \in \mathbb{R}^d$ denotes the embedding of the i-th token (a discrete unit of a sequence—such as a word, subword, symbol, or individual time-series sample—that serves as the basic element processed by the model). In the absence of convolution, the transformer introduces positional information through an explicit positional encoding $P \in \mathbb{R}^{n \times d}$, yielding the effective input

$$H^{(0)} = X + P. \tag{3.88}$$

This additive formulation reflects a weak inductive bias with respect to sequence order: positional structure is not hard-coded into the architecture but instead provided as auxiliary information that the model can learn to exploit. In transformer attention, each time point t is updated as a weighted combination of all time points:

$$\tilde{x}_t = \sum_{s=1}^{n} \alpha_{ts}\, x_s, \tag{3.89}$$

where the weights α_{ts} reflect the similarity between time points t and s:

$$\alpha_{ts} = \frac{\exp\big((x_t W_1) \cdot (x_s W_2)\big)}{\sum_{u=1}^{n} \exp\big((x_t W_1) \cdot (x_u W_2)\big)}, \tag{3.90}$$

with learned matrices $W_1, W_2 \in \mathbb{R}^{d \times d_k}$ and $\cdot$ denoting the dot product. This operation can be inter-

preted in analogy to a covariance matrix. If we define the similarity matrix

$$S = XW_1(XW_2)^\top \in \mathbb{R}^{n\times n}, \tag{3.91}$$

then each row of S contains the unnormalised pairwise similarities between one time point and all others. Applying the row-wise softmax produces the attention weights:

$$A_{ts} = \frac{\exp(S_{ts})}{\sum_{u=1}^{n}\exp(S_{tu})}, \quad \tilde{X} = AX, \quad t, s = 1, \ldots, n. \tag{3.92}$$

In this sense, attention acts like a data-driven covariance, where each time point aggregates information from others proportional to their similarity. Unlike the classical covariance, however, the similarities are learned via W_1 and W_2 and normalised to sum to one, allowing flexible modelling of both short- and long-range temporal dependencies without assuming locality or sequential order beyond positional encodings.

The transformer's attention mechanism induces an inductive bias towards pairwise, content dependent interactions between all time points. Unlike recurrent models, which prioritise local temporal continuity, or convolutional models, which encode fixed temporal locality and translation equivariance, transformers impose no intrinsic assumptions about temporal neighbourhoods or ordering beyond those introduced through positional encodings. Overall, transformers applied to time series exhibit a deliberately weak inductive bias. By avoiding strong assumptions about temporal structure, they provide high expressive flexibility but demand substantial data, accounting for both their strong performance at scale and their comparatively weaker behaviour in low-data settings where stronger temporal inductive biases can be advantageous.

3.11.4 Implicit Bias of Neural Networks to Learn Simple Functions

This example illustrates the implicit inductive bias of over-parameterised neural networks in learning simple functions. We consider a one-dimensional target function

$$f(x) = \sin(3x) + 0.3\,x \tag{3.93}$$

sampled at $N = 20$ points with small Gaussian noise. A degree-100 polynomial fit to these points exhibits severe oscillations outside the training locations, a manifestation of classical overfitting. In contrast, a feedforward network with a single hidden layer of 20 `Tanh` units ($\sim$ 100 trainable parameters) fits the same data smoothly. Despite the network being highly over-parameterised relative to the number of training points, gradient-based optimisation coupled with the network architecture induces a strong bias towards low-complexity functions. This results in effective generalisation across the test domain, as shown in 3.29.

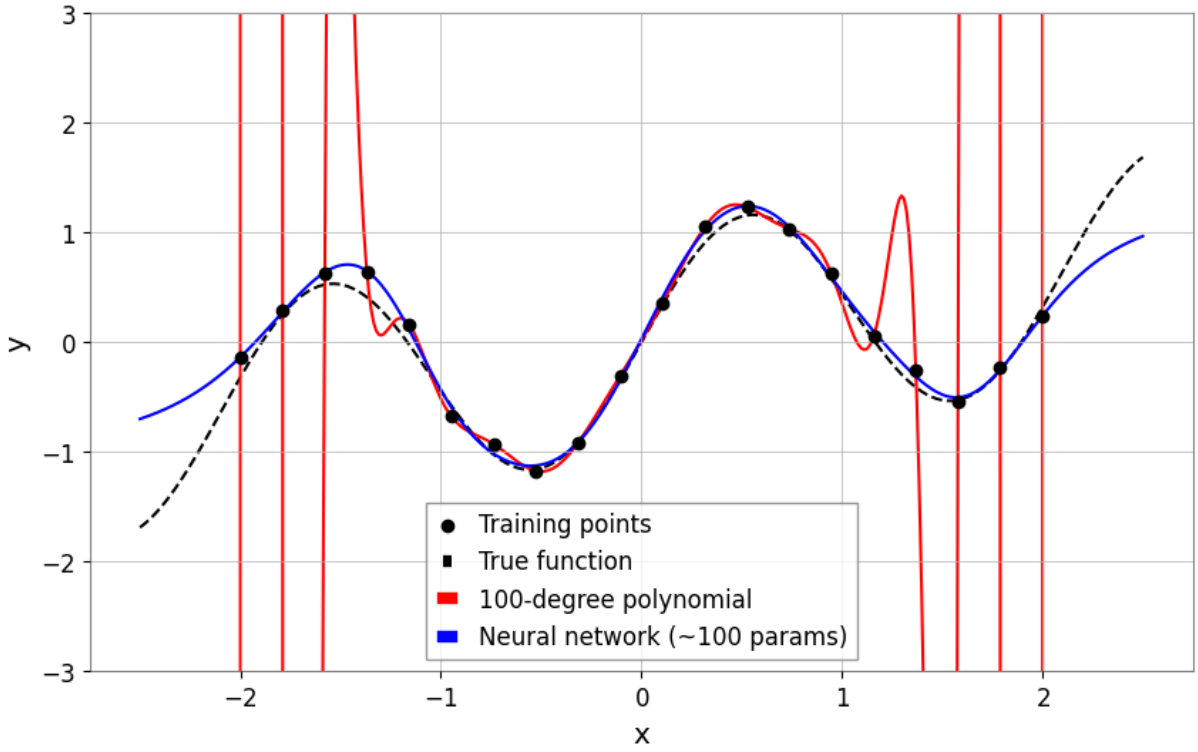


Figure 3.29: Comparison of fits to a one-dimensional function $f(x) = \sin(3x) + 0.3\,x$ based on 20 noisy training points. Black dots indicate the training data, the dashed line shows the true function, the red line is a degree-100 polynomial fit, and the blue line is a neural network fit with ~ 100 parameters.

The comparison demonstrates that, unlike over-parameterised polynomials, neural networks do not simply interpolate the training points but favour solutions that explain the data in the simplest possible manner. Neural networks, thus, strongly constrain the set of learnable solutions and have an implicit inductive bias to learn simple functions over more complex functions [125, 165, 166].

3.11.5 What *is* Data Augmentation?

Typically, in the classification setting, data augmentation refers to the process of systematically increasing the diversity and size of a training dataset by applying label-preserving transformations to the existing data. The goal is to improve a model's ability to generalise to unseen inputs by exposing it to a wider variety of examples during training. However, given our new-found insight into equivariance and invariance, in some cases it can also be seen as a mechanism to approximate this behaviour.

Mathematically, let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ be the original dataset, where $\mathbf{x}_i$ is an input and y_i its corresponding label. Data augmentation applies a transformation function $T_\phi(\cdot)$ parameterised by $\phi \in \Phi$, drawn from a transformation space Φ, to produce new samples

$$(\mathbf{x}'_i, y_i) = \Big(T_\phi(\mathbf{x}_i),\ \ y_i\Big), \tag{3.94}$$

such that $T_\phi(\mathbf{x}_i)$ remains in the same class as $\mathbf{x}_i$. The augmented dataset $\mathcal{D}'$ is then given by

$$\mathcal{D}' = \mathcal{D} \cup \{(\mathbf{x}'_i, y_i)\}. \tag{3.95}$$

It is also possible to do a class-changing transformation instead of a label-preserving one, but this is not traditionally viewed as data augmentation, but rather sample generation. In a classification setting, assuming label-preserving data augmentation, beyond simply enlarging the training set, data augmentation acts as a form of regularisation by introducing variability that forces the model to learn features that are invariant to the applied transformations. This reduces overfitting and improves robustness to distributional shifts. In certain domains, augmentation can also encode domain-specific invariances — for example, rotational invariance in astronomical images or time-translation invariance in gravitational wave time series — thereby embedding prior knowledge into the learning process. This sort of learnt invariance does not guarantee that the network activation will be invariant to the transformation in question, unlike the example shown in section 3.11.2. The invariance of the activation increases with training, and need not necessarily be uniformly invariant for all transformations ϕ considered within the transformation space Φ.

We remove the global average pooling from the example shown in section 3.11.2 and train a neural network on a dataset comprised of a uniform distribution of the four different types of images shown in Fig. 3.26, i.e. $U(I_{\text{top}}, I_{\text{bottom}}, I_{\text{right}}, I_{\text{left}})$. Some noise is added to the positions of the white squares for variation. We see in Fig. 3.30 that, as expected, we achieve learnt invariance of the neural network activations for a testing dataset with the same image distribution as the training dataset. To complete this experiment, we train a network on only one type of input image, I_{top}, with some noise added to the position of the white square, and then test it on $U(I_{\text{top}}, I_{\text{bottom}}, I_{\text{right}}, I_{\text{left}})$. The expected failure of the neural network to achieve invariance for the testing dataset images (see Fig. 3.31) enunciates the learnt nature of the invariance seen in Fig. 3.30. Moreover, it also shows the inability of the neural network to generalise outside its training distribution when it is not inherently invariant to them via its architecture.

The preceding experiments illustrate several fundamental principles of neural network learning. These concepts are directly relevant to GW detection, where neural networks are increasingly employed to extract weak signals from noisy time-series data. GW signals are sparse, embedded in high-dimensional detector outputs, and exhibit structured dependencies across multiple features such as frequency, phase, and polarisation. ML models for GW detection should be designed to prioritise the most discriminative aspects of the waveform. Similarly, insights from capacity and overfitting analyses guide the choice of network depth, width, and regularisation to ensure robust generalisation to unseen signals, which is crucial for real-time detection and parameter estimation pipelines. Thus, the principles explored in controlled experiments provide a conceptual and practical foundation for applying ML techniques to the complex task of GW discovery.

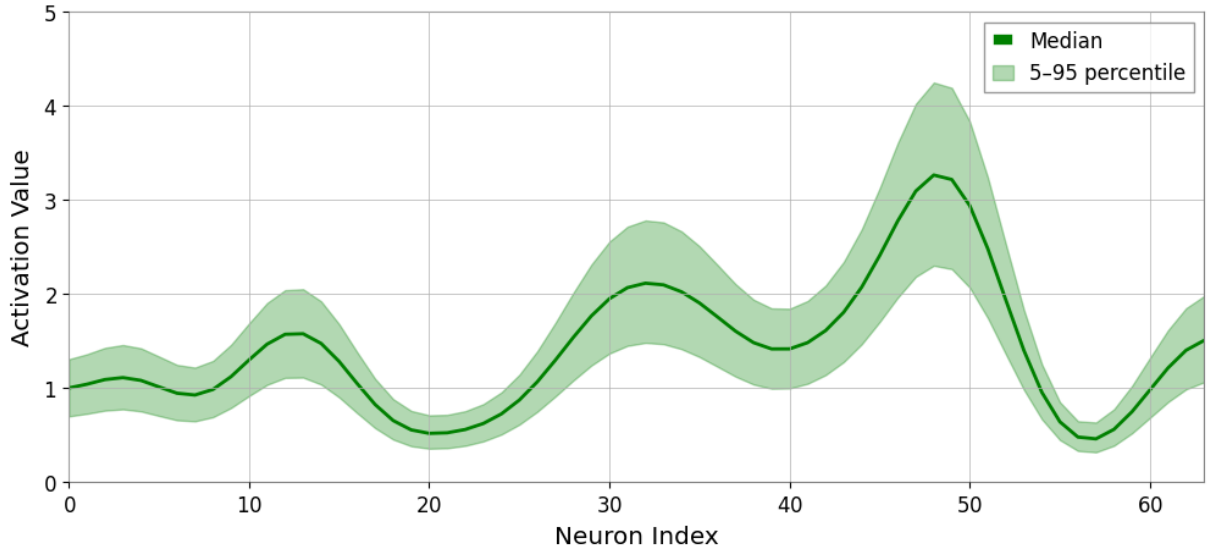


Figure 3.30: Activation summaries and training behaviour of a fully connected network trained on square images shown in Fig. 3.26. The curve shows the smoothed median feature activations with error bands (between the 5th and 95th percentile) for a network trained with augmentation.

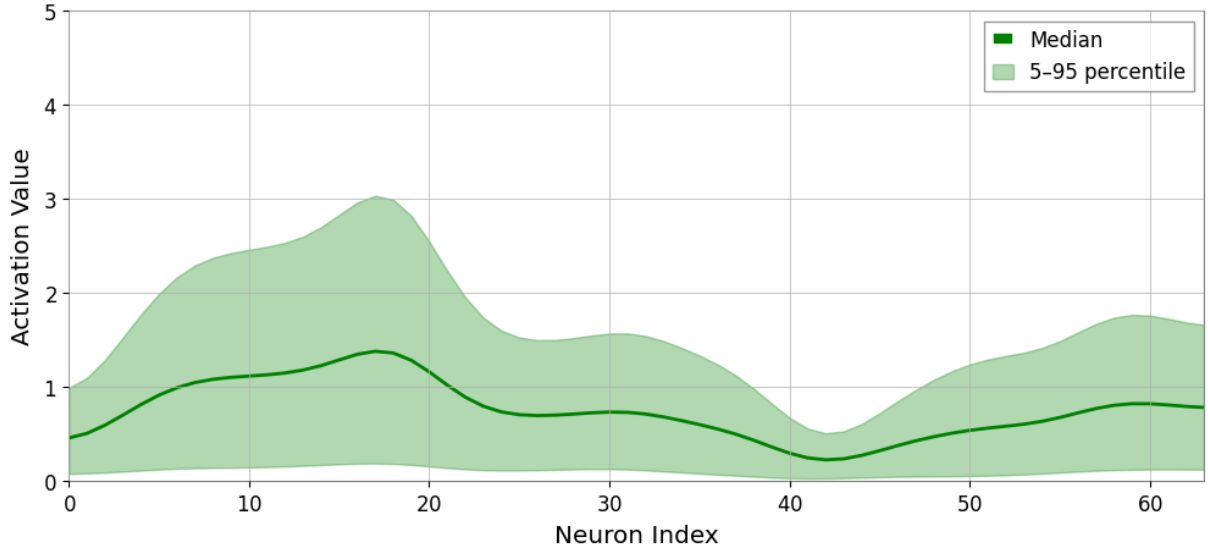


Figure 3.31: Activation summaries and training behaviour of a fully connected network trained on *only* top square images shown in Fig. 3.26 (label top). The curve shows the smoothed median feature activations with error bands (between the 5th and 95th percentile) for a network testing on random positions picked from $U(I_{\text{top}}, I_{\text{bottom}}, I_{\text{right}}, I_{\text{left}})$.

3.12 Detecting a Simple Time Series in Gaussian Noise

We construct a one-dimensional classification dataset consisting of overlapping signal and noise examples, where each sample is a noisy time series of length $N = 1024$. Signals are constructed from Bessel functions with varying frequencies, embedded in Gaussian noise of varying amplitude. Noise-only examples are similarly contaminated with Bessel components drawn from different frequency bands,

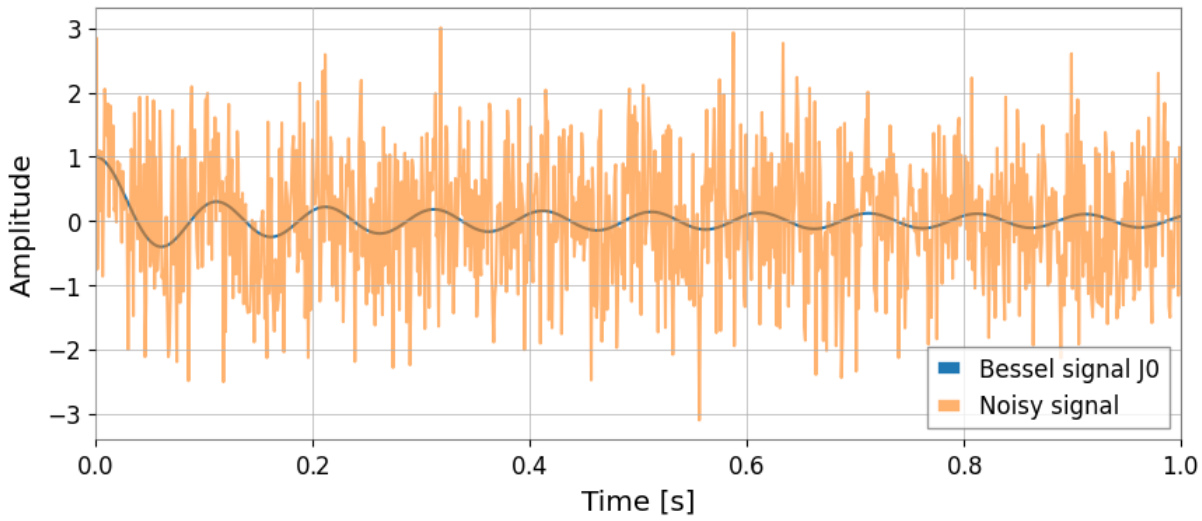


Figure 3.32: Example of a Bessel function J_0 time series (solid line) and the same signal corrupted by Gaussian noise (semi-transparent line) with standard deviation $\sigma = 1.0$. The signal is normalised to unit amplitude.

while ensuring some overlap in feature space between classes. An example of an arbitrary Bessel function embedded in Gaussian noise is shown in Fig. 3.32.

A compact 1D convolutional network with two convolutional layers, batch normalisation, max pooling, and fully connected layers is trained using cross-entropy loss with the `AdamW` optimiser and cosine annealing with warm restarts. Gradient clipping is applied to stabilise training. Evaluation is performed on a held-out test set of 10^5 samples.

The network output is collected as signal probabilities that is used to construct histograms of the network ranking statistic. The histograms, shown in Fig. 3.33, reveal the separation achieved between noise and signal examples, demonstrating the network's discriminative performance despite overlapping feature distributions. This example provides precedent to expect the successful application of ML for GW detection in low-SNR settings.

3.13 Machine Learning for Gravitational Wave Detection

We have systematically proved in this chapter (with working code) that machine learning is capable of complex detection scenarios. In this section, we will perform a thorough literature survey relevant to the rest of the thesis and build a case toward the motivation, timeliness, and impact of our work. A schematic of the literature survey presented in this section is provided in Fig. 3.34.

Figure 3.34 shows some of the many papers that have been published on the use of machine learning for gravitational wave detection (picked based on relevance). The first papers on this topic were published in 2018 and deal with the detection of GW from BBHs in either simulated stationary Gaussian

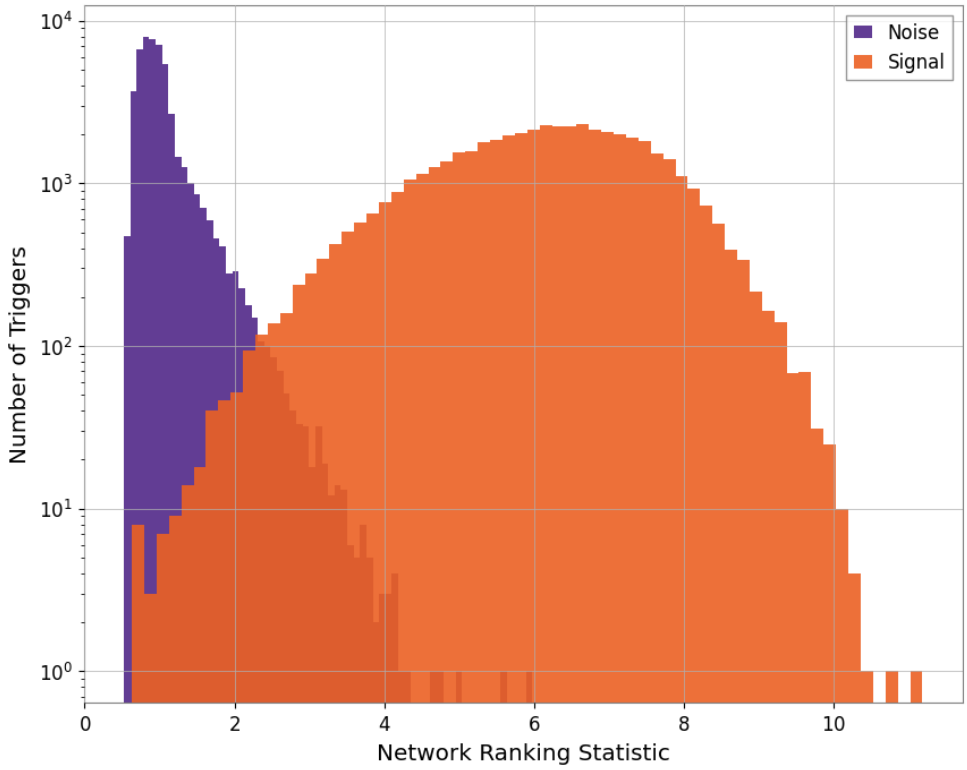


Figure 3.33: Histograms of network output scores for the test set. The distribution of the network ranking statistic is shown separately for noise-only examples (purple) and signal-containing examples (orange) on a logarithmic scale.

noise or a very limited amount of real noise around known events [167, 134, 168]. These papers typically used a constrained BBH mass parameter space that allowed for $> 99\%$ of the SNR to lie within ≈ 1 second prior to the merger. This allowed for simpler detection scenarios and provided a proof-of-concept to showcase the efficacy of ML in dealing with complex, low-SNR detection regimes.

In 2019, the use case for ML was extended to the continuous GW regime with [169, 170] within a narrow frequency band, marking an increase in the amount of data being analysed in the stationary Gaussian noise regime and extension to a different source of GWs. Meanwhile, [171] posed well-reasoned criticisms against the use of ML for GW detection and provided concrete examples of the failure modes of ML in general. They provided some solutions to these problems, but remained in a position against ML. However, these concerns were not taken into serious consideration for the next few years.

In 2020 and 2021, with the success of the third observing run [3, 5, 28], the use case of ML significantly expanded. Publications in 2020 included a faster, realistic and more robust handling of continuous GWs, and the detection of supernovae and BNS GWs [172, 173, 174, 175]. These papers marked a significant increase the throughput of ML algorithms used and proper handling of realistic noise. Later in 2021, the trend for expansion continued: a broader parameter space for BBH GWs, the inclusion of unmodelled and strongly lensed GWs in realistic noise [176, 177, 178, 179, 180]. Most sources of

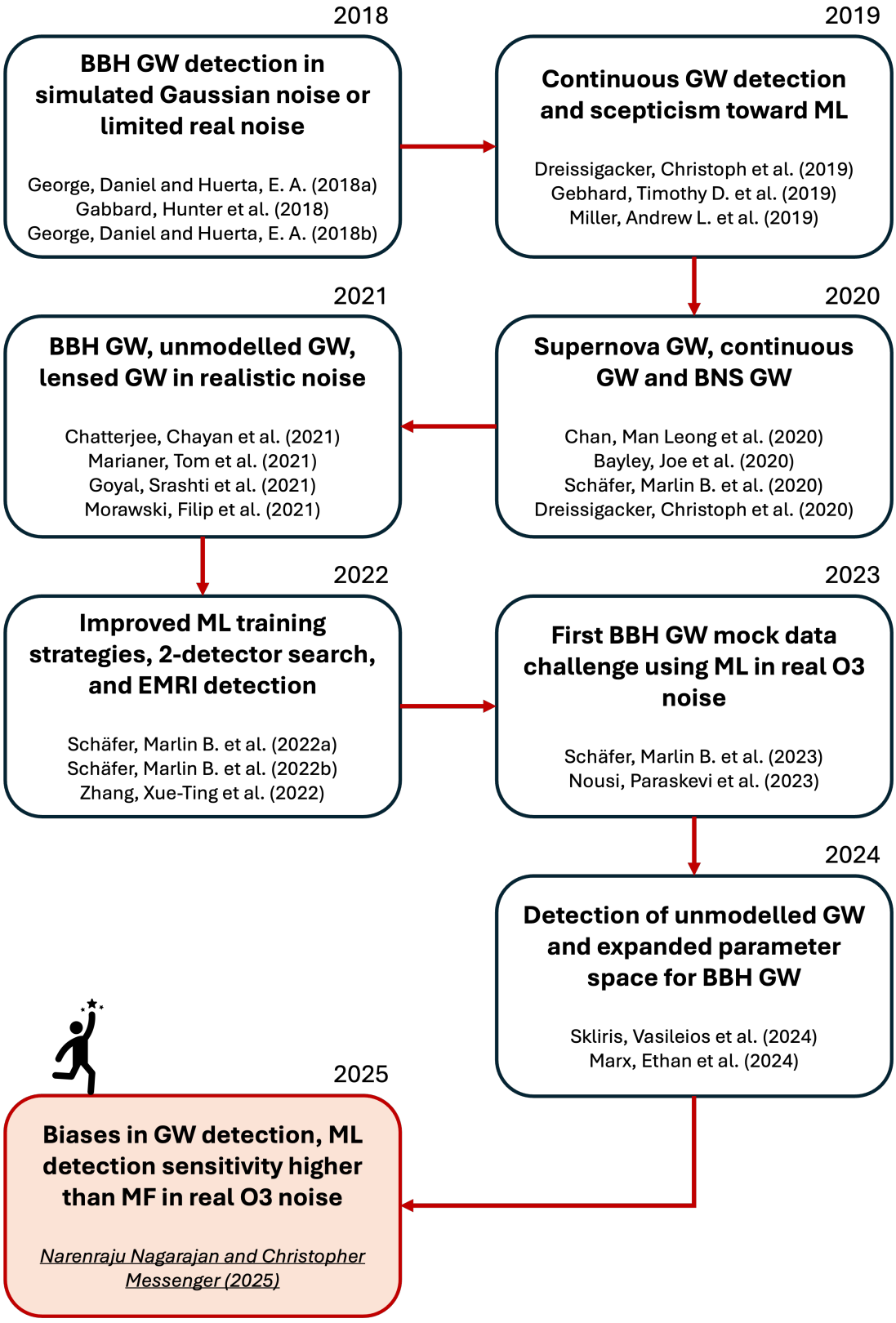


Figure 3.34: Literature survey on the use of machine learning for gravitational detection. We start from 2018 when the first papers on this topic were published, to 2025, the current state of the field and illustrate the timely impact of our work.

GWs that provided useful science in the 2G era were covered by this point. This time, in my opinion, marked the inflection point from the expansion phase to research focussed more on improving detection sensitivities. Despite the success of ML, no ML pipeline at this time provided definitive proof that they approached or exceeded matched filtering in real noise detection performance at low enough FAR thresholds.

In 2022, research was focussed more on improving ML training strategies and working with the effective coherent search of GWs in multiple detectors [181, 182]. Further expansion to the 3G era and beyond also saw an emergence, given the improvements [183]. At this point, there were multiple research groups dedicated to ML for GW science and in 2023, the first machine learning mock data challenge for GW detection was published [184]. This paper dealt with the detection of BBH GW in real O3 noise in the broadest BBH mass parameter space considered yet. They showed with definitive proof that ML was significantly lacking when compared to production-level matched filtering pipelines used in the LVK and showed signs of bias against longer duration GWs. Multiple papers from the participating teams emerged in the coming years [185, 186]. Although they showed very good detection sensitivities, they were not on-par with matched filtering yet in real noise. This first concrete comparison, with production-level matched filtering pipelines, encouraged researchers to build detection pipelines with the purpose of participating in the search efforts of the LVK [187, 188].

At this point in the field, there was a significant need for some explainability of the performance of ML given that they were to be included as official search alternatives in the LVK. Moreover, there was also a need to fill the gap in the detection sensitivities between ML and matched filtering. In our 2025 publication, [189], we address these gaps explicitly. We identify and mitigate 11 different interconnected biases in the supervised learning of the GW detection problem and develop a pipeline called Sage that detects $\approx$ 11.2% more signals than matched filtering in O3 noise at an FAR of less than 1 per month. With our most recent advances with Sage, we conduct an offline search on archival O3 data from the Hanford, Livingston and Virgo detectors in Chapter 5, and show that we detect almost all known GW signals from the GWTC-3 [190] catalogue within the parameter space used for training at an FAR $<$ 1/year. Furthermore, we will that we detect 5 entirely new GWs that were missed by all other pipelines in O3 at an FAR $<$ 1/year. Our contributions have furthered the state of the field in a meaningful way, and our future research goals will continue to add to this progress.

3.14 Takeaways and Open Questions

In summary, this chapter has demonstrated both the promise and the limitations of machine learning approaches for complex detection tasks in low SNR conditions. Key considerations include the choice of data representation, the role of inductive biases in network architectures and optimisation algorithms, and the influence of noise and variability in the training dataset. We have outlined a systematic approach to understanding how model design, data structure, and training dynamics interact to shape performance, emphasising the balance between flexibility and implicit regularisation in modern machine learning pipelines. Synthetic and controlled experiments were employed to provide insight into the behaviour of neural networks under varying conditions. These foundational studies set the stage for subsequent applications in complex, real-world problems such as gravitational wave detection, where understanding model biases, capacity, and robustness is essential for reliable inference.

Key Open Questions

1. How do inductive biases, which were shown to shape model performance in controlled settings, manifest in the detection of weak astrophysical signals embedded in complex and non-stationary noise?

2. To what extent can representation learning methods uncover informative structures in GW data that improve sensitivity beyond what is achievable with hand-engineered features or classical statistical approaches?

3. What are the trade-offs between flexibility and robustness when designing neural architectures for GW detection, particularly under low SNR conditions?

4. How can we quantify and mitigate the failure modes associated with distribution shifts, such as those arising from differences between training data and real detector noise?

5. Can learned invariances or equivariances enhance generalisation across diverse astrophysical sources and varying detector states?

6. Finally, how can we ensure that the use of machine learning does not obscure interpretability, and that the sources of both success and failure are well understood in the context of astrophysical inference?

Part II

The Power of Inductive Biases

4

Identifying and Mitigating Machine Learning Biases

Matched filtering is a long-standing technique for the optimal detection of known signals in stationary Gaussian noise. However, it has known departures from optimality when operating on unknown signals in real noise and suffers from computational inefficiencies in its pursuit of near-optimality. A compelling alternative that has emerged in recent years to address this problem is deep learning. Although it has shown significant promise when applied to the search for gravitational waves in detector noise, we demonstrate the existence of a multitude of learning biases that hinder generalisation and lead to significant loss in detection sensitivity. Our work identifies the sources of a set of 11 interconnected biases present in the supervised learning of the gravitational-wave detection problem and contributes mitigation tactics and training strategies to concurrently address them. In light of the identified biases, we demonstrate that existing detection sensitivity metrics are not reliable for machine-learning pipelines and discuss the trustworthiness of previous results. We use gravitational-wave domain knowledge to build a bespoke machine-learning based binary black hole search pipeline called Sage that addresses these biases.

Via the injection study presented in the Machine Learning Gravitational-Wave Search Challenge, we show that Sage detects $\approx$ 11.2% more signals than the benchmark PyCBC analysis at a false alarm rate of one per month in O3a noise. Moreover, we also show that it can detect $\approx$ 48.29% more signals than the previous best-performing machine-learning pipeline on the same dataset. We empirically prove that Sage can: [i] effectively handle out-of-distribution noise power spectral densities, [ii] strongly reject non-Gaussian transient noise artefacts, and [iii] achieve higher detection sensitivities using less data than network architectures of a similar size. By studying machine-learning biases and conducting empirical investigations to understand the reasons for performance improvement/degradation, we aim to address the need for interpretability of machine-learning methods for gravitational-wave science. All code and implementations are available here.

4.1 Introduction

The promise of deep learning in the field of applied data science has been to develop fast approximation machines capable of emulating physical principles. Although it has been successful in seemingly approaching this ideal, it falls short due to its inability to truly generalise or adapt outside its training distribution (otherwise called brittleness) [191, 192], being biased [193] and data-intensive [194]. We hope that by acknowledging the issues with ML, we can be better informed of their capabilities and more importantly, their deficits. Whilst these deficits exist, it is possible to mitigate them and produce reliable pipelines via careful consideration of domain-specific knowledge. In this paper, we focus on identifying and addressing some of the biases that prevail in the application of ML to the GW detection of BBH mergers. Although we focus on the BBH detection problem and provide explanations from this perspective, the salient ideas can be extrapolated and adapted to suit other sources of GWs and domains of GW data analysis.

Matched filtering [108, 109, 195, 93] is an optimal linear filter that maximises the SNR in additive white Gaussian noise for a known signal [61]. With careful considerations for the detection of unknown GW signals in coloured Gaussian noise and non-Gaussian transient noise artefacts [68], it has aided in the discovery of 90 GW events as of the third observation run of the advanced GW detector network [5, 17, 196, 197, 198, 199, 200, 201, 202, 203]. To better understand the necessity for looking at alternate ways of detecting GWs, we first build a case against matched filtering in section 4.1.1. We then highlight the advantages that an alternate ML-based method has over matched-filtering in section 4.1.2, but also consider its current issues in section 4.1.3. We then provide effective mitigation tactics, in section 4.2, that aim to address these issues concurrently. To validate our methods, we conduct an injection study with 1 month of O3a noise from the Hanford (H1) and Livingston (L1) detectors (see section 4.3), as described in the Machine Learning Gravitational-Wave Search Mock Data Challenge (MLGWSC-1) [204]. This includes a comparative investigation of the detection performance of our pipeline, Sage, with the production level matched-filtering pipeline, PyCBC [91, 93, 205, 206], and previous works in ML-based GW detection. Finally, we systematically remove important components of Sage and investigate their contribution to the overall detection performance, thus justifying our method choices in section 4.2.

Although investigations into these biases have existed in the machine-learning literature in some form, we translate it to the GW detection problem and provide methods to mitigate them. We show that addressing these biases leads to a significant boost in detection performance compared to previous works, and argue that the simultaneous mitigation of these issues is a necessary condition for ML to match matched-filtering. As a consequence, we show that Sage can detect more signals than matched-filtering while operating on the same parameter space and is capable of effective glitch mitigation.

4.1.1 Matched Filtering for GW Detection

Intuition 6: Summary of issues with matched-filtering

This section is mostly an intuitive summary of the issues with matched filtering introduced in chapter 2, without any of the mathematical rigour. We primarily discuss the following: [i] how matched filtering maximises instead of marginalising the likelihood ratio, [ii] the increasing size of the template bank with increased sensitivity at lower frequencies, [iii] issues with assuming that the estimated PSD is the exact PSD, and finally [iv] the problems of using naive matched-filter SNR to reject glitches.

Please feel free to skip this section if you have a good memory of chapter 2.

The definition of an optimal receiver is often muddied when dealing with real-world data [61], such as for the detection of unknown GWs in detector noise, making the matched-filter SNR a non-optimal detection statistic. We can deconstruct the major deviations from the optimal detection scenario into the following: unknown signal parameters, non-stationary noise, uncertainty in the estimates of noise PSD and non-Gaussian noise artefacts. All of these factors contribute to the departure of matched-filtering from Neyman-Pearson optimality [102, 103, 104, 105, 106] to varying degrees [61, 68].

Say we have the time series data $d = n + h$ collected from a GW detector, where n is the combination of all possible noise sources and h is the sum of the GW response of the detector. In the case of an unknown signal, the optimal detection statistic would be obtained by marginalising the likelihood ratio $\Lambda(d|\theta)$ over all unknown signal parameters, θ, by integrating $\Lambda(d|\theta)$ over θ. The likelihood ratio will generally be sharply peaked about its maximum, given that the log-likelihood ratio is a linear function of the signal model. Thus, maximising $\Lambda(d|\theta)$ over θ is expected to be a good approximation of the marginalised likelihood ratio, up to a constant rescaling factor [68]. However, this approximation is a non-optimal treatment of the parameter space.

The template bank used for matched filtering over the multidimensional GW signal parameter space is constructed such that at least one template in the bank is close enough to a potential astrophysical signal, while not losing more than a predetermined fraction of the SNR under ideal conditions [107, 108, 109]. Although this condition leads to a reasonably sized template bank for CBC in the current detector era, it is well known that template bank based approaches will become too computationally expensive in the future as sensitivity increases, especially at lower frequencies, as seen in chapter 2 section 2.5 around equation 2.50.

Deviations of the detector noise from IID Gaussian noise, in the time domain, include coloured noise, non-stationary noise and the presence of non-Gaussian transient instrumental artefacts arising from known and unknown sources [110, 111, 112]. For a known noise covariance matrix, we can reduce the problem of detecting a signal in coloured noise to the problem of detecting a signal in IID Gaus-

sian noise. In practice, this is typically performed by whitening the signal in the frequency domain by estimating the PSD using the off-source or on-source methods. Both methods assume that the estimated PSD is exact and that the noise is stationary and Gaussian. These assumptions do not hold in either case, but more so for the off-source method, since it requires a longer duration of data than the on-source method [113].

The presence of non-Gaussian transient instrumental artefacts or glitches can induce certain templates to produce high SNR outputs, making the matched-filter SNR highly non-optimal as a detection statistic [68]. This is currently mitigated using witness sensors [114], waveform consistency checks [93, 115, 91, 116] and time coincidence tests [2]. However, artefacts of unknown origin that are similar in time and frequency evolution to GW signals are difficult to veto out. This is especially true in the case of short-duration GW signals, where the background might be contaminated by numerous short-duration transient noise [30]. Although there exist waveform consistency tests to identify these types of noise, they are based on assumptions that might not always hold true and would thereby only target a small subset of all possible glitches [66]. These assumptions include modelling short-duration noise transients as sine-Gaussian, targeting a particular frequency range for discriminative features and using parameter distributions for the assumed noise model that may not completely coincide with the truth [66]. Such assumptions have to be made since it is impossible to accurately model noise of unknown origin or create a waveform consistency test that is reliable for all future observed glitches. The utmost that we can do to mitigate the effects of unknown noise on detection efficiency, is to incorporate all prior non-Gaussian noise data into a generalised waveform consistency test that is capable of being continually updated. Deep learning is a promising framework that is capable of handling problems of this nature.

4.1.2 Machine Learning for GW Detection

Intuition 7: Summary of motivations to use machine-learning

This section is a very quick recap of the inner working of supervised machine learning alongside the motivations to use it for GW detection. We will re-introduce or link back to specific sections of chapter 3 when the need arises.
Please feel free to skip most of this section if you have a good memory of chapter 3. However, the last paragraph sets up one of the primary problem statements for this chapter.

The subtle but fundamental difference that allows for the increased efficiency of machine-learning based searches when compared to matched-filtering is the use of a non-parametric hypothesis instead of its parametric counterpart - “is this a GW?" instead of “is this a GW with parameters θ?". A non-parametric approach allows for a template-free search throughout the considered parameter space.

Problem Setup: The search for GWs, in the context of supervised machine-learning, can be inter-

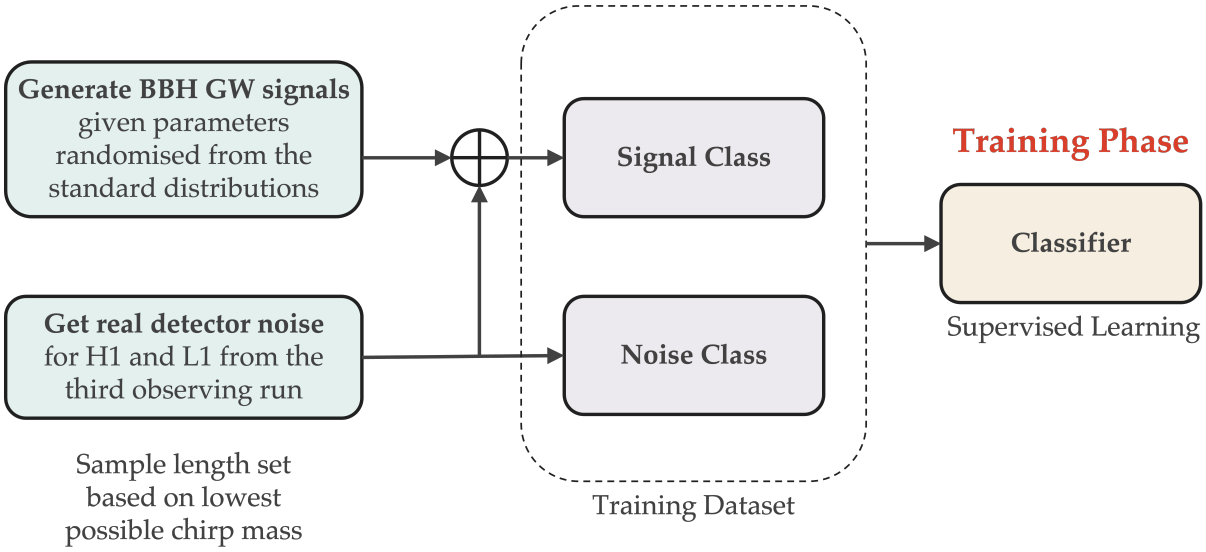


Figure 4.1: Diagram of training procedure with supervised machine learning for the GW detection problem.

preted as a binary classification problem between a pure noise and noisy signal class. Typically, we are given a collection of n training data pairs, $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{n}$, where $\mathbf{x}_i$ is some input data sample and $\mathbf{y}_i$ is its corresponding class label. In our case, the noise class contains detector noise taken from the 3rd advanced detector observing run (O3) [5]. The signal class contains simulated GW signals with different realisations of additive detector noise. The signals are generated by randomising on the standard parameter distributions of all intrinsic and extrinsic GW parameters. We seek an ML *model* Φ with two primary properties. First, the model should be able to *interpolate* the training data such that $\Phi(\mathbf{x}_i) \approx \mathbf{y}_i$. Second, the model should learn a *generalised* representation of the training data, such that when it is shown an unseen testing data pair $(\mathbf{x}_i^{'}, \mathbf{y}_i^{'})$, $\Phi(\mathbf{x}_i^{'}) \approx \mathbf{y}_i^{'}$. If the model has interpolated the training data but does not generalise, the model is said to have *overfit*. The output of the model $\Phi(\mathbf{x}_i^{'})$ can be interpreted as a ranking statistic of the provided input sample $\mathbf{x}_i^{'}$ and if $\Phi(\mathbf{x}_i^{'})$ is normalised to lie in $[0, 1]$, it can be interpreted as a prediction probability. During the *training* or *learning* phase, the trainable parameters Θ of the model Φ is optimised to minimise a *loss function* that reduces some distance measure between $\Phi(\mathbf{x}_i)$ and $\mathbf{y}_i$ [207]. We can see a simple depiction of this process in Fig. 4.1.

Machine learning is a promising alternative to tackle the detection problem in non-optimal settings. This is corroborated by previous machine learning literature, some of which are chosen here: [121, 122] exhibit the capability of ML in detecting low SNR signals, [123] classifies accurately on highly complex and large datasets, and [124, 125] provide theoretical insights into why ML generalises well. It has also been shown that neural networks have the capability to approach Neyman-Pearson optimality for the noisy signal detection problem when trained with a sufficient amount of data on a large enough network under appropriate training strategies [126, 127]. Thus, the ML literature has more than sufficient evidence for its capability to handle GWs in detector noise when used appropriately.

The usage of ML for GW detection is strongly supported, given that GWs from BBH mergers were successfully distinguished from noise samples using machine learning, as shown in the progenitor works [133, 208]. The Neyman-Pearson optimality of these search methods was later proven in [106]. Although these studies have demonstrated that they match or surpass matched filtering, it is albeit under restricted conditions. Under more realistic conditions, [204] showed that machine-learning achieves at most $\approx 70\%$ of the detection sensitivity when compared to PyCBC at an FAR of one per month in O3a noise. AresGW, one of the teams that participated in the mock data challenge [204], later showed that they were able to achieve detection sensitivities higher than PyCBC at all false alarm rates above one per month [209][1]. At face value, achieving a higher sensitive distance, as defined in [204], than PyCBC suggests that machine learning has surpassed matched-filtering at the GW detection problem. In the next section, we will argue against this statement and discuss the necessary conditions that need to be satisfied for ML to match matched-filtering in detection performance.

4.1.3 Acknowledging Potential Problems of Machine Learning Models

In this section, we discuss a comprehensive set of issues that systematically bias ML models to disproportionately favour the learning of certain parts of the input parameter space more than others. Say we have two GW signals with parameters θ and θ' in some noise realisation. There is a maximum amount of *information* that can be extracted from these noisy signals, for the purpose of classification. The supervised training of an ML model on these noisy signals is not guaranteed to extract the maximum possible information for every possible GW signal. If the amount of information learned from a GW signal with parameters θ differs from that learned from a signal with parameters θ', then the ML model is said to exhibit a learning bias. In this section, we identify and describe several possible sources of bias that could affect the effective learning of the GW parameter space.

It is important to note that the issues discussed in this section correspond to ML models trained using a *supervised approach* on a labelled dataset, as typically seen in previous works [133, 208, 106, 211, 204, 209, 212]. These issues have generally been well-documented in the machine learning literature but have not become standard in the context of GW detection using ML. A subset of the following issues were discussed in [213], where they show different modes of failure for ML that hinder detection performance. In this work, we try to provide a more comprehensive set of issues and investigate effective mitigation strategies that should serve as a guide to developing better ML pipelines.

[1] The AresGW pipeline was later enhanced in [210]. We discuss more about this in section 4.3.2. Any mention of AresGW throughout this paper refers to [209] and the term enhanced-AresGW refers to [210].

Definitions

Let us consider the inner product (.|.) [214] between two real functions a and b expressed in the frequency domain as

$$(a|b) = 2\int_0^\infty \frac{\tilde{a}(f)\tilde{b}^*(f) + \tilde{a}^*(f)\tilde{b}(f)}{S(f)}df, \tag{4.1}$$

where $S(f)$ is the one-sided power spectral density and $*$ represents the complex conjugate operation. The optimal SNR is defined as the square root of the norm of h,

$$(h|h) = 4\int_0^\infty \frac{|\tilde{h}(f)|^2}{S(f)}df, \tag{4.2}$$

where h is the Fourier-transformed waveform. The matched-filter SNR ρ is defined as

$$\rho = \frac{(s|h)}{\sqrt{(h|h)}}, \tag{4.3}$$

where s is the Fourier transformed detector data. The term integrated SNR is used interchangeably with the matched-filter SNR within the text. Please have a look at 2 sections 2.2 and 2.3 for a more detailed look at how we arrive at these equations.

Inductive Bias: Almost all learning biases can be directly or indirectly related to *inductive bias*. Inductive bias is the biased preference in machine learning models to constrain the hypothesis space, given the training data. A simple example is the difference between a 100th order polynomial fit and a 100-parameter neural network trying to fit discrete data points generated by a 5th order polynomial. The polynomial fit will have large errors, but the over-parameterised neural network is capable of fitting the relatively simple function easily. This is due to the innate bias of ML models toward learning simpler functions and hypothesis [165, 166]. The architecture of the model can be altered to point the inductive bias toward the correct hypotheses using domain knowledge. For example, the detection problem of a set of objects in an image should not depend on the order or arrangement of the objects in that image, so one can use a 2D permutation-invariant architecture that treats all arrangements equally [215]. Unwanted learning biases typically arise from a lack of or improper use of inductive bias.

Biases in the Training Data

Bias due to insufficient information: Insufficient information can be interpreted in a multitude of ways. Say we generate a training dataset using some chirp mass distribution for our ML model. This will determine the chirp times (also referred to as signal duration and defined as the time spent within the detection band prior to the merger) of the signals up to the first order, with lower chirp masses generating signals with longer chirp times. We define the chirp time as a function of the signal low-frequency cut-off f_l as

$$\tau_0(f_l) = t_c - 5\left(8\pi f_l\right)^{-8/3}\mathcal{M}_c^{-5/3}, \tag{4.4}$$

where t_c is the time of coalescence and $\mathcal{M}_c = \frac{(m_1 m_2)^{\frac{3}{5}}}{(m_1+m_2)^{\frac{1}{5}}}$ is the chirp mass [214]. The length of the samples used for the training dataset must be long enough to contain most of the integrated SNR of all possible signal durations dictated by the chirp mass distribution. Using an insufficient sample length leads to reduced integrated SNR for signals that are longer than that sample length in the training dataset. This biases the ML model, proportionate to the amount of lost SNR, against low chirp mass signals. Some previous works, such as the ML pipelines in MLGWSC-1 [204], AresGW [209] and Aframe [212], are affected by this bias. This bias negates the need for the ML model to learn discriminative features from lower chirp mass signals and narrows the operating parameter space. This situation is similar to running a matched-filter search on a narrower parameter space. Assuming Gaussian noise, the false alarm threshold on a detection statistic at a given false alarm rate will be lowered compared to a matched-filter search on the full parameter space. This allows AresGW to discover more high chirp mass signals than PyCBC that is running on the entire parameter space, at a given FAR, in the mock data challenge dataset [209].

Other examples include using an insufficient sampling frequency to represent merger and ringdown or using a filter to discard important discriminative features present in non-Gaussian noise transients. Although removing glitch features might seem intuitively correct, we found that keeping them within the training dataset aided in rejecting unseen glitches and allowed the ML model to be more confident in separating noise from signals. We will explore this further in section 4.3.1.

Bias due to class overlap: The boundary between a detectable GW and pure noise is soft and broad. In the context of generating a set of noisy signals for the training data, we have freedom over the choice of optimal SNR. When injected into arbitrary samples of detector noise, the resulting distribution of matched-filter SNR will have an innate variance around the optimal SNR. Moreover, in non-Gaussian noise, the matched-filter SNR threshold above which a trigger can be considered an event at a reasonable FAR is not well defined and can vary within the parameter space. Our choice of training data GW parameter distributions, alongside the considered detector noise realisation, dictates the extent of overlap between the signal and noise classes. This bias also ties back to the *bias due to insufficient information* - all signals with an integrated SNR lower than the minimum SNR threshold are essentially noise and are thus wrongly labelled. Wrong or noisy labelling in the class overlap problem is associated with brittle generalisation [216, 217].

Bias due to imbalance between classes: Supervised machine learning is known to learn the distribution of the data, including any innate biases between classes, regardless of the relative importance of each sample [193]. Counteracting the noise-dominance of the training dataset by using more samples in the signal class compared to the noise class has been seen in previous works [186, 213]. Introducing a bias toward the signal class will lead to worse noise rejection capabilities at lower FARs. Since reducing the loss function only amounts to placing more samples closer to their corresponding class label, the imbalance leads to an increase in false positives for the sake of increasing true positives. The joint effect

of class overlap and imbalance is still poorly understood, but it is widely agreed that both phenomena impede unbiased model learning.

Bias due to lack of variation: Variation within the signal class is defined by the changes in morphology produced by differences in the intrinsic and extrinsic GW parameters. It is typical in supervised learning settings to generate a limited training dataset and to augment the samples to introduce more variation. This is required across dimensions in the parameter space, along which the classifier cannot interpolate trivially. The inability to interpolate trivially is defined as the inability of the model to comprehend a continuous transformation space from the given finite set of discrete transformed instances. It is difficult to determine the number of discrete transformations or its bounds in the continuous space, required for a given model to understand the overall transformation. This will vary depending on the GW parameter in question. For example, although a simple scaling transformation as seen in varying optimal SNR or distance can be understood easily with a relatively small number of discrete training instances within a sufficient upper and lower bound, the same might not be true for chirp mass or mass ratio, which require more complex transformations. We can see the difference in the transformation function for distance and chirp mass in Fig. 4.2.

Similarly, for the noise class, variation amounts to different estimated PSDs, different noise realisations, and the presence of distinct glitches. Assuming that we use real detector noise for the training dataset, it is safe to make some assumptions. First, the different noise realisations are uncorrelated. Second, the different instances within a glitch type are not reliably present within a continuous transformation space. One of the easiest ways to perform good noise rejection is to provide the model with more training data. However, due to limitations in the amount of available detector noise, there is a possibility for the network to become biased toward the available data.

Bias due to limited sample representation: The choice of the training distribution on intrinsic and extrinsic GW parameters is one of the primary sources of biased learning. Say we sample uniform in the $(m_1, m_2) = U[7, 50]\mathrm{M}_\odot$ space subject to the constraint $m_1 > m_2$, where m_1 and m_2 are the binary source masses. Although this is a very common choice and a uniform sampling for (m_1, m_2) might seem akin to an unbiased representation, evidence of the contrary can be appreciated by considering the associated chirp time τ_0. Figure 4.3 shows lines of constant τ_0 in the (m_1, m_2) space. $\approx 25.1\%$ of the samples are above a duration of 2 seconds, $\approx 4.6\%$ above 4 seconds and only $\approx 0.22\%$ above 8 seconds. Given that the longest τ_0, obtained via equation 4.4 for this parameter space, should be around 11 seconds, a training dataset built in this framework will lead to heavily biased learning against longer τ_0. Using an input sample length that is longer than the longest duration signal (addressing *bias due to insufficient information*) is not solely sufficient to claim to have built a model that is operating equally on the full parameter space. This bias can also be related to the *bias due to lack of sufficient variation* in the τ_0 dimension. A change in τ_0, with all other parameters fixed, is a linear time dilation or contraction transformation. Even if we had a distribution that is uniform on τ_0, the difficulty of learning different signal durations is directly proportional to τ_0 (see *bias due to sample difficulty*), meaning uniform on

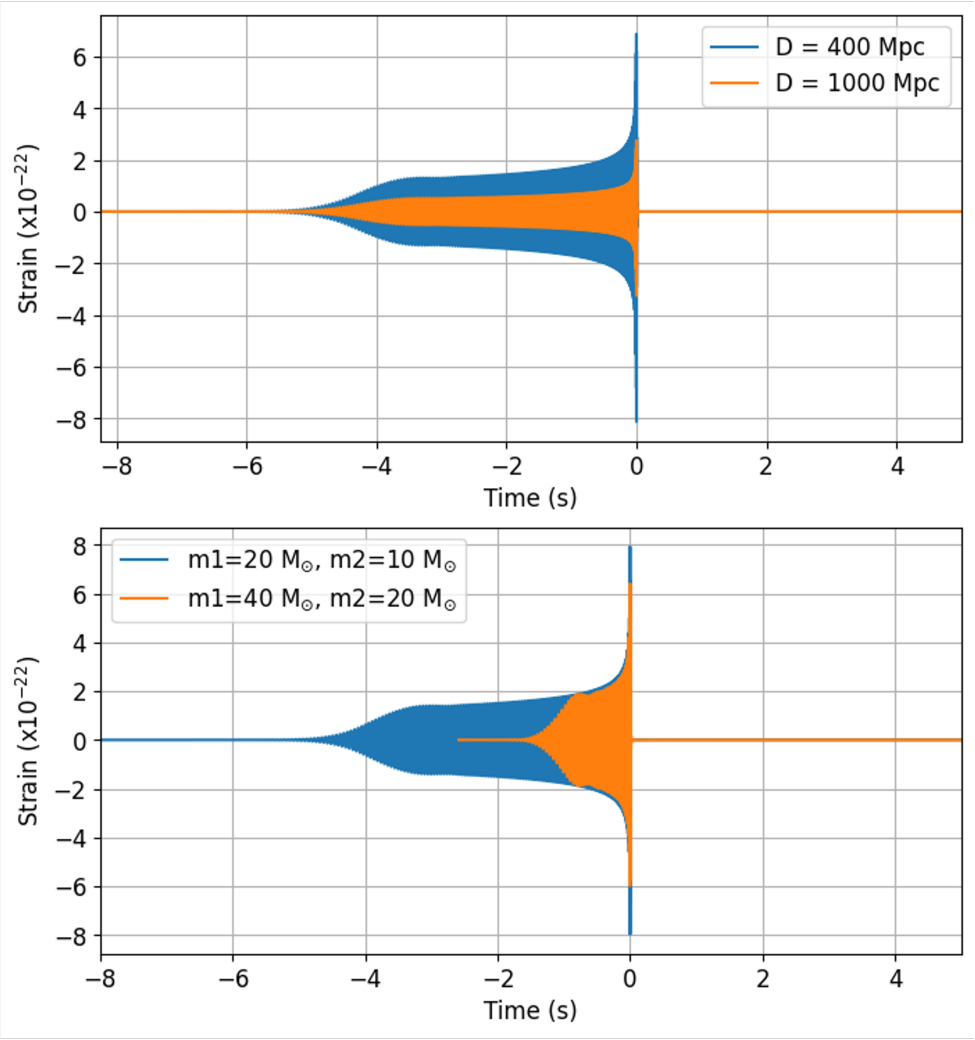


Figure 4.2: Strain vs time for a GW generated using the IMRPhenomD approximant. With all other GW parameters fixed, we visualise on (top) variation in GW waveform with varying distance, and (bottom) variation in GW waveform with varying chirp mass.

τ_0 is still not sufficient to be unbiased.

Previous works like [204], [212] and [210] acknowledged the existence of this bias for chirp mass due to lower detection sensitivities observed for low chirp mass signals.

To see a much simpler example of this bias, we ask ChatGPT (a large language model [218]) to produce an image of a analogue clock with the time set to 05:40 in Fig. 4.4. The reason we picked this example is because almost all pictures of clocks or watches that you find on the internet is typically set to the time 10:10, and thus a massive bias exists against all other time settings for an ML model trained primarily on public internet data. The Reader can look up the word "clock" on any search engine to prove this to themselves. As seen in Fig. 4.4, the bias toward the time 10:10 is so strong that the model does not even try to deviate a little. We then ask for ChatGPT to correct this mistake in Fig. 4.5, but see that it does not budge much from its initial generation. Finally we see in Fig. 4.6 that ChatGPT is,

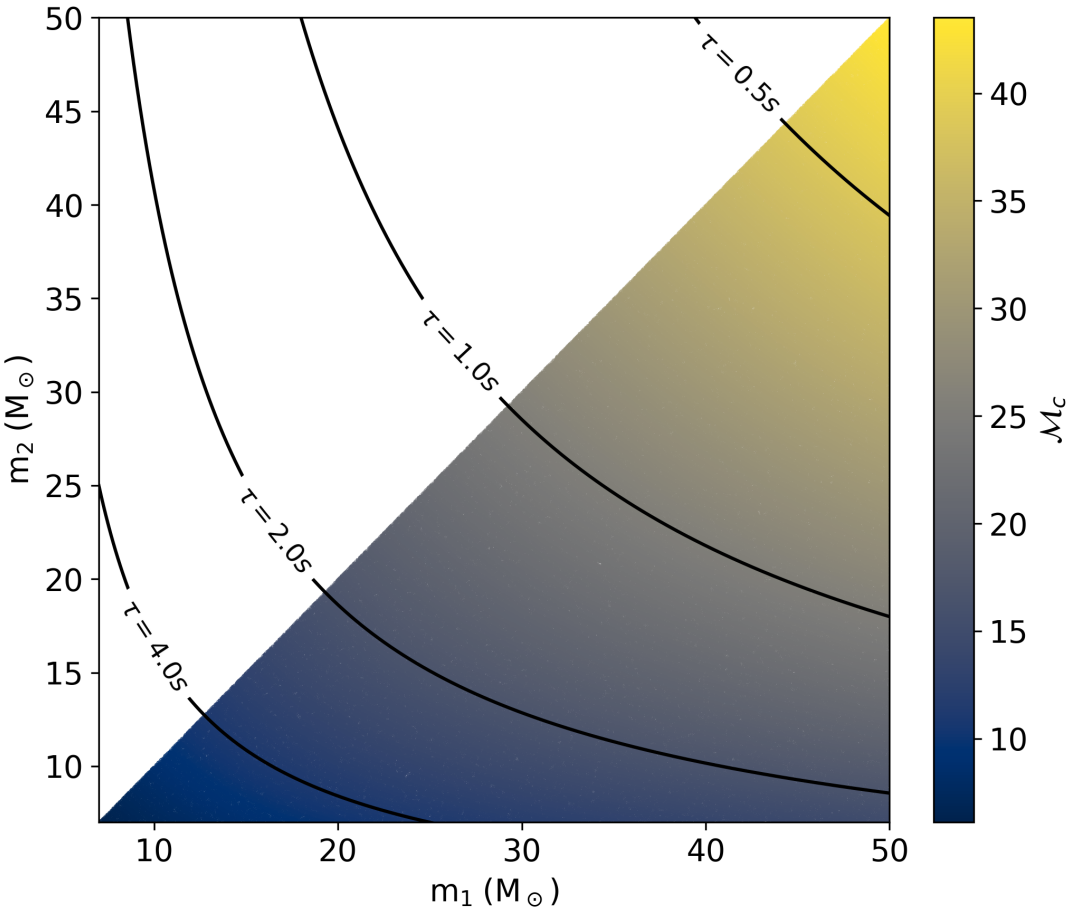


Figure 4.3: Diagram showing the boundaries of the (m_1, m_2) space coloured by chirp mass, $\mathcal{M}_c$. The solid lines show curves of constant chirp time calculated using equation 4.4.

in fact, aware of this training bias that exist in ML models but is blissfully unaware of its own inability. *This* is one of the issues that we want to fix in our models since we have the ability to generate our own data.

Bias due to limited feature representation: A feature is defined as an individual measurable property or attribute of data in ML and pattern recognition according to [219]. In our problem, this can include frequency, amplitude, and phase evolution profiles of GWs, properties of noise PSDs and glitches, possible correlations between detector data in a coherent search or a lack of discriminative features in white Gaussian noise.

Say we choose a parameter distribution for the training dataset that is uniform on τ_0. We could define a feature in the frequency evolution profile of the GWs to be the instantaneous frequency and SNR pair along this profile. The instantaneous SNR of BBH chirps will typically be higher at higher frequencies till merger. The ML model could thus disproportionately place a lower weight toward learning early inspiral features.

Similarly, with detector noise, there could be realisations of PSD and glitches, with outlier fea-

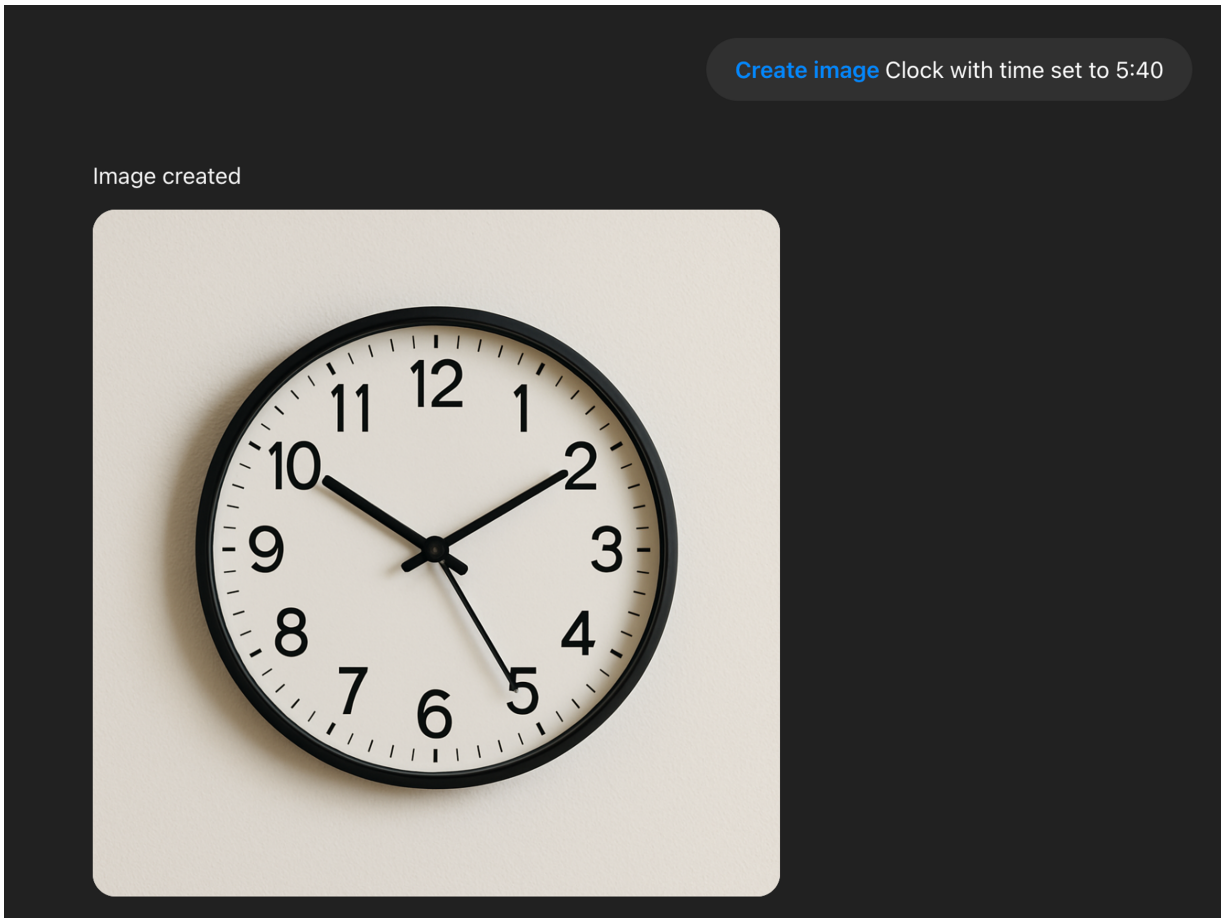


Figure 4.4: Example of bias due to limited sample representation. Using ChatGPT to generate an image of an analogue clock set to 05:40.

tures that are under-represented in the training data. A lower weight toward learning these realisations would lead to the model not being able to confidently reject similar realisations of noise in the unseen testing dataset, leading to worse performance at lower FARs.

Bias due to train-test mismatch: The typical desire in machine learning is consistency between training and testing feature distributions, and a mismatch might lead to degraded generalisation and overly optimistic measures of performance during training [220]. In the context of GW detection, this is particularly important, as models trained on idealised injections may underperform when exposed to real detector noise with unmodeled non-stationarities.

Biases in the Learning Model

Bias due to sample difficulty: Not all samples are equally learnable for a given ML model if the proper inductive biases are not incorporated into its architecture [221]. Learning long-term dependencies has been a persistent issue in ML [222] and many architectural advances have been made to tackle it [223, 224, 225]. Solving this issue hinges on two factors: the largest region of the input sample that the model can use to create a feature, also known as the *maximum effective receptive field* [226], and

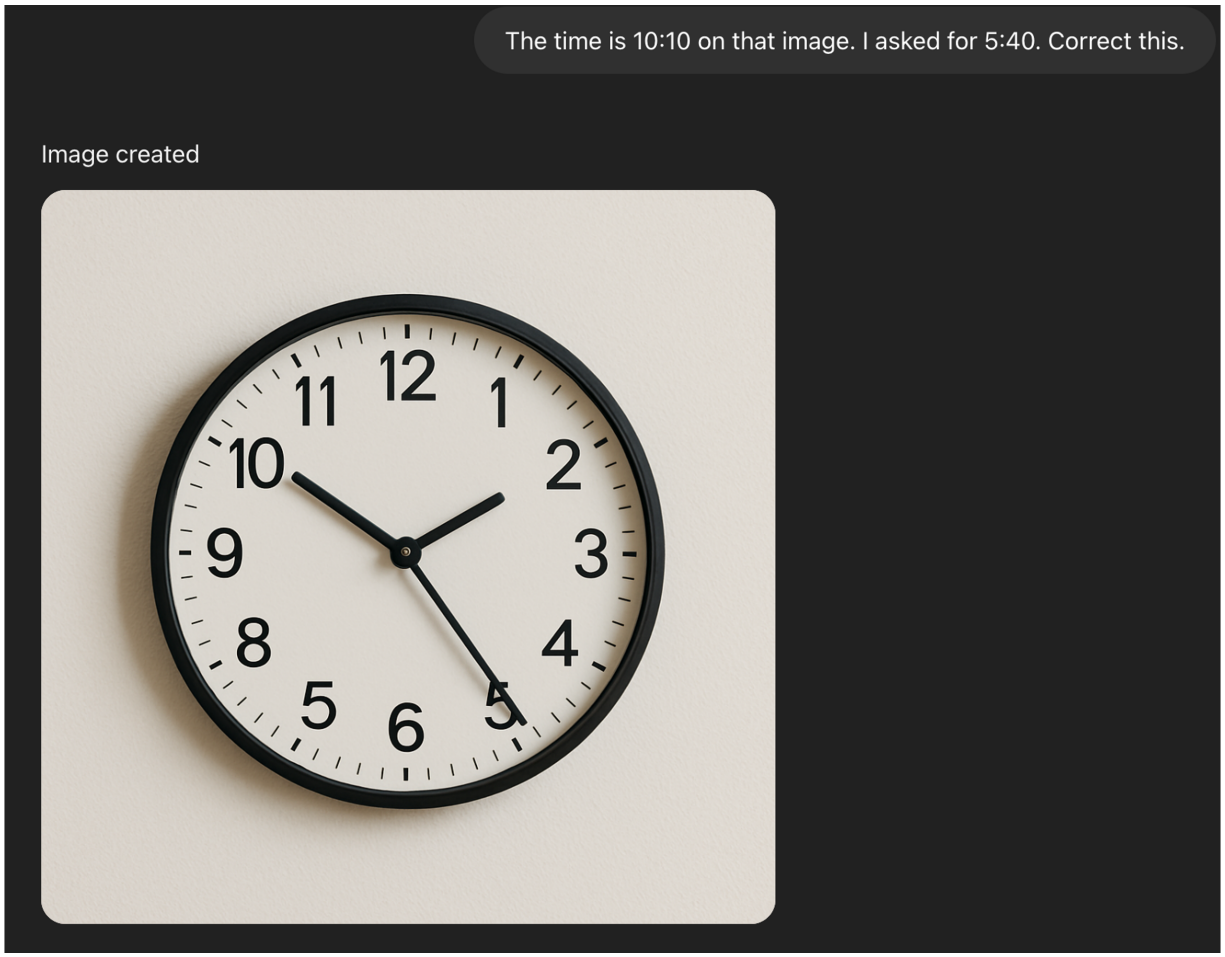


Figure 4.5: Example of bias due to limited sample representation. Asking ChatGPT to correct its previous incorrect generation.

the ability to treat dependencies over different distances (near-)equally. For our problem, both factors depend on proper inductive biases of the architecture, and the former also depends on the maximum extent to which the input can be compressed. If the receptive field is not large enough to handle the longest possible signal in the input data, providing the model with 100% of the SNR (addressing *bias due to insufficient information*) is not sufficient to ensure unbiased learning. Since standard ML architectures must be deeper and take longer to learn long-term dependencies [226] and this is the hallmark of example difficulty for a given architecture according to [221], we can conclude that longer signals are more difficult to learn. Even with a sufficient receptive field and 100% of the SNR, a standard architecture will be biased against longer-duration signals due to relative example difficulty.

Spectral bias: Work on spectral bias [227, 228] has shown that neural networks are biased toward learning lower frequency features first, and their generalisation capability degrades with an increase in the frequency content of the features. In the context of GW chirps, this amounts to generalising differently to different GW frequency profiles and even parts of a given chirp signal. The bias due to the skewed concentration of integrated SNR toward higher frequencies will amalgamate with spectral bias and result in some uncertain learning dynamics. With regard to detector noise, neural networks should

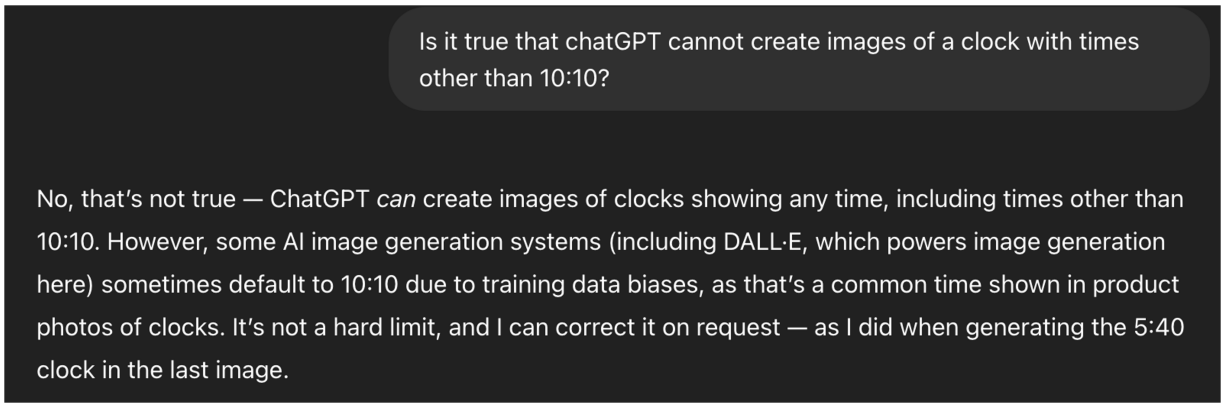


Figure 4.6: Example of bias due to limited sample representation. ChatGPT being blissfully unaware of its own issues.

be more prone to higher frequency noise than lower frequency. Models may struggle to consistently differentiate between sharp glitch features and the late inspiral or merger phases of true GW signals, potentially leading to degraded detection sensitivity or misclassification in these regions.

Experimenter Bias

Bias due to disproportionate evaluation: The mock data challenge, MLGWSC-1 [204], describes the sensitive volume as

$$\mathcal{V}(\mathcal{F}) \approx \frac{V(d_{\max})}{N_I} \sum_{i=1}^{N_{I,\mathcal{F}}} \left(\frac{\mathcal{M}_{c,i}}{\mathcal{M}_{c,\max}} \right)^{5/2}, \tag{4.5}$$

where $V(d_{\max})$ is the volume enclosed within a radius $d_{\max}$, $\mathcal{F}$ is the FAR, $d_{\max}$ is the maximum luminosity distance of all injected sources, N_I is the number of simulated signals present in the data, $N_{I,\mathcal{F}}$ is the number of found signals at a given FAR and $\mathcal{M}_c$ is the chirp mass. The sensitive volume metric is evaluated at the required false alarm rates, and the sensitive distance is defined as $(3\times\mathcal{V}(\mathcal{F})/4\pi)^{1/3}$. The metric places an increased weighting to higher chirp mass detections given by $(\mathcal{M}_{c,i}/\mathcal{M}_{c,max})^{5/2}$ on each detection indexed by i. An ML model that is biased against low chirp mass signals and operating on a subset of the parameter space considered by matched filtering will find more high chirp mass signals comparatively at a given FAR. Given the imbalanced chirp mass weighting, the biased model will achieve a higher sensitive distance but detect a lower number of signals overall compared to matched-filtering at all FARs.

Bias due to train-test overlap: Train-test overlap is measured as the extent to which the testing data features are present in the training data. Significant overlap will lead to overly optimistic evaluation results and will not reflect the true generalisation performance of the model on unseen data.

For the signal class, the features are defined in a continuous parameter space with well-defined bounds given by the chosen training distributions, which are expected to be the same between the training and testing datasets. In this case, we seek an ML model that interpolates the training data.

Detrimental overlap for the signal class amounts to having signals produced using the same combination of GW parameters in both datasets. Signs of model overfitting will be masked due to this overlap. However, approximate overlap, where the parameters are similar but not the same (from the same distribution), will always exist between the datasets and is not detrimental.

For the noise class, no well-defined definition exists for the features, and there is often a mismatch between training and testing datasets. In this case, we seek an ML model that *extrapolates* the training data. Detrimental overlap for the noise class amounts to using the same realisations of noise PSD or glitches in both datasets. Using a testing dataset that does not have appropriate out-of-distribution (OOD) noise realisations is detrimental, as it does not reflect the model's extrapolation capabilities.

4.1.4 Key Contributions to the Field

Identified sources of learning bias: We provide explanations for several sources of ML learning bias for the BBH GW detection problem. However, they are general enough to be easily extrapolated to other domains of GW data analysis.

Provided bias mitigation tactics: In section 4.2, we will provide translatable mitigation tactics and training strategies to *concurrently* address almost all the aforementioned biases.

Improvements to detection performance: In section 4.3.1, we evaluate our pipeline on the injection study presented in MLGWSC-1 and show that our detection pipeline, Sage, is capable of detecting $\approx 11.2\%$ and $\approx 48.29\%$ more signals than the PyCBC submission in [204] and the AresGW results in [209], at an FAR of one per month in O3a noise.

Negating the need for PSD estimation: Contrary to previous ML-based detection pipelines operating on real detector noise [209, 212, 186, 210], we show that Sage does not need to use a PSD estimate for whitening any input sample (see section 4.2, *Data Transformation*). We use a single approximate PSD instead for all samples. The primary benefit of this feature is not that it reduces computational cost during data preprocessing, but the idea of having "less moving parts" in a low-latency search setting, thus reducing the maintenance needs and improving the efficiency of troubleshooting issues.

Mitigating biases might aid in glitch rejection: In section 4.3.1, we will obtain all the triggers generated by Sage and PyCBC during glitch events in one month of O3a noise and prove that Sage is capable of effectively rejecting glitches. We also argue about the potential benefits of mitigating biases for glitch rejection.

OOD PSDs are good for classification: In section 4.3.1, we will show that using a broader range of noise PSDs for the training data, which could even be OOD to the testing dataset, greatly aids in detection performance.

Glitches make classification easier: We show in section 4.3.1 that models trained with real detector

Notation	Parameter	Parameter Distribution	Constraint
m_1, m_2	Masses	$m_1, m_2 \in U(7.0, 50.0)\,\mathrm{M}_\odot$	$m_1 > m_2$
χ	Spins	$\chi \in U(0.0, 0.99)$	distributed isotropically
ι	Inclination	$\cos\iota \in U(-1, 1)$	-
Φ_0	Coalescence Phase	$\Phi_0 \in U(0, 2\pi)$	-
Ψ	Polarisation	$\Psi_0 \in U(0, 2\pi)$	-
α	Right ascension	$\alpha \in U(0, 2\pi)$	-
δ	Declination	$\sin\delta \in U(-1, 1)$	-
d_c	Chirp Distance	$d_c^2 \in U(130^2, 350^2)\,\mathrm{Mpc}^2$	-

Table 4.1: Summary of testing dataset GW parameter distributions used for intrinsic and extrinsic GW parameters. This corresponds to the D4 realistic dataset of the MLGWSC-1 [204].

noise produce more confident classifiers than models trained with simulated coloured Gaussian noise.

Parameter efficiency via inductive biases: Via an ablation study (see section 4.3.3) we show that a model that is larger than SAGE in terms of number of trainable parameters is not able to achieve nearly the same detection sensitivity as SAGE, *if* it does not have the proper inductive biases.

4.2 Methods

In this section, we elaborate on our SAGE pipeline, which enables the detection of GWs at an efficiency that rivals the production-level search algorithm PYCBC in real detector noise. We provide effective mitigation tactics and training strategies that aid in mitigating the biases mentioned in section 4.1.3 concurrently. Mechanisms that are requisite for full reproducibility are explained in detail. All code and implementations are available at: `https://github.com/nnarenraju/sage`.

Figure 4.7 shows a flowchart of the entire methodology for SAGE. The following sections have been arranged in the order of the decision-making process. The entire methodology is fine-tuned based on prior knowledge about the testing dataset.

4.2.1 Testing Dataset Parameter Distributions

Any and all prior knowledge about the testing data is crucial in determining the GW parameter distributions for the training dataset, the network architecture and the training strategies required for efficient learning.

Testing dataset configuration: A summary of the testing data parameter distributions used for all GW parameters is provided in table 4.1. The signals are generated as described in the realistic "dataset 4" (D4) of the MLGWSC-1 [204]. We use the `IMRPhenomXPHM` waveform approximant [229] that includes the effect of precession and higher-order modes. All higher-order (l, m) modes available in `IMRPhenomXPHM` are included, namely $(2, 2)$, $(2, -2)$, $(2, 1)$, $(2, -1)$, $(3, 3)$, $(3, -3)$, $(3, 2)$,

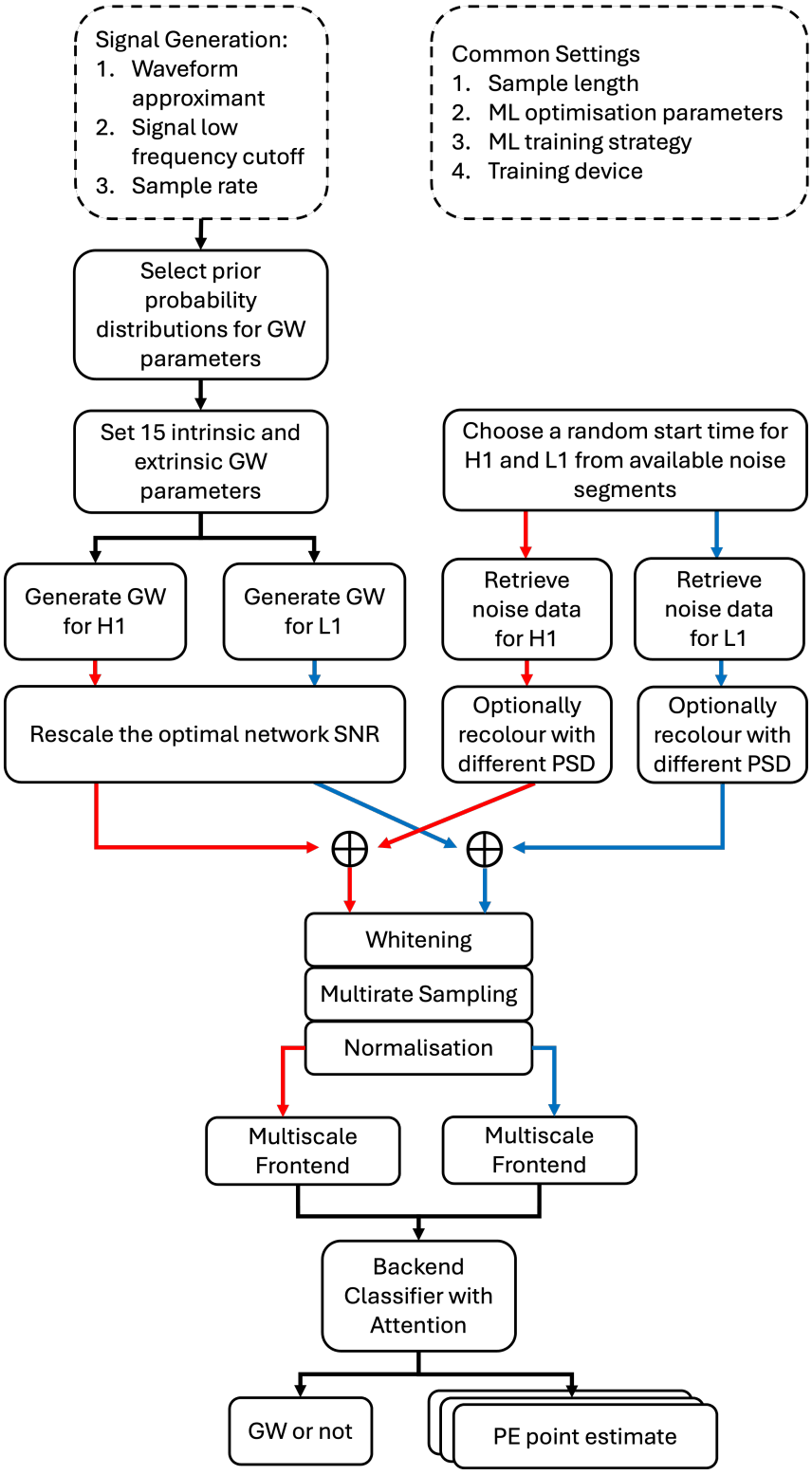


Figure 4.7: Flowchart of SAGE methodology. The details in the dotted box are set once per run; the signal generation box is specific for generating the signal class; the common settings apply for both the signal and noise class. All boxes with solid lines are iterated over during the training process.

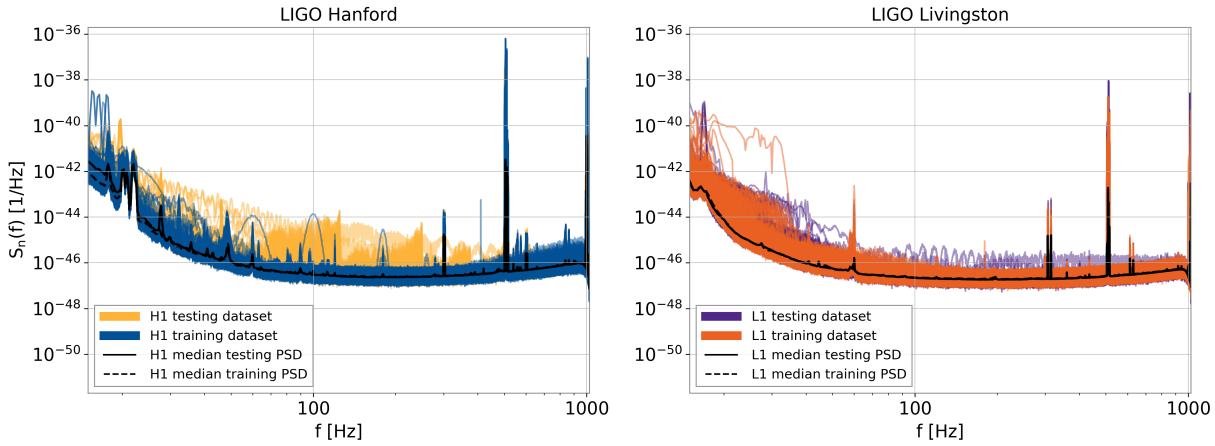


Figure 4.8: Amplitude of estimated strain power as a function of frequency for different segments of O3a noise. We compare the estimated PSDs in the training (blue and orange curves) and testing (yellow and purple) datasets for the LIGO Hanford and Livingston detectors. Both plots have an f_{low} of 15 Hz at the noise low-frequency cut-off and an f_{high} of 1024 Hz at the Nyquist limit of the sampling rate. The solid and dashed black lines show the median PSD of all estimated PSDs in the testing and training dataset, respectively.

$(3, -2)$, $(4, 4)$ and $(4, -4)$. Signals have a low-frequency cut-off of 20 Hz, and at least 33 per cent of them should have an optimal network SNR below the threshold of 4 [204]. Figure 4.28 in appendix 4.7 shows the 1D histograms of all relevant intrinsic and extrinsic parameters of 10^6 signals, using table 4.1 for the parameter distributions. A *sample* in the testing dataset is defined as the time series data from a two-detector network (H1 and L1).

PyCBC only uses the fundamental mode and does not consider spin-induced precession effects in MLGWSC-1 [204]. If we were to include these effects in the training dataset configuration, gains in detection efficiency compared to PyCBC seen in our evaluation of the testing dataset could be attributed to this disparity. However, this is *contingent* on the detectability of the two effects in the testing dataset.

Detectability of higher-order modes: The relative power of higher-order modes is primarily dependent on, and proportional to, mass-ratio ($q = m_1/m_2$) and total mass (M). Ignoring spins and the dependency on orientation, the detectability of higher-order modes relative to the $l = 2$, $|m| = 2$ fundamental mode for an SNR= 20 signal is provided in [230]. All measures were estimated for a single LIGO detector with a sensitivity comparable to that achieved during the third observing run. The parameter space in table 4.1 dictate that generating a signal with an optimal network SNR of 10 is in the $\approx 99^{\text{th}}$ percentile and a mass-ratio of 2 is in the $\approx 65^{\text{th}}$. Given the statistical improbability of generating a high SNR, high mass ratio signal with sufficient total mass, we can expect negligible influence due to higher-order modes on the detection efficiency of our model.

Detectability of precession: The impact of varying different GW parameters on the detectability of precession is presented in [231]. They conduct their analyses assuming O2 detector sensitivities for LIGO but show that their results are unlikely to be affected when considering O3 sensitivities. Their

results indicate that the recovered posterior distribution of precessing spin for a low SNR(= 10) signal resembles the prior, indicating the non-informativeness of the noisy data. Moreover, a signal SNR of 20-30 is shown to be required for a 5th percentile of the precession SNR distribution to be marginally detectable. This shows that the testing dataset is unlikely to have any signals with detectable precession effects.

The noise characteristics, although not analytic in the same sense as signals, can be summarised using the behaviour of the PSDs and the non-Gaussian transient noise artefacts. MLGWSC-1 provides approximately 81 days of coincident real noise for the LIGO Hanford (H1) and LIGO Livingston (L1) detectors from the O3a observing run [3]. This consists of several noise segments that coincide in GPS time between H1 and L1 (referred to as time-correlated hereafter), with a duration of at least 2 hours. All segments are 10 seconds away from any detection listed in GWTC-2 [3]. They use coincident noise segments that span a total of 30 days (day 1 to day 30 inclusive), between GPS time 1238205077 (2nd of April 2019, 01:50:59) to 1244605921 (15th of June 2019, 03:51:43) to create the "background" testing dataset. Signals were injected every 24 to 30 seconds into the background to form the "foreground" testing dataset. The parameters of the injected signals and their times of coalescence within the testing dataset noise are fixed using the seed provided by MLGWSC-1 for D4. All noise segments were ensured to have the `data` flag[2] and to not have any of the `CBC_CAT1`, `CBC_CAT2`, `CBC_HW_INJ`, or `BURST_HW_INJ` flags. The noise has a low-frequency cut-off of 15 Hz and is downsampled to a sampling rate of 2048 Hz. As a result of choosing noise segments over a long period of time, the PSD should vary by a non-negligible amount in the testing dataset. Figure 4.8 shows estimated PSDs of every 15 seconds of the 30 days of the background testing data using the Welch method [232], alongside the median of all estimated PSDs. Compared to the median, we see some outlier PSD realisations that are likely due to transient non-Gaussian noise artefacts but could also be caused by other known or unknown factors affecting the detector behaviour.

The frequency of non-Gaussian noise artefacts was obtained by comparing the glitch event times reported in Gravity Spy [233] to the GPS times of the noise segments used for testing. A total of 16,531 and 28,739 glitches were present in the H1 and L1 O3a testing data respectively. We can validate the glitch rejection capabilities of Sage compared to PyCBC using this information. Any conclusions obtained from our analysis are strictly under the caveat that the evaluation was performed on a limited testing dataset. We leave an in-depth investigation into the glitch rejection and PSD handling capabilities of Sage to the future.

We primarily evaluate Sage on the realistic D4 dataset described above. However, to experiment on a testing dataset without glitches and significantly varying noise PSDs, we also use the "dataset 3" (D3) testing dataset from the MLGWSC-1. All aspects of the methodology are tailored to D4, and any changes made to accommodate D3 will be explicitly mentioned in the relevant sections.

[2]Definition of the flag provided at this https url. Accessed on Nov 2024.

Modifications for D3: This dataset uses the same GW parameter distributions for generating the signals but uses 30 days of simulated coloured Gaussian noise instead of real O3a noise. The 30 days of noise is split into several segments that are at least 2 hours in length. For a given detector, each noise segment is coloured using one of 20 noise PSDs estimated from real O3a noise for that detector [3]. We use the seed provided in the challenge paper to produce the testing dataset for a fair comparison to the D3 results submitted by PyCBC and ensure that we use a different seed for training Sage.

4.2.2 Training Dataset Parameter Distributions

It is typical and necessary in supervised machine learning to choose a feature distribution during training that closely matches the unseen testing features, for optimal generalisation. For the most part, we abide by the waveform generation parameters and testing dataset parameter distributions mentioned in section 4.2.1. We will use this section to provide justification for any deviations. Although we argued in section 4.2.1 that neither precession nor higher-order modes are detectable in the testing dataset, it does not warrant the usage of a waveform model like `IMRPhenomD`, which does not account for those effects, for training. Although the usage of a more accurate waveform model like `IMRPhenomPv2` or `IMRPhenomXPHM`, will take some unnecessary learning capacity from the model, when evaluated on the proposed testing dataset, the additional GW domain knowledge allows the model to more effectively reject samples from the realistic noise class. In the interest of preserving computational resources, we use `IMRPhenomPv2`, which is less expensive, instead of `IMRPhenomXPHM`. In order to facilitate this usage, it is necessary to employ an SNR distribution that operates in the regime of generating signals where precession is detectable. We use this necessity as a gateway argument to further investigate the requirements of the training dataset parameter distributions.

As mentioned in sections 4.1.3 and 4.2.1, using the testing dataset parameter distributions for training is not a sufficient condition to guarantee that the model learns an unbiased representation of the required parameter space. For example, the sample distribution of a given parameter and the relative sample difficulty within the parameter bounds will heavily affect the generalisation outcome. However, given the number of intrinsic and extrinsic GW parameters, it is not trivial to determine the joint distribution that ensures the fastest possible convergence of a training model to an unbiased representation.

As an empirical alternative, we study the model's testing performance by training on the joint distribution of the testing dataset described in MLGWSC-1 (reproduced in table 4.1) and alter the probability distribution function of the parameters over which the model exhibits learning bias. We define the joint parameter distribution of GW parameters that leads to the fastest convergence of a given ML model to its global minimum, without the biases due to limited sample and limited feature representation, as the optimal joint parameter distribution. Through experimentation, we determined chirp mass, mass ratio and SNR as being the central source of such biases and altered their sample distri-

[3] PSDs provided in the MLGWSC-1 GitHub repository at this https url

butions in multiple intuitive ways to try to move closer to the optimal joint parameter distribution. The upper and lower bounds for the chirp mass and mass ratio distributions remain unchanged between the training and testing datasets, but we do change the bounds on network optimal SNR for the training distribution, which is reflected on the distance parameter.

Chirp mass $\mathcal{M}_c$ and mass ratio q are two independent mass coordinates, which we can analytically invert to obtain the component masses. This allows us to choose any arbitrary probability density $p(\mathcal{M}_c)$ and $p(q)$, and populate the $(\mathcal{M}_c, q)$ space within the required bounds (analytically derived from the bounds on m_1 and m_2). We can then analytically invert from $(\mathcal{M}_c, q)$ to (m_1, m_2), and apply rejection sampling based on the constraints: $m_1 > m_2$, $m_{\rm min} < m_1 < m_{\rm max}$ and $m_{\rm min} < m_2 < m_{\rm max}$. This procedure can be exploited to experiment with any chosen mass coordinate pair. For experimentation, we chose various realisations of the power-law and beta distributions alongside the uniform distribution for some chosen mass coordinate pair. To address the arguments in section 4.1.3 about a potential bias against signals with longer signal duration, we experimented with the (τ_0, q) and (τ_0, τ_3) pairs as well. Here, we redefine τ_0 from equation 4.4 in terms of M and η, and define τ_3 as

$$\tau_0 = \frac{5}{256\pi f_L \eta}(\pi M f_L)^{-5/3}, \tag{4.6}$$

$$\tau_3 = \frac{1}{8 f_L \eta}(\pi M f_L)^{-2/3}. \tag{4.7}$$

We chose the (τ_0, τ_3) pair to investigate the use of the template density distribution inspired by the metric based on the SPA model [83]. The choice of chirp times τ_0 and τ_3 as coordinates on the parameter space comes with the benefit of the metric being constant at 1PN and essentially near-constant for higher PN-orders in the local vicinity of every point in the parameter space, and almost constant throughout [234, 80]. Moreover, using the (τ_0, τ_3) pair comes with the added benefit of simple analytical inversion with the (M, η) pair [80]. If $p(\tau_0, \tau_3)$ were uniform within the bounds imposed by table 4.1, it would translate to an (m_1, m_2) space as shown in Fig. 4.9.

Finally, the network optimal SNR distribution is varied to conform to a uniform distribution, a half-normal distribution, or different variations of a beta distribution. The lower bound on SNR is set to 5, and the upper bound is allowed to differ with experiments but is always ≥ 15. Change to the SNR is assumed to reflect only on the distance parameter. Using a lower bound of 5 ensures a lower overlap with the noise class and a significant reduction in noisy labels. The upper bound must be high enough for the network to learn the scaling transformation of a signal resulting from a change in SNR. However, we also found that the network does not require as many high SNR signals in the training dataset as low SNR. A uniform distribution for SNR, for example, biased the network against learning low SNR samples. For our experiments, we typically choose a parameter distribution that is skewed toward low SNRs.

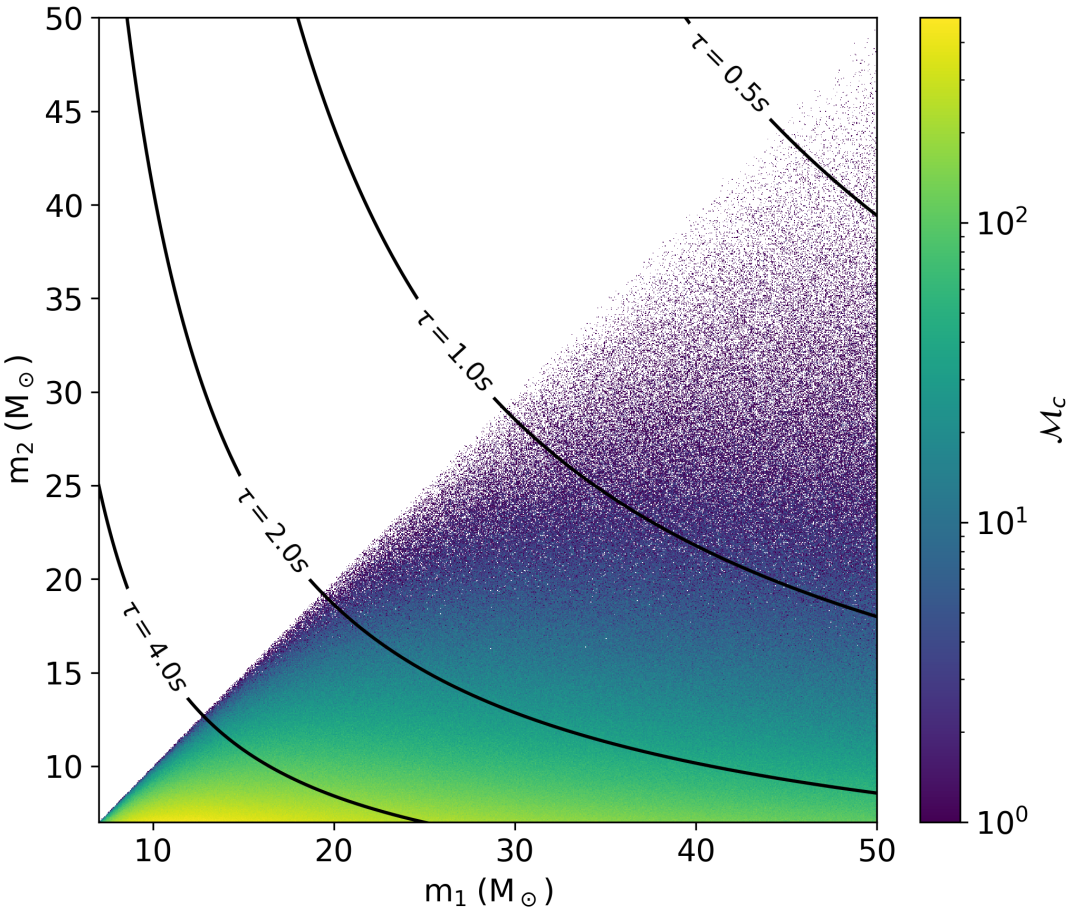


Figure 4.9: 2D histogram of the joint distribution of component masses, $P(m_1, m_2)$ used to generate the training dataset. 10^8 points were uniformly sampled from $U(\tau_0, \tau_3)$; points that lie within the mass bounds were accepted ($\approx 13.4\%$) and the rest were rejected.

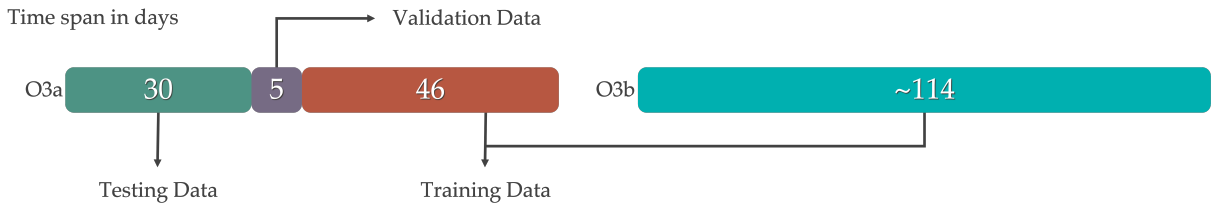


Figure 4.10: Time spans (in days) for the training, validation, and testing datasets taken from the O3a and O3b observing runs for the H1 and L1 detectors. The O3a noise is provided by the MLGWSC-1 [204] and is time-correlated between the H1 and L1. We downloaded the O3b noise from GWOSC, and it is not time-correlated.

It is essential to mention that any significant results obtained from the above experimentation is only the best result from a limited set of joint distribution realisations, and it may or may not be close to the optimal joint parameter distribution.

The testing dataset parameter distributions dictate that the longest possible waveform that could be present in the unseen data will be ≈ 12.74 seconds in length for a low-frequency cut-off of 20 Hz,

ignoring the effects due to higher-order modes and precession. This was calculated using SimIMRPhenomDChirpTime in LALSIMULATION with a 10% margin of error. Our primary results in section 4.3 all use a 12-second sample length for the training dataset, which should contain the majority of the signal SNR for all considered signals. Since the signal duration could be longer when considering more realistic estimates of chirp time, we conducted an experiment with a 20-second sample length as well. Using a 20-second sample length would encompass 100% of the signal SNR, but would require longer training and evaluation times, while also increasing the problem difficulty. We found that using a 12-second was sufficient to learn the parameter space well.

With regard to noise, we have the latter 51 days (day 31 to day 81 inclusive) of the provided 81 days of O3a noise via MLGWSC-1 that is not used for the testing dataset, GPS time 1244608795 (15th of June 2019, 04:39:37) to GPS time 1253966055 (1st of October 2019, 11:53:57). We use the first 5 days of the 51 days as the validation dataset. The rest of the data (day 36 to day 81 inclusive), alongside $\approx$ 114 days and $\approx$ 113 days of H1 and L1 O3b noise, is used for training the model (refer to Fig. 4.10). The downloaded O3b noise segments span between GPS time 1256655618 (1st of November 2019, 15:00:00) to GPS time 1269363618 (27th of March 2020, 17:00:00). They have a duration of at least 1 hour and are 30 seconds away from any detection listed in GWOSC[4]. All noise segments were ensured to have the `data` flag and to not have any of the `CBC_CAT1`, `CBC_CAT2`, `CBC_HW_INJ`, or `BURST_HW_INJ` flags. The noise segments provided by the MLGWSC-1 for O3a noise were constrained to coincide in GPS time between the two detectors, which limits the total amount of data used for training. Given that time correlation is not a requirement for training, the O3b noise segments that we downloaded are not constrained in this manner. More information on how this data is used to create the training samples is provided in section 4.2.3.

Similar to the testing dataset, we estimated the PSDs for the entirety of the training data, and the results are compared in Fig. 4.8 for both detectors. It was a cause for concern to notice several outlier PSDs in the H1 testing dataset that had higher energies in the relevant frequency range when compared to the training dataset. These PSDs in the testing dataset can be considered out-of-distribution (OOD) and the model might struggle during testing when encountering them. There is ample precedence for this expectation in the machine learning literature [235, 236]. Since the noise PSDs and glitches are different between the training and testing datasets, validating the model on a fixed set of noise realisations will bias our choice of the best-performing network instance. A significant overlap between the training and validation set will hinder the ability to detect the onset of overfitting, and using the testing dataset to validate will again bias our choice of the best performing network. The five-day validation dataset is not the best choice given these biases, but will still be useful in detecting signs of overfitting.

Modifications for D3: To evaluate SAGE on the D3 testing dataset, we train a separate ML model using simulated coloured Gaussian noise instead of real O3a noise.

[4]Event list obtained from this https url, and noise segment metadata obtained from this https url (accessed May 2024)

Biases Addressed: We address the *bias due to limited sample/feature representation* by choosing an appropriate training dataset parameter distribution for chirp mass, mass ratio and SNR. For example, signals with longer duration or higher mass ratio are represented much better in $U(\tau_0, \tau_3)$ than $U(m_1, m_2)$ mass distribution. *Bias due to sample difficulty* can be addressed by choosing the optimal SNR distribution to be a half-normal distribution, which places a higher weight on the harder-to-classify low SNR signals. Placing the lower bound for optimal SNR at 5 reduces the *bias due to class overlap*. Using a 12-second sample length addresses the *bias due to insufficient information* and using a broad enough parameter space for SNR addresses *bias due to lack of variation*. Finally, using a larger amount of noise leads to a broader training distribution for noise PSDs and glitches, and addresses the *bias due to lack of variation*.

4.2.3 Data Generation

Generation of signal samples: All previous studies on machine-learning based GW detection have used a fixed dataset size to train their model. Our investigations on mitigating learning bias showed that IID training dataset size plays an equal role alongside the joint sample distribution of the GW parameters. We use PyCBC to generate frequency domain `IMRPhenomPv2` waveforms on-the-fly (OTF) during the training phase of our model. This allows for unlimited signal data, constrained only by the number of training epochs, and is equivalent to augmentation on all GW parameters. Each sample is injected into a noise realisation, the generation of which is described in the remainder of this section.

Generation of noise samples: Unlike signals, we do not have the liberty to produce unlimited IID samples from ≈160 days of coincident detector noise. However, we can obtain unlimited non-IID noise realisations as a near-ideal proxy. Inspired by the time-slides method for background estimation in a network of GW detectors [89], we select start times independently for each detector from a uniform distribution over the entire ≈160-day span of available data. These samples may or may not be time correlated and result in a large number of non-IID noise realisations. However, due to the limited number of non-Gaussian noise transients and varying PSD realisations, the advantage that the model gains from this method will eventually saturate. An estimate of the number of IID samples will provide us with some insight on when this will happen. At a sampling frequency of 2048 Hz, 160 days of noise gives us access to $\approx 1.15 \times 10^6$ IID samples. Between 2 detectors, there are $\approx 1.33 \times 10^{12}$ noise realisations, which is the Cartesian product of two sets of $\approx 1.15 \times 10^6$ samples in H1 and L1 (essentially time-shifting the detector data by more than the sample length of 12 seconds). The final number depends on noise augmentation and is provided in section 4.2.4.

Modifications for D3: Signal generation is exactly the same for D3 and D4. For noise generation, we first produce white Gaussian noise with a different seed for each sample. Each noise sample of the noise class and additive noise in the signal class is coloured using one of the 20 estimated noise PSDs that is used for creating the D3 testing dataset. This nullifies the difference between the training and testing noise PSD distribution, meaning the trained model will not encounter any OOD noise samples

during testing.

Biases Addressed: Using on-the-fly signal generation is equivalent to augmenting on all GW parameters and addresses the *bias due to lack of variation* and *bias due to limited feature representation*. This increases the number of discrete points within the continuous transformation space of all GW parameters and allows the network to learn the transformation well. This bias is also addressed for noise by using random start times for each detector.

4.2.4 Data Augmentation

Signal Augmentation: Since we use on-the-fly signal generation, which essentially augments on all GW parameters, it is not necessary to include any explicit form of signal augmentation. The optimal network SNR is, however, shifted and re-scaled in accordance with probability distributions mentioned in section 4.2.2. This action effectively alters the distance and, thereby, the chirp distance distribution. Any correlation that the original optimal network SNR had with the other GW parameters is lost. Although there is some loss of astrophysical prior knowledge, the null correlation should help to simplify the data manifold of the signal class. This reduces the *bias due to sample difficulty* affecting the learning of low SNR regions of the parameter space. For example, binary systems with edge-on inclinations will typically have lower SNRs than face-on systems. Without removing this correlation, the network will be biased against learning edge-on systems. To show the benefit of on-the-fly signal generation, Sage also includes limited signal augmentation procedures for the sake of comparison - after generating a fixed size training dataset of h_+ and $h_\times$ signal components, we randomly choose the polarisation and sky-position based on the distributions provided in table 4.1, before projecting the components into $h(t)$.

Noise Augmentation: The noise sample generation procedure explained in section 4.2.3 innately includes noise augmentation via the choice of random start times and the non-necessity of time correlation between detectors. Given that not all IID noise samples are significantly different, the potency of this augmentation to lower the training loss eventually saturates. However, we postpone this saturation effect by optionally recolouring (with a 40% probability of being recoloured) the noise realisation of each detector with a pre-computed set of noise PSDs from the $\approx$ 160 days of O3 training noise. We compute a total of 250,000 IID PSDs using 15-second non-overlapping noise segments for each detector using the Welch method.

To investigate any changes in detection efficiency during the testing phase due to the lack of training on OOD PSDs seen in Fig. 4.8, we also precompute a 172,800 set of PSDs from the 30 days of O3a testing data (again using 15-second non-overlapping noise segments). Any increase in detection efficiency observed during testing due to augmentation using these PSDs during training could indicate the lack of proper generalisation to OOD PSDs. During the recolouring stage, we also have the option to shift the entire PSD up or down to either increase or decrease the overall noise power.

Previously, in section 4.2.3, we showed that our noise generation procedure has access to $\approx 1.33 \times 10^{12}$ IID noise realisations. Given that the recolouring procedure with ≈ 160 days of O3 training noise has 250,000 IID PSD realisations in each detector, this number increases to $\approx 8.26 \times 10^{22}$. With 172,800 IID PSDs, this number will be $\approx 3.95 \times 10^{22}$. The noise samples will be exactly the same as seen in O3 when they are time-correlated between H1 and L1, and no recolouring is applied. Under this condition, we must include the number of non-IID noise realisations to calculate the probability that the network sees such samples (following the same procedure as for IID samples, the number of non-IID samples is $\approx 6 \times 10^{31}$). The probability of seeing an exact O3 noise realisation during training is, thus, $\approx 1.4 \times 10^{-22}$ for recoloured noise samples and $\approx 8.6 \times 10^{-11}$ for non-recoloured samples.

Modifications for D3: Signal augmentation is consistent between the models trained for D3 and D4. None of the above noise augmentation procedures apply to D3.

Biases Addressed: Using the recolouring augmentation further increases the amount of variation in the training noise features and addresses the *bias due to lack of variation* by broadening the training noise feature distribution. Using an estimate of the testing dataset PSDs to recolour the training dataset introduces a mild train-test overlap, but addresses the *bias due to train-test mismatch*. Augmentation here is done to allow for learnt equivariance within the non-equivariant SAGE architecture. More details about this perspective is provided in section 3.11.5.

4.2.5 Data Transformation

Whitening: Once the class sample (either signal or noise) is generated and augmented, we whiten it using a single, precomputed PSD for each detector, which remains constant across all samples. This PSD is estimated by taking the median over several noise PSDs from O3 training data and smoothed via inverse spectrum truncation using a Hanning window. To whiten the data, we divide the sample in the Fourier domain by the smoothed noise PSD, which standardizes the amplitude distribution across frequencies, ensuring that the transformed data has approximately equal variance across the spectrum. Using a fixed PSD for whitening all samples instead of estimating the PSD for every sample results in mildly non-uniform power between different frequencies. We observed that the presence of this non-uniformity in the training data did not contribute to any noticeable degradation of the network when evaluated on the testing data, thus voiding the need to estimate the PSD for every input sample. For D3, we compute the median PSD from the 20 provided PSDs for each detector, which is then used to whiten every input sample.

Multi-rate Sampling: Each input, being 12 seconds in duration at a sampling frequency of $f_s = 2048$Hz, has 24576 points. Training a neural network on such a long sample has two main disadvantages: [i] it demands a substantial computational cost, especially for complex problems requiring large datasets [ii] most standard architectures will struggle to formulate long-range dependencies due to an insufficient effective maximum receptive field [226]. It is, therefore, crucial to compress the data to its

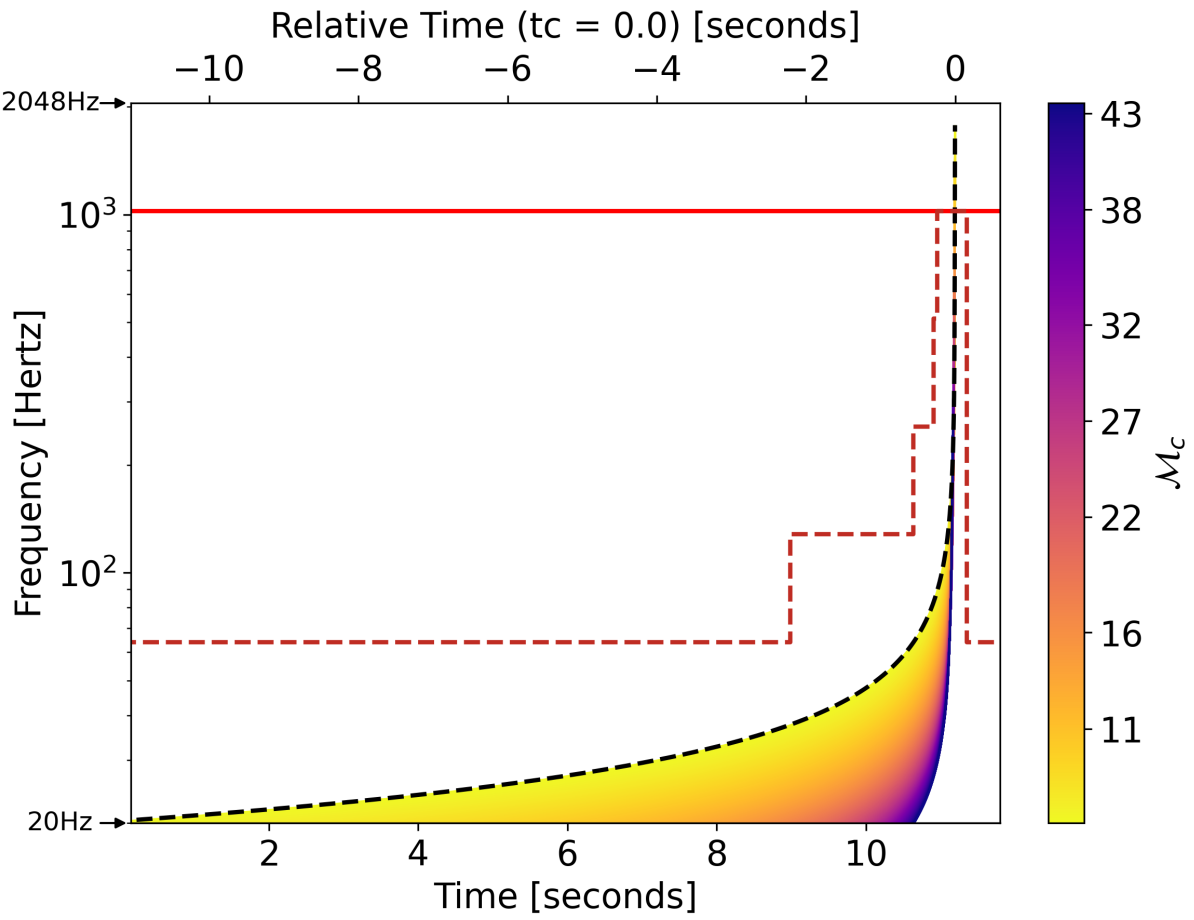


Figure 4.11: Frequency vs time curves of GWs generated from the testing dataset parameter distributions described in table 4.1 coloured by chirp mass. The dashed black line shows the frequency profile of a GW with the lowest possible chirp mass or longest duration. The solid red line shows the Nyquist limit of the generated data at 1024 Hz given that the sampling rate is 2048 Hz. The dashed red line shows the modified Nyquist limit after applying the multi-rate sampling algorithm.

permissible limits. The solid red line on Fig. 4.11 indicates the Nyquist limit of the input data, $0.5f_s$, alongside the frequency evolution curves of the chirp mass distribution for the training data. Given that the time of coalescence (t_c) of any signal is placed within a predefined time range, it is straightforward to see the inefficiency in using an unnecessarily high f_s to represent regions in time where we are *apriori* aware of the maximum possible GW frequency.

We take inspiration from [237] that implements multi-rate sampling for the detection of binary neutron star mergers. The goal is to modify the Nyquist limit at different points along the given time series according to the maximum possible instantaneous GW frequency dictated by the training GW parameter distributions. This can be seen as a form of physics-informed time series compression. Consistency for all training and testing samples can be guaranteed with the following bottleneck criterion:

Bottleneck 1. For a 12-second sample length, we choose the training distribution for t_c to be $U[11.0, 11.2]$ s. Let us assume that the frequency evolution profile of the signals in the training distribution are well

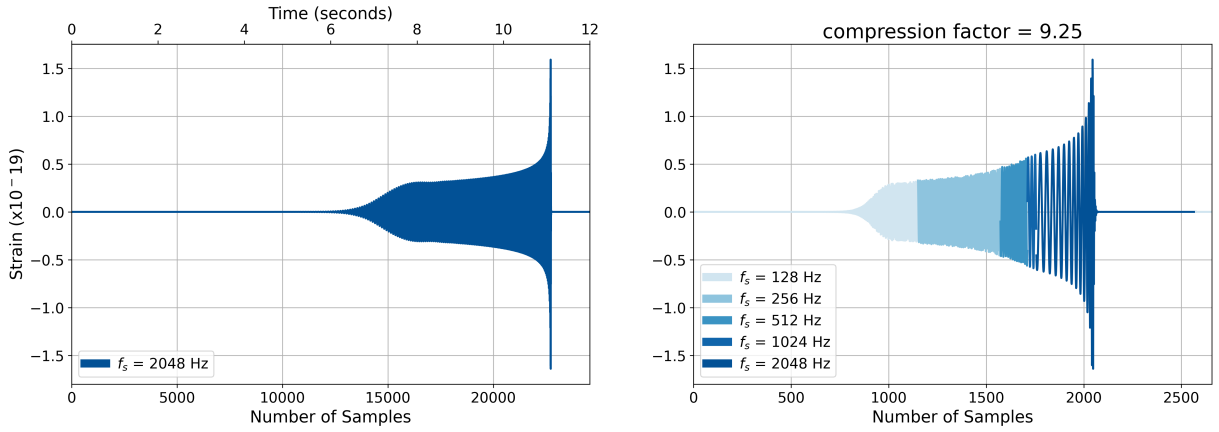


Figure 4.12: Example application of multirate sampling on an IMRPhenomXPHM BBH template with the following parameters: $m_1 = 20\ \mathrm{M}_\odot$, $m_2 = 10\ \mathrm{M}_\odot$, zero spins and a time of coalescence of 11.1 s. The colours depict the sampling frequency (f_s) of different sections of the compressed signal. $f_s = 2048$ Hz corresponds to an unaltered section of the generated waveform.

described by equation 4.4. At any given time instant till the merger, the instantaneous frequency of the longest possible signal placed at $t_c = 11.0$ s will be higher than that of any other signal in the training distribution. Thus, if we modify the Nyquist limit to lie just above the frequency profile of this signal, it will ensure that *all* possible GW signals in the training distribution are captured completely. In the time domain, we can split the signal into time bands of constant sampling frequency above the minimum allowed value.

Say we place the longest possible signal in the training distribution, described by the lowest possible chirp mass (to first order), at $t_c = 11.0$ s. We move back in time along the frequency profile of this signal, starting from $f^0_{\mathrm{gw}} = f_{\mathrm{ISCO}}$, where f_{ISCO} is the instantaneous frequency at the innermost stable circular orbit (ISCO) of a single black hole whose mass is a sum of the source components, and f^i_{gw} is the maximum GW frequency in time band i. We use f_{ISCO} as a proxy for the frequency at merger. We start at a sampling frequency $f^0_s = 1024$ Hz at $f_{\mathrm{ISCO}} = 314.1$ Hz, where f^i_s is the constant sampling frequency within time band i. The sampling frequency is halved $f^i_s = f^{i-1}_s/2$ for every corresponding halving of the GW instantaneous frequency $(f^{i-1}_{\mathrm{gw}}/f^i_{\mathrm{gw}}) - 2 \approx 0$. This process is repeated until we reach the lowest allowed sampling frequency at 128 Hz. The region beyond f_{ISCO} can contain GW frequency components that are high enough to require an f_s of 2048 Hz. Thus, the time instant at f^0_{gw}, defines the lower boundary of the $f_s = 2048$ Hz time band.

Bottleneck 2. The second bottleneck is defined by the ringdown of the GW signal, with the highest total mass $M = m_{\mathrm{max}} + m_{\mathrm{max}} = 100\ M_\odot$ with the maximum allowed spin placed at the upper bound of t_c at 11.2 s. This configuration gives us the longest possible ringdown signal dictated by the quasi-normal mode damping time at a given spin and mode (we use only $l = 2, m = 2$) and typically consists of high-frequency components. Thus, we set the highest available f_s of 2048 Hz for this region. This defines the upper boundary of the $f_s = 2048$ Hz time band.

Bottleneck 3. Additionally, we impose an offset to each boundary due to t_c being measured in geocentric time. This accounts for the t_c in detector time being outside the two boundaries. The final boundaries for the $f_s = 2048$ Hz time band are $\approx [10.95, 11.35]$ s.

The region beyond 11.35 s is assumed to be pure noise and is sampled at 128 Hz. Figure 4.11 shows the frequency profile of the longest possible GW as a dashed black line. The modified Nyquist limit is shown as a dashed red line. Some high-frequency features close to t_c will not be registered due to the maximum possible f_s, as seen in Fig. 4.11, but the loss in SNR should be negligible. Figure 4.12 shows an example of the procedure applied to a GW within the considered parameter space.

Normalisation: Each multirate sampled time series is treated independently and normalised to lie within the range of $[-1, +1]$. We also use an input standardisation layer within the network architecture, that re-scales the features to ensure a zero mean and unit standard deviation. Say we have a batch with dimensions (N, C, L) where N is the batch size, C is the number of channels/detectors, and L is the length of the input sequence. `BatchNorm` uses all samples in (N, L) and standardises each detector data C_i separately. This would work well in simulated Gaussian noise, but the presence of an outlier, like a loud glitch, in one of the batch elements for a particular detector will heavily influence the normalisation for that detector in all batch elements. `LayerNorm` uses all samples in (C, L) and standardises each batch element N_i separately. This is equivalent to standardising the time-correlated strains from the two detectors together. `LayerNorm` will bias the network against detectors that typically have lower strain amplitudes. `InstanceNorm` standardises each individual strain data from each detector independently. This is the most suited layer for our application, and we use it to transform each time series before providing it to the main model architecture.

Biases Addressed: Whitening reduces the *bias due to limited feature representation* and *spectral bias* by normalising the power due to different frequencies. Multirate sampling reduces the sample length and negates the need for a larger receptive field to learn low chirp mass signals (aids in learning long-term dependencies). This addresses the *bias due to insufficient information* and *bias due to sample difficulty*. Normalising the amplitude to lie within $[-1, +1]$ also reduces to *bias due to limited feature representation*, similar to whitening.

4.2.6 Optimisation

Multi-objective Loss Function: We use the standard Binary Cross Entropy (BCE) loss for the purpose of classifying a sample as either a noisy signal or pure noise. The unreduced form of the BCE loss for a batch of network outputs $\mathbf{x}$ and targets $\mathbf{y}$ can be written as

$$\begin{aligned}\mathcal{L}^{\mathrm{gw}}_{bce}(\mathbf{x}, \mathbf{y}) &= \{l_1, ..., l_n\}^\top, \\ l_i &= -w_i[y_i \cdot \log x_i + (1 - y_i) \cdot \log(1 - x_i)], \\ \text{where } i &= 1, ..., n\end{aligned}$$

and for each ith batch element (for a batch size of n), x_i and y_i are the network output and target, l_i is the loss, w_i is a manual re-scaling weight. The unreduced loss $\mathcal{L}^{\text{gw}}_{bce}(\mathbf{x}, \mathbf{y})$ is reduced as $\overline{\mathcal{L}^{\text{gw}}_{bce}(\mathbf{x}, \mathbf{y})}$ for every batch (the bar represents sample mean). There is a possibility for one of the log terms to be mathematically undefined or for the network to tend to an infinite gradient. PyTorch deals with this possibility by clamping its log function outputs to be greater than or equal to -100. Thus, the loss function is always finite, and backpropagation is linear. Alongside the BCE loss, we also optimise on a Mean-Squared Error (MSE) loss function for each GW parameter ϕ in $t_c, \mathcal{M}_c$. The unreduced form of this loss is given by

$$\begin{aligned} \mathcal{L}^{\phi}_{mse}(\mathbf{x}^{\phi}, \mathbf{y}^{\phi}) &= \{l^{\phi}_1, ..., l^{\phi}_n\}^{\top}, \\ l^{\phi}_i &= (x^{\phi}_i - y^{\phi}_i)^2, \\ \text{where } i &= 1, ..., n \end{aligned}$$

and noise samples that do not have a target y^{ϕ}_i and are ignored. For each ith batch element that is not a noise sample (for a batch size of n), x^{ϕ}_i and y^{ϕ}_i are the network output and normalised target for the given GW parameter and l^{ϕ}_i is the loss. Each network output x^{ϕ}_i is limited within the range $[0, 1]$ using a Sigmoid function, and the unreduced loss $\mathcal{L}^{\phi}_{mse}(\mathbf{x}^{\phi}, \mathbf{y}^{\phi})$ is reduced as $\overline{\mathcal{L}^{\phi}_{mse}(\mathbf{x}^{\phi}, \mathbf{y}^{\phi})}$ for every batch. The primary purpose of these loss functions is *not* to obtain a point estimate of the parameters, but to instead regularise the classification problem by acting as a soft boundary condition. The final batch loss is given by

$$\mathcal{L} = \overline{\mathcal{L}^{\text{gw}}_{bce}(\mathbf{x}, \mathbf{y})} + \sum_{\phi \in \{t_c, \mathcal{M}_c\}} \overline{\mathcal{L}^{\phi}_{mse}(\mathbf{x}^{\phi}, \mathbf{y}^{\phi})}.$$

Optimisation Parameters: The learning dynamics is optimised using an `Adam` optimiser [142] with an initial learning rate of 2×10^{-4} and a weight decay of 10^{-6}. The learning rate is then scheduled using `CosineAnnealingWarmRestarts` that cycles every 5 epochs, with a minimum learning rate of 10^{-6}. We use gradient clipping set to a maximum allowed gradient norm of 1 and a batch size of 64 for all our runs.

Biases Addressed: The MSE loss term, $\overline{\mathcal{L}^{\phi}_{mse}(\mathbf{x}^{\phi}, \mathbf{y}^{\phi})}$, provides an inductive bias to learn the GW signals properly. In other words, GW signals become easier to learn, thus addressing the *bias due to sample difficulty*. Refer to section 3.5 for more details about why this particular loss structure helps with learning regularisation.

4.2.7 Network Architecture

Sage uses a bottom-up hierarchical architecture with two main stages. The first stage is a frontend multiscale feature extractor for each detector that highlights the essential regions and, more importantly, de-emphasises unwanted attributes at various scales in the time domain. The second stage is a

backend classifier powered by attention to form temporal relationships within each detector and to emphasise any correlation, or lack thereof, between detectors. There is ample precedence in the machine learning literature to suggest that a multi-stage detection process that fragments the primary problem into multiple sub-problems will lead to better generalisation and faster convergence over single-stage approaches [238, 239, 240]. The rest of this section is dedicated to explaining parts of the two-stage architecture, their intended purposes, advantages, and necessity.

Frontend Feature Extractor: The frontend architecture is inspired by two main works[5]: [i] [241] that argues for the need for large kernels and provides design principles for learning contradictory tasks such as classification and localisation of features simultaneously [ii] [242] that designs a multiscale residual block called the OSNet.

The key feature of the OSNet [242] is to simultaneously analyse an input over multiple scales to create a comprehensive representation. The scale, dictated by the effective receptive field, can be varied by stacking n_c convolutional layers, with a higher value of n_c resulting in larger receptive fields. Stacking small kernel convolutional layers was initially introduced as an efficient alternative to using a single large kernel, as it results in a similar effective receptive field with a lower computational cost. However, [241] argue and empirically prove that large kernels lead to better generalisation in per-pixel classification tasks, compared to an equivalent stacking of small kernels. They conclude that the denser connection between large kernels and the output enables the network to handle input transformations more resiliently. Additionally, according to [241], to retain the ability to localise important features, the architecture must be fully convolutional and not contain any fully connected layers or global pooling layers.

Based on these key insights, we build our frontend architecture. A diagram of our large kernel multiscale residual block is shown in Fig. 4.13 (a). In Fig. 4.13 (b), these blocks are then stacked with intermittent downsampling via 1D max pooling to create our full feature extractor. Although we downsample the features along the temporal dimension using a stride S, this is performed after generating C_{out} multiscale output channels, where S and C_{out} are chosen such that the 2D output feature map is larger overall than the input. Thus, we retain the dense connections between kernels and the extractor feature output. Refer to sections 3.10.3 and 3.10.6 for an explanation of why SiLU is better than ReLU for our problem.

Reproducibility. With reference to Fig. 4.13(a), the large 1D kernels used for the multiscale feature extractor uses $N_s = 5$ scales with prefactors $s = \{0.25, 0.5, 1, 2, 4\}$. In Fig. 4.13(b), the initial kernel size k^0 is set to $q = 64$ and the initial number of output channels C^0_{out} is set to $p = 32$. These values were fine-tuned such that they minimise the loss during the training phase.

The input time series for each detector is provided to a separate frontend feature extractor (Fig.

[5] We also took inspiration from and had useful discussions with the G2Net 2021 Kaggle competition winners via private communications[180]

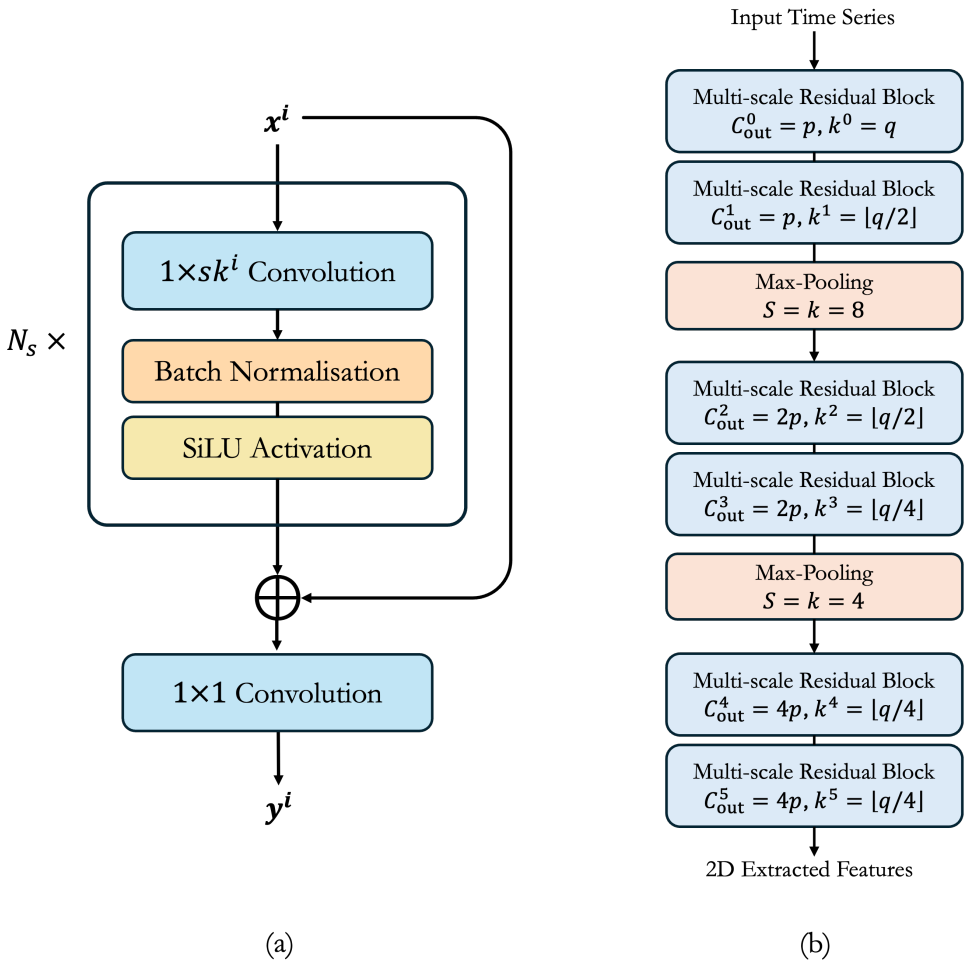


Figure 4.13: A diagrammatic depiction of the multiscale frontend feature extractor. (a) shows a single multiscale residual block. The input x^i is analysed separately and simultaneously by N_s different 1D large convolutional kernels with kernel size sk^i, where N_s is the number of scales, and s is a set of constant prefactors applied to the base kernel size k^i. The $\bigoplus$ is an element-wise addition. (b) shows the full frontend architecture.

4.13) and there is no weight sharing at this point between detector data. This is done to ensure that subtle differences within the data collected from each detector can be learnt without any interference. The small and long-range features extracted with the multiscale architecture should encode rich spectro-temporal information from the GWs, non-Gaussian noise artefacts and varying noise PSDs.

Backend Classifier: We use a variation of the ResNet architecture [243] introduced in [244] which integrates the Convolutional Block Attention Module (CBAM) into the ResBlock of a ResNet. The ResNet typically comprises a stack of ResBlocks, with the output feature map of one block feeding into the next. The residual nature of each ResBlock refers to the architectural motif $y = \Phi_s(x) + x$, where Φ_s is some arbitrary neural network module acting on an input feature map x. The recasting of the input feature map x into $\Phi_s(x)$ is called a residual connection, and it allows for the model to mitigate the vanishing gradient problem [243].

The salient design feature of CBAM is to sequentially infer attention maps along the channel and spatial dimensions of $\Phi_s(x)$. Say we define the output of the arbitrary neural network module within the ResBlock $F = \Phi_s(x)$ as an intermediate feature map. CBAM applies the spatial and channel-wise attention maps (M_s, M_c) to produce a final output $F'' = M_s(F') \otimes F'$ where $F' = M_c(F) \otimes F$ and $\otimes$ denotes element-wise multiplication. So y will be $F'' + x$ instead of $\Phi_s(x) + x$ (in reference to ResBlock motif). We run SAGE on a CBAM-ResNet50 backend variant for all experiments; refer to section 3.10.2 for an explanation of why a deeper network is better for this problem. The channel-wise attention $M_c(F)$ is computed as,

$$M_c(F) = \sigma\left[\Phi_c(P_{\text{avg}}(F)) + \Phi_c(P_{\text{max}}(F))\right] \tag{4.8}$$

where Φ is some multi-layered perceptron (MLP), $(P_{\text{avg}}, P_{\text{max}})$ are average pooling and max pooling operations respectively, and σ is the sigmoid activation function. The MLP has a single hidden layer that is smaller than the input, thus acting as a bottleneck, similar to an autoencoder architecture [245]. The spatial attention $M_s(F')$ is computed as,

$$M_s(F') = \sigma\left[P_{\text{avg}}(F') \circledast P_{\text{max}}(F')\right] \tag{4.9}$$

where $\circledast$ is the convolution operation. The usage of pooling operations for the aggregation of information is well-founded [246, 247] and CBAM exploits this feature across the spatial and channel dimension and empirically prove that the usage of both pooling operations together greatly improves detection capability. The spatial attention will aid greatly in forming temporal relationships within the time series of each detector, and the channel-wise attention will construct correlations between detectors or indicate a lack thereof. Please look at Fig. 4.14 for a diagrammatic representation of equations 4.8 and 4.9.

Point-estimate Outputs: The CBAM-ResNet50 is made to output a 512-dimensional embedding, which is passed through three separate (i.e., no shared weights) fully connected layers with 512 neurons each. Each layer is then responsible for one unnormalised output. A sigmoid activation function is then applied to each output to restrict it between $[0, 1]$. The three normalised outputs are point estimates on the probability of a GW event, p_{GW}, and the normalised GW parameters $\{t_c, \mathcal{M}_c\}$. The normalised values are used to compute the loss function and the unnormalised counterpart of p_{GW} is used as the output ranking statistic (also referred to as the network output). Through backpropagation, the 512-dimensional embedding should encode information about all outputs, given the composite loss function in equation 4.2.6. We choose chirp mass as it is the best-measured parameter for the most observed mergers, as it determines the leading-order GW radiation during the inspiral phase [248]. This acts as a soft constraint and provides an additional inductive bias to aid the network in distinguishing between the signal and noise classes. The time of coalescence estimate acts as a reference point and aids in feature localisation. Furthermore, we can also use the t_c estimate as input to parameter estimation pipelines to investigate the generated triggers. An example of this estimate for

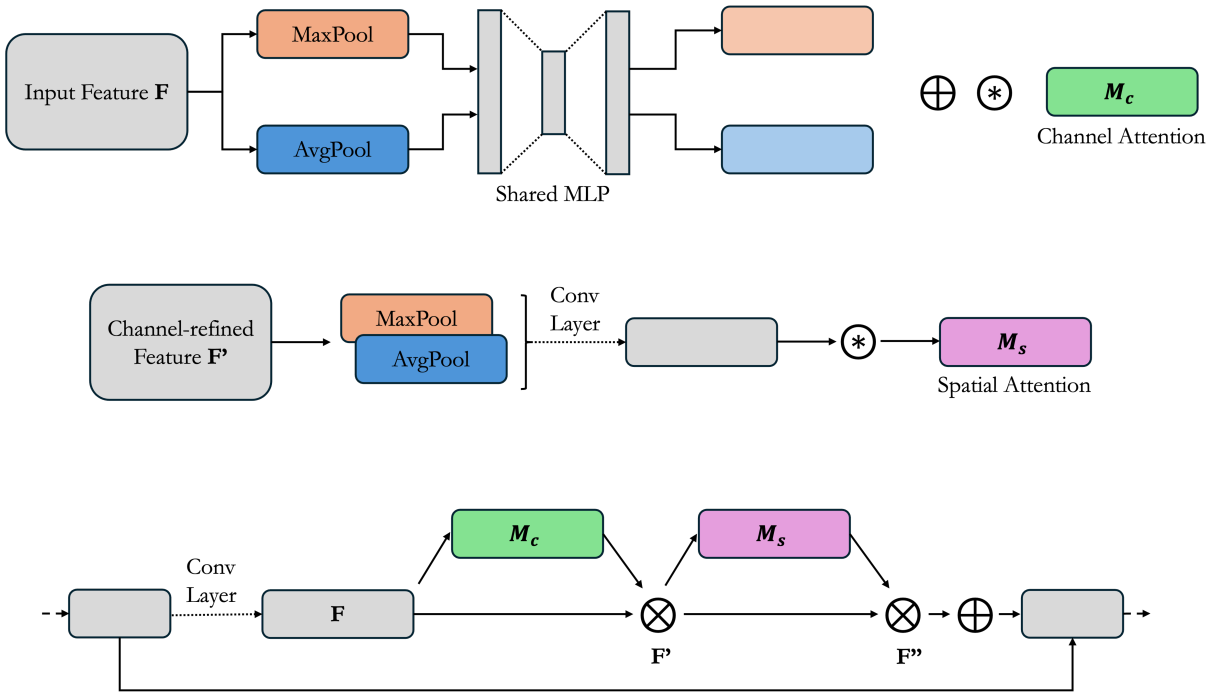


Figure 4.14: Diagram of the CBAM mechanism for spatial and channel attention.

parameters $\{t_c, \mathcal{M}_c\}$ is shown in Fig. 4.31 (see appendix 4.10) for two different SAGE runs (see section 4.3.1 for details about runs).

Biases Addressed: The multiscale frontend with large kernels is useful when learning time series signals with a high dynamic range of frequencies [249, 241]. Spatial attention aids the model in learning amplitude, frequency, and phase evolution profiles by forming temporal relationships using the multiscale features. The attention mechanism automatically increases the weight of under-represented scales, thus addressing *spectral bias*. The attention mechanism and multiscale architectures have been used before to address this bias, as seen in [250, 251, 252]. Channel-wise attention aids in forming a correlation between the temporal relationships formed for the signal in each detector. The two attention mechanisms aid significantly in learning long-term dependencies [225, 253, 254], addressing the *bias due to sample difficulty*. Refer section 3.11.3 for more details on why the attention mechanism is advantageous for our detection problem. Moreover, training a deeper network for longer (without overfitting) increases the effective receptive field, which also aids in learning long sequences [226]. Our architecture is built to take advantage of increasing representation capacity for a given parameter count; see section 3.10.4 for more information. We will see later in ablation study III why this matters.

In Fig. 4.15 we provide a recap of the full SAGE methodology and indicate some of the mitigation tactics introduced so far.

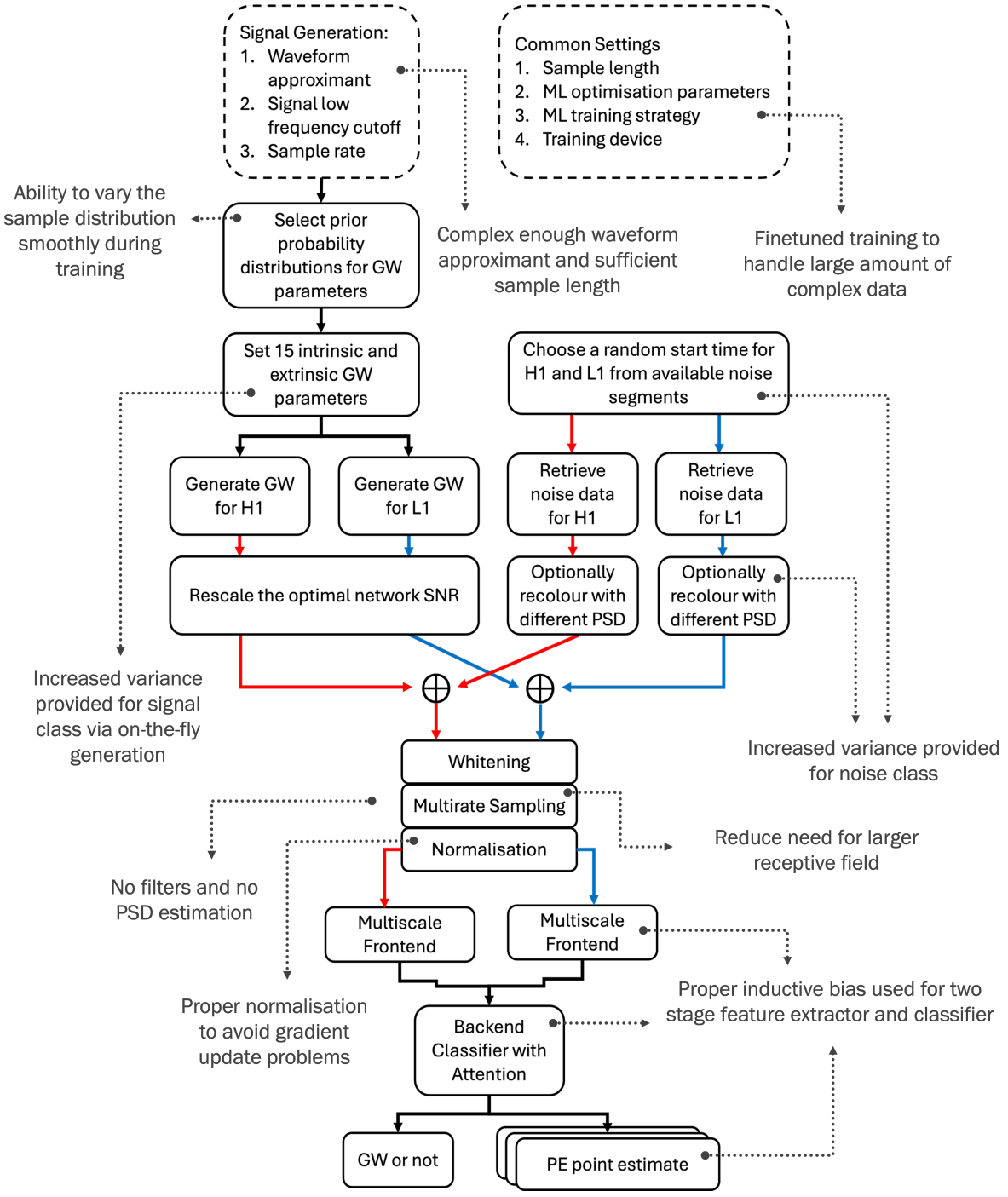


Figure 4.15: Recap of the Sage methodology with information about the biases addressed so far. Please look into the individual sections for more information.

4.3 Results and Discussion

We will use this section to further describe the injection study, elaborate on the various training configurations employed, and compare our results with previous works on the same datasets (D_3 and D_4) [204, 209, 186]. Furthermore, we analyse the behaviour of Sage under different conditions and discuss the validity of juxtaposing different results on established evaluation metrics. For the sake of completeness, we also investigate the performance of Sage by removing certain important components from our pipeline to understand its contribution to the overall system (also known as an ablation study).

We ran Sage on multiple training configurations to investigate the impact of different biases, time-varying PSDs and glitches. Table 4.2 gives a brief overview of the runs that we will be discussing in this section. Any configuration that is not mentioned in this table is consistent between all runs and is as described in section 4.2. The steps taken to evaluate Sage after the end of training are provided here in sequential order:

Run Name	Training Data - Mass Distribution	Noise Segment for Recolouring
Sage — Broad	$U(m_1, m_2)$	{O3a(Day 1 – 30)}
Sage — Limited	$U(m_1, m_2)$	{O3a(Day 36 – 81), O3b(Day 1 – 113)}
Sage — D_4 Metric	$U(\tau_0, \tau_3)$	{O3a(Day 1 – 30)}
Sage — Annealed	$U(m_1, m_2) \longrightarrow U(\tau_0, \tau_3)$	{O3a(Day 1 – 30)}
Sage — D_3 Metric	$U(\tau_0, \tau_3)$	–

Table 4.2: Summary of all Sage runs discussed in the results section. Column 1 gives a shorthand for the discussed runs and will be used to describe a singular training instance of that type that achieved the highest sensitive distance at an FAR of one per month. Column 2 describes the distribution of component masses (m_1, m_2). The τ_0-τ_3 coordinate pair can be analytically inverted to the $\mathcal{M}$-η pair, as described in section 4.2.2, which can then be used to derive the distribution of component masses. The $\longrightarrow$ symbol refers to a particular training strategy of transitioning from one distribution to another (refer to the Sage — Annealed Train run in section 4.3.1 for more details). Finally, column 3 provides details about the noise segment used to obtain precomputed PSDs that are meant to recolour the training noise realisations.

Common settings: As mentioned in sections 4.2.2 and 4.2.1, the training noise is O3a(Day 36-51) and O3b(Day 1-113) and the testing noise is O3a(Day 1-30) for all our runs. All runs were trained with 80 million samples (1:1 ratio between the signal and noise class) split equally over 40 epochs.

Best trained model: The validation noise dataset will be slightly different, compared to the training and testing datasets, in terms of noise PSDs and glitches. Thus, we retrieve the network learnable parameters (i.e., weights and biases) that correspond to the 5 lowest epoch validation losses for testing. We then test the model on the "background" and "foreground" testing datasets discussed in section 4.2.1 for each set of network parameters, and pick the one that detects the most signals at an FAR of 1 per month.

Testing iteration: The 30-day testing data is split as sequential windows with a step size of 0.1 seconds. Each window is whitened, multirate sampled and normalised, consistent with the training dataset, prior to being provided as input to the trained network. The network outputs a ranking statistic for each 12-second window, which we assume originates due to a possible GW event with a time of coalescence within $[11.0, 11.2]$ s (the bounds used during training). This is called a *trigger*. Given that the step-size is 0.1 s, a GW event can have at most 3 triggers.

Trigger clustering: To identify independent gravitational wave candidates, we cluster nearby triggers based on a predefined time threshold (2 seconds). We iterate through the list of triggers, grouping consecutive ones if their time difference is below the threshold; otherwise, a new cluster is initiated. Each cluster is then represented by its most significant trigger, determined by the maximum ranking statistic within the cluster. Any subsequent use of the term "trigger" must be interpreted as this independent cluster maxima. This approach ensures that multiple responses to the same underlying signal, or to closely spaced noise fluctuations, are not overcounted.

Labelling cluster maxima: Only triggers above a ranking statistic of 0 are retained (to reduce compute cost). No prior knowledge about the location of the injected events is taken into account when assigning the trigger a ranking statistic. For the foreground dataset, triggers within $[t_c^{\text{inj}} - 0.2,\ t_c^{\text{inj}} + 0.2]$ seconds of the 12-second window, where t_c^{inj} is the time of coalescence of any one of the injected signals, are labelled as *true positive* triggers for the corresponding injection (not to be confused with a binary ML label). For each true positive trigger, we get a point estimate of the time of coalescence and chirp mass from Sage (see section 4.2.7). All other foreground triggers that do not correspond to a known event are labelled as *false positives*. For the background dataset, all triggers are labelled as false positives. The labels provided to the triggers are purely for the purpose of computing the following evaluation metrics. [6]

Evaluation metrics: The sensitive distance metric at different FARs is computed as described in section 4.1.3. To compute receiver operating characteristic (ROC) curves, we use the true positive triggers for each signal from the foreground dataset and false positive triggers from arbitrary noise windows of the background dataset. Triggers in the foreground dataset marked as false positive can originate from windows where a GW injection is present with a t_c outside the $[11.0, 11.2]$ s range. The behaviour of the ML model for these windows is not guaranteed to resemble that of a pure noise sample or a noisy signal sample, and thus the false positive label is not accurate. Since we cannot confidently assign the false positive label to these samples, it is not possible to include all foreground dataset triggers for the computation of the ROC curve.

All tests were run on 16 CPU cores and an NVIDIA Quadro RTX 8000 GPU. Each epoch of training takes $\approx$ 2 hours, excluding validation. Running Sage on evaluation mode, it takes $\approx$ 16

[6] For a fair comparison to the MLGWSC-1 benchmark results, we use the code provided at https://github.com/gwastro/ml-mock-data-challenge-1/blob/master/evaluate.py to evaluate our model.

hours to generate triggers on 1 month of testing data.

4.3.1 Comparison with Matched Filtering

We compare Sage to the PyCBC submission in MLGWSC-1 [204] for the realistic D4 injection study. The submission is based on the standard configuration of PyCBC used for archival searches of compact binary mergers, as seen in [255] (particularly the "Broad" configuration *without* an extra tuning term in the ranking statistic for a specific signal population). The template bank they used to evaluate this injection study included templates that account for spins aligned with the orbital angular momentum, did not include precession or higher-order modes, and had mass parameters that were bound by the testing distributions provided in table 4.1 and were placed in a stochastic manner. They account for the presence of glitches [256, 257, 258] using waveform consistency checks [66, 116] and ensure that the amplitude, phases, and time of arrival in each detector are consistent with the astrophysical population [259]. PyCBC was run separately on the "background" and "foreground" datasets, and a ranking statistic was assigned to potential candidates based on an empirically measured noise distribution and the consistency with the expected GW [259] (without attempting to optimise sensitivity for a particular signal population).

Although PyCBC uses a template bank that does not account for precession or higher-order modes, we don't expect the detection efficiency to be too different from a template bank that does, since neither of those effects is detectable in this injection study (refer to section 4.3). Although there would exist slight differences in the signal morphology between injections generated using `IMRPhenomXPHM` and the relatively simpler templates used by PyCBC, the network SNR of each detected injection should not vary significantly due to the simpler templates. Thus, although we use `IMRPhenomPv2` to train Sage, there is no precedent to expect a better detection efficiency from Sage solely due to the usage of a waveform model that is relatively more similar to the injections.

Injection Study

In this section, we describe each of the runs in table 4.2, compare them and describe how ML biases influence their detection performance. All aspects of the methodology are consistent with section 4.2, and any changes or choices made are described within each run description. Some choices are common for all runs and are described here:

For all runs, we rescale the optimal network SNR of all samples during training, as mentioned in section 4.2.2. From experimentation, we found that using a half-normal distribution allowed for a higher detection efficiency than using a uniform distribution in the same range. The half-normal is described by the probability density function,

$$f(x) = \sqrt{\frac{2}{\pi}} \exp \frac{-x^2}{2}, \tag{4.10}$$

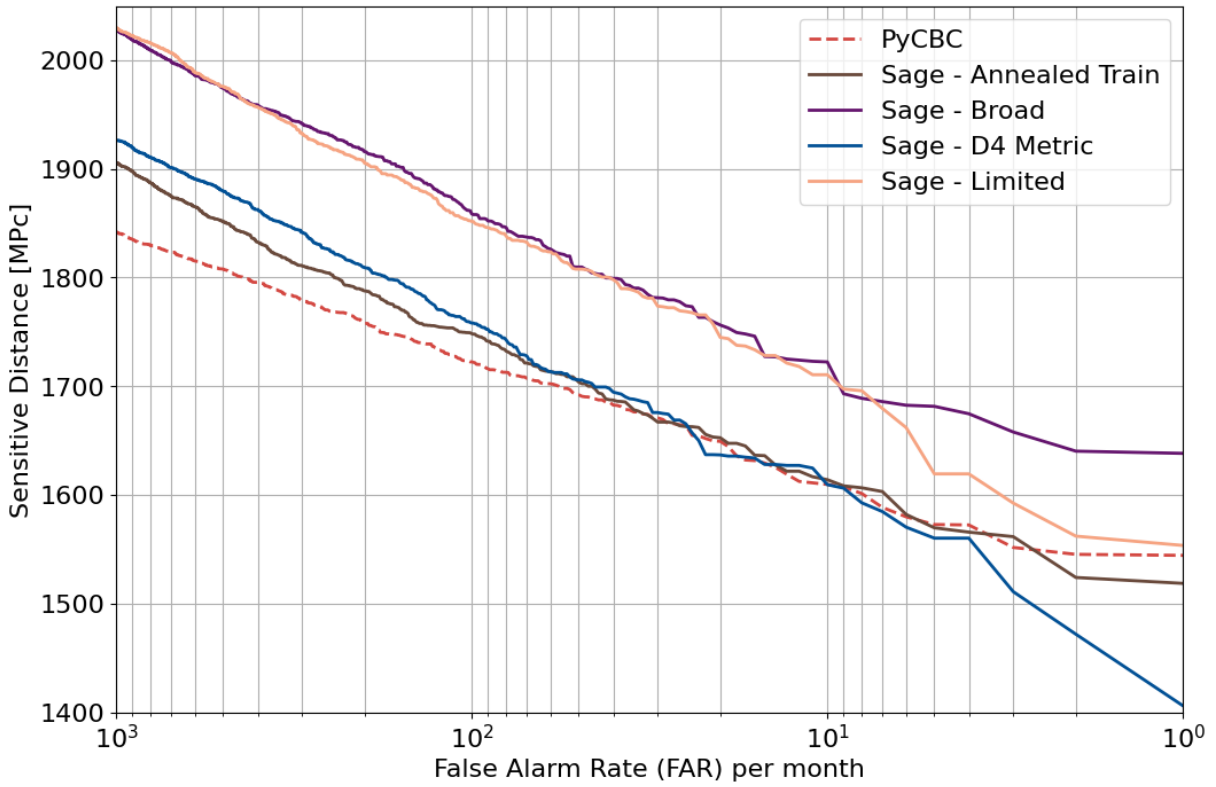


Figure 4.16: The sensitive distance metric as a function of false alarm rate for PyCBC and a set of different Sage analyses of the testing dataset.

where $x \geq 0$. The samples generated using this density are scaled by a factor of 4 and shifted by a factor of 5. This places the lower bound at an SNR of 5. From a sample size of 10^6, we find that 99% of the distribution lies below an SNR of ≈ 15.3 and the maximum SNR is ≈ 24.3.

Sage - Limited: We use detector noise from O3a(Day 36-51) and O3b(Day 1-113) for training. This is then recoloured using precomputed PSDs from the same data with a probability of 0.4 (value chosen via fine-tuning). Figure 4.16 shows a comparison of the sensitive distances of Sage - Limited and PyCBC as a function of FAR. We see that our run achieves a higher sensitive distance than PyCBC at all FARs from 1000/month to 1/month. However, Fig. 4.17 (top left) shows that the run does not find as many signals as PyCBC. The higher sensitive distance is due to finding a few more high chirp mass signals than PyCBC. As described in section 4.1.3, this is a consequence of a bias against low chirp mass signals (incurred from biases due to insufficient information, limited sample representation, sample difficulty and spectral bias) and a parameter-dependent weighting of the sensitive distance metric. Although our bias mitigation strategies improved the detection efficiency at lower chirp masses compared to runs without them, it is still not sufficient to match PyCBC's detection efficiency.

Sage - Broad: The presence of OOD PSDs in the 30-day D4 testing dataset, as seen in Fig. 4.8, might lead to reduced detection efficiency due to the model's inability to extrapolate to unseen PSDs. In this experiment, we determine whether closing this gap would affect the detection efficiency. The

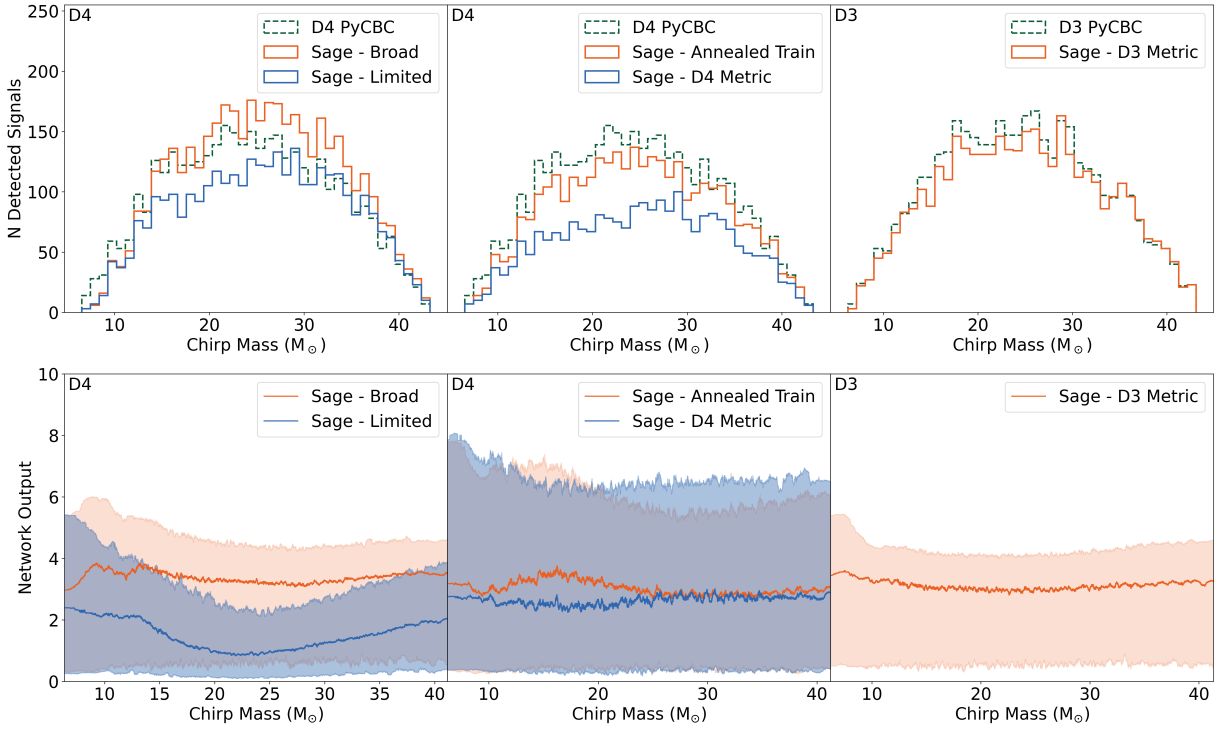


Figure 4.17: *(top)* 1D histograms of detected signals along the chirp mass dimension of different Sage runs with PyCBC at an FAR of 1 per month. The *(bottom)* row of Fig. 4.17 shows the ranking statistic output by the network compared to the chirp mass $\mathcal{M}_c$ of 250,000 signals. The ranking statistic is put through a running mean across τ where we use a window size of 500 samples and a step size of 1 sample. The error bar on each plot indicates the region between the 5th and 95th percentile region of the network output at a given chirp mass.

only change we introduce in this experiment is to use O3a(Day 1-30) to precompute the PSDs used to recolour the training data O3a(Day 36-51) and O3b(Day 1-113), instead of using the training data itself. This intentionally introduces some mild overlap between the training and testing datasets, although the recoloured noise realisations are strictly speaking different from the testing dataset.

From Fig. 4.16, it is immediately evident that we see a significant boost in the sensitive distance, from an FAR of 10/month to 1/month. We see an increase of $\approx 5.5\%$ and $\approx 6.5\%$ in sensitive distance when compared to Sage - Limited and PyCBC respectively at an FAR of 1 per month. Figure 4.18 shows the median sensitive distance for this run with an error bar made by repeating this run 3 times with different seeds. The seeds were set such that both signal and noise realisations for all epochs during training are different but within the same distribution and that the networks have different random initialisations for their trainable parameters. We also provide the parameter-independent ROC curve that compares this result to PyCBC in Fig. 4.19. At a false alarm probability of 10^{-5} we see an $\approx 38\%$ increase in the true alarm probability. Figure 4.17 (top left) also shows the 1D chirp mass histograms of the detected signals, comparing Sage - Limited and Sage - Broad. The two runs perform similarly at chirp masses below $\approx 10\,M_\odot$, but the latter outperforms the former significantly everywhere else.

The increase in detection performance when compared to Sage - Limited points to two possible causes: [i] an increase in the variance of training noise realisations (via their PSD), or [ii] all testing noise PSDs are in-distribution for Sage - Broad, but the same is not true for Sage - Limited. We will investigate the root cause in the following runs and in section 4.3.1.

Caveats of the evaluation metrics: We generally attribute an increase in the true alarm probability of an ROC curve at a given false alarm probability, and the sensitive distance metric at a given FAR to better detection performance. However, this is not entirely correct, and both evaluation metrics have their disadvantages. Assuming Gaussian noise, a model that is biased against low chirp mass signals will operate on a narrower parameter space and thus detect more high chirp mass signals at any given FAR. This bias would result in a significant increase in the sensitive distance metric, given the chirp mass dependence of the sensitive volume (see equation 4.5). A parameter-independent metric, like the ROC curve, would still only account for the number of detections made and not the distribution of the detections, thus masking any biases. The use of only a single figure of merit based on a pre-determined set of signals may be of very limited application to the real-world problem of detecting signals with an unknown parameter distribution. For instance, an algorithm may achieve a high figure of merit while being almost completely insensitive to signals in some region of parameter space, which may be much more prevalent in reality than in the evaluation set. In other words, optimising detection for a known target population may imply a greater risk of losing sensitivity for the unknown real population.

We visualise the 1D and 2D distributions of the detected signals as histograms in Fig. 4.20, to compare and visualise any differences between Sage - Broad and PyCBC. We find that 3344 signals are detected by both pipelines, 513 signals were found only by PyCBC and 946 signals were found only by Sage at an FAR of 1/month. This is equivalent to an increase of $\approx 11.2\%$ in the number of signals detected compared to PyCBC in the MLGWSC-1 realistic injection study. We find no anomalous biases present across any parameter that is not directly related to chirp mass, mass ratio or SNR. A full set of 1D histograms for all relevant intrinsic and extrinsic parameters are provided as 1D histograms in Fig. 4.29 in appendix 4.8.

Important note: In Fig. 4.20, Sage - Broad and PyCBC operate on two different mass distributions - the former on $\mathrm{U}(m_1, m_2)$ and the latter on $\mathrm{U}(\tau_0, \tau_3)$. For a fair comparison of their detection sensitivities, the search statistic for PyCBC must be fine-tuned and optimised for the $\mathrm{U}(m_1, m_2)$ mass distribution (see [104, 105] for relevant information). This fine-tuning will likely lead to an increase in the number of signals detected by PyCBC in the high chirp mass region. However, given that the effect of blip glitch noise is much higher in this region, it is not straightforward to speculate on the extent of improvement that might be seen in a fine-tuned PyCBC [66, 95]. This study is left for future work. Furthermore, the small number of low SNR, low chirp mass signals missed by Sage in Fig. 4.20 could be attributed to this difference in mass distribution.

Sage - D3 Metric: We found many training training distribution configurations that reduced the

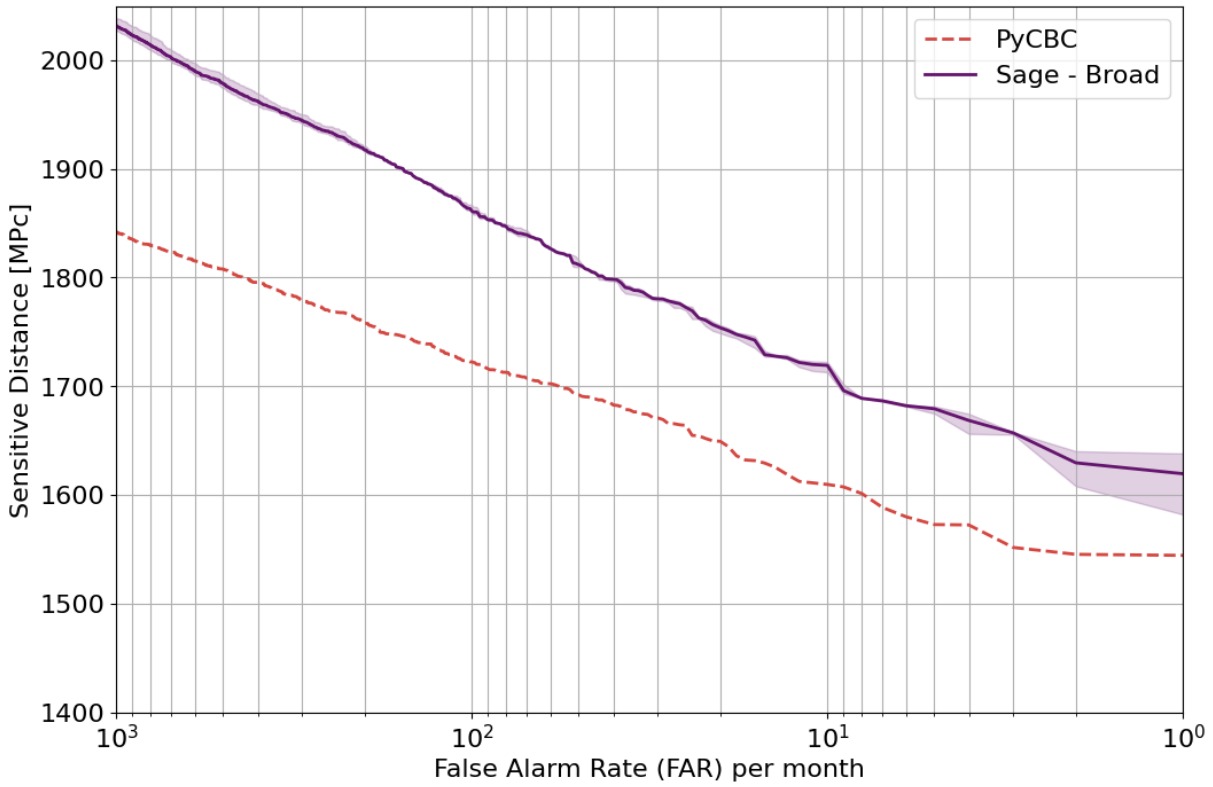


Figure 4.18: The sensitive distance metric as a function of FAR for PyCBC and the Sage - Broad run on the testing dataset. The solid violet line is the mean sensitive distance of 3 runs made with the "Broad" configuration, and the error bar represents the max and min values obtained from the runs.

bias across the chirp mass and mass ratio dimensions, but for this run, we investigate using $U(\tau_0, \tau_3)$ () to set the training distribution for the component masses. A detailed description for using this distribution for training is defined in section 4.2.2. The key difference between $U(m_1, m_2)$ (which we used in the previous runs) and $U(\tau_0, \tau_3)$ is that the latter has an underdensity of training data samples in the high chirp mass region (see Fig. 4.9). For this run, we train and test on the D3 dataset setting with simulated coloured Gaussian noise instead of real noise. This setup is ideal for studying the effects of varying the training distributions of GW parameters, as it avoids introducing learning biases from real detector noise.

Figure 4.17 (top right) compares the 1D histograms of the detected signals at an FAR of 1 per month of Sage - D3 Metric to PyCBC. We see that using $U(\tau_0, \tau_3)$ as the training mass distribution alongside simulated Gaussian noise produces a detection efficiency very similar to PyCBC at all chirp masses. This justifies the use of the parameter distribution on real detector noise as well.

Sage - D4 Metric: This run also uses $U(\tau_0, \tau_3)$ to set the training distribution for component masses, but uses real detector noise (similar to all other runs on D4) instead of simulated coloured Gaussian noise. Compared to Sage - D3 Metric we see that Sage - D4 Metric has significantly lower detection efficiency, in Fig. 4.17 *(top middle)*. This reduction is solely due to the attributes of real

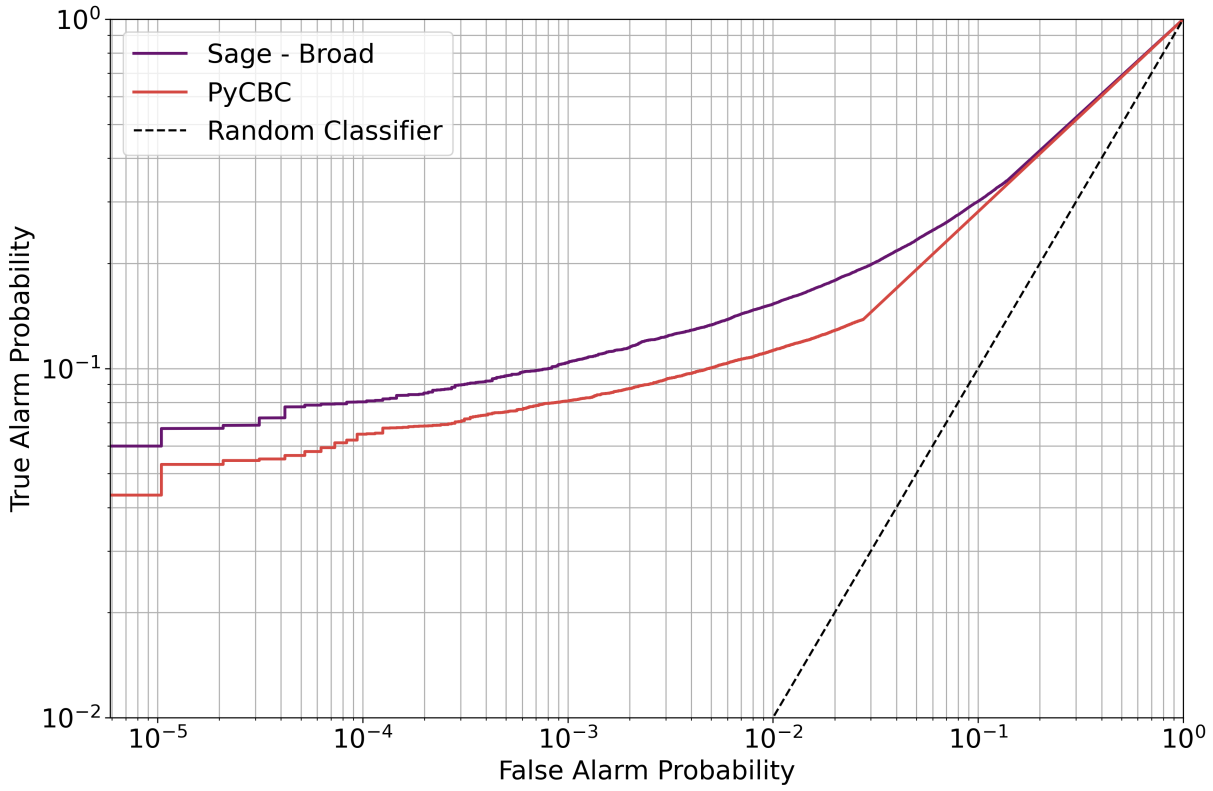


Figure 4.19: The true alarm probability as a function of false alarm probability (ROC curves) for PyCBC and the Sage - Broad run that achieved the highest sensitive distance on the testing dataset.

noise; it is either due to the inability to properly reject non-Gaussian artefacts or due to the difference between the training and testing noise PSDs.

Sage - Annealed Train: An underdensity of training data samples in the high chirp mass region results in worse noise rejection capability, as seen in Sage - D4 Metric, and an overdensity results in a bias against long duration signals, as seen in Sage - Broad. We tackle this issue by viewing the GW detection problem as the learning of two independent tasks: noise rejection capability and unbiased learning of the GW parameter space. Since the learning of these tasks simultaneously is not trivial, we introduce the concept of annealed training. For the Sage - Annealed Train run, we train a network similar to Sage - Broad for 20 epochs (with a mass distribution given by $\mathrm{U}(m_1, m_2)$) and then transition the mass distribution to $\mathrm{U}(\tau_0, \tau_3)$ over the course of the next 20 epochs. During the transition phase, the ratio between the number of samples generated using $\mathrm{U}(\tau_0, \tau_3)$ to $\mathrm{U}(m_1, m_2)$ varies as a function of epoch.

The intent of annealing is two-fold: taking advantage of catastrophic forgetting [260] and the concept of "unforgettable" examples [261, 262]. In the context of continual learning, catastrophic forgetting is the tendency of a neural network to abruptly lose knowledge relevant to a previous task as information about the current task is incorporated [260]. This is undesirable in continual learning, but beneficial to our problem for shifting the training distribution from $\mathrm{U}(m_1, m_2)$ to $\mathrm{U}(\tau_0, \tau_3)$. Unfor-

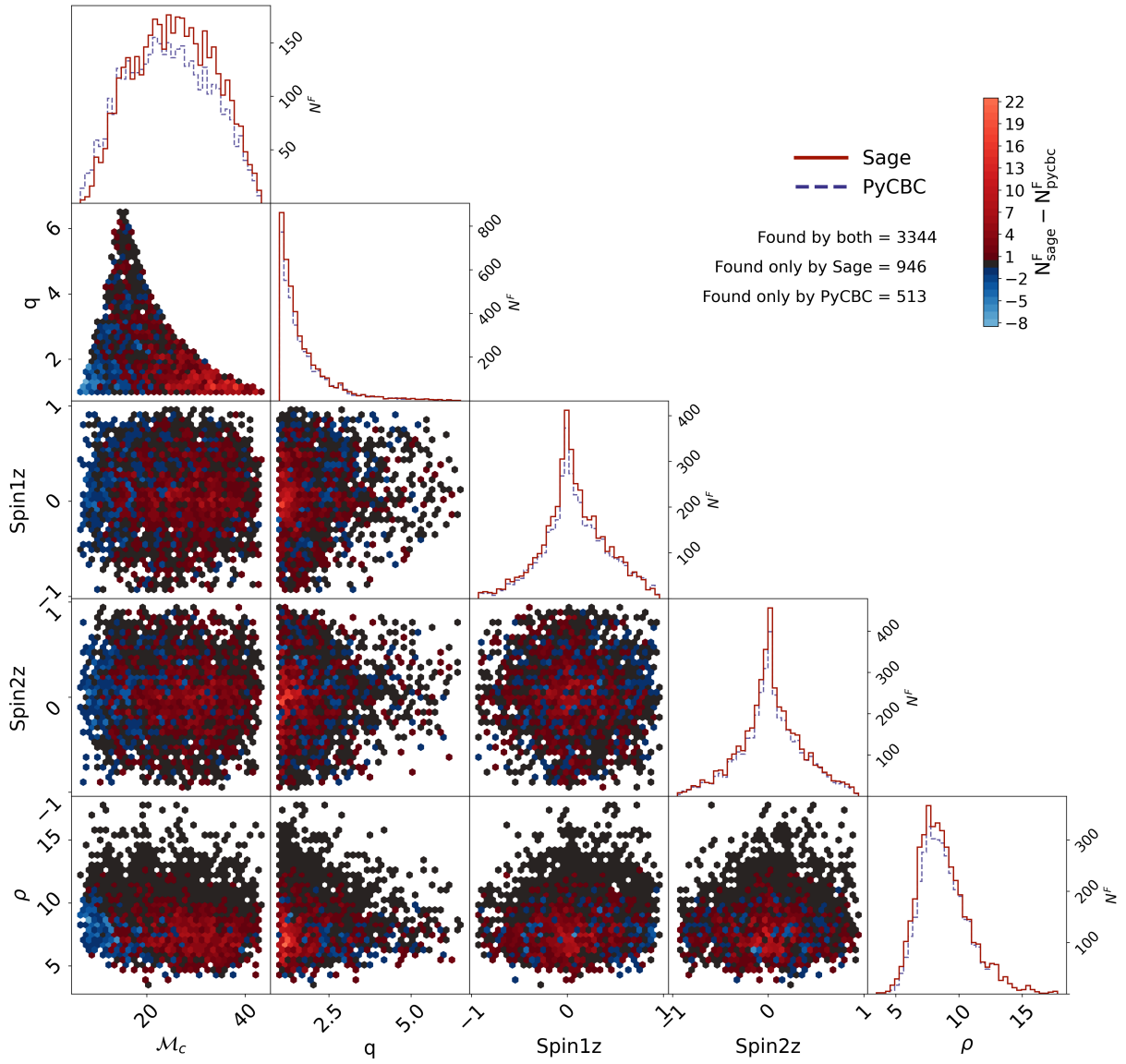


Figure 4.20: 1D and 2D histograms of detected signals for different GW parameters at an FAR of 1 per month, comparing the best SAGE - BROAD run and PYCBC. We chose GW parameters where we expected to see learning bias. The diagonal subplots contain 1D histograms of the detected signals for a given parameter. The 2D histograms are hexagonal binned plots, where each bin is coloured using $N^F_{\text{sage}} - N^F_{\text{pycbc}}$ where N^F_{sage} and N^F_{pycbc} is the number of signals detected by SAGE and PYCBC respectively. We use a divergent colour map that is set to be black when $N^F_{\text{sage}} - N^F_{\text{pycbc}}$ is zero, is progressively brighter in red along the positive axis, and is progressively brighter in blue along the negative axis. So, red and blue patches can be attributed to regions where SAGE and PYCBC detect more signals, respectively.

gettable examples (or more precisely, difficult-to-forget examples) are input data samples that are learnt early during training and typically have low losses during the bulk of the training [262]. This allows for all high SNR signals to be remembered well throughout training and not be forgotten during the annealing process. This should allow for any noise rejection capability learnt via high SNR signal features to also be retained.

Figure 4.17 (top middle) compares the 1D chirp mass histograms of the detected signals at an FAR of 1 per month between the Sage - Annealed Train and Sage - D4 Metric runs. We see a significant increase in the detection efficiency of the former run compared to the latter, proving the efficacy of the annealing training strategy.

Network output behaviour: We evaluate all the above runs on a validation dataset of 500,000 samples (1:1 ratio between signal and noise classes) comprised of signals and noise from the testing dataset distribution. Figure 4.17 (bottom) shows the network output obtained from this validation dataset for the signal class as a function of its corresponding chirp mass. The larger the network output, the more confident the model is to place that signal in the signal class. Notable observations have been described below.

Observation 1: There is a significant increase in the confidence of Sage - Broad when classifying signals across the whole range of chirp masses when compared to Sage - Limited. This is especially evident above a chirp mass of $\approx 15\ M_\odot$. Sage - Broad has the least classification confidence for chirp masses below $\approx 10\ M_\odot$ which contributes to the lower number of detections in this chirp mass regime compared to PyCBC.

Observation 2: The annealing procedure for Sage - Annealed Train improves the classification confidence from a chirp mass of $\approx 10\ M_\odot$ till $\approx 25\ M_\odot$. This indicates that the increase in detection efficiency seen in Fig. 4.17 (top middle) for Sage - Annealed Train over Sage - D4 Metric is primarily due to retaining information in this chirp mass region when annealed from the $U(m_1, m_2)$ mass distribution. Surprisingly, signals with a chirp mass above $\approx 25\ M_\odot$ do not contribute much to the improved noise rejection capability.

Observation 3: On average the Sage - D3 Metric network is a lot more confident with classification than Sage - D4 Metric throughout the chirp mass range. Although the latter, on average, has lower confidence, it detects many signals across the whole range of chirp mass with higher confidence than Sage - D3 Metric and even Sage - Broad. Using the metric as the training distribution leads to a lower *bias due to limited sample representation*, and in turn, the network is provided with more GW domain knowledge. This allows it to be more confident with some signals. However, Sage - Annealed Train and Sage - D4 Metric do not seem to perform well at lower SNRs leading to the worse average performance (there are more low SNR than high SNR signals; see the SNR distribution in Fig. 4.28 in appendix 4.7).

Observation 4: Interestingly, the upper error bar on all runs shows that the networks find longer signals with the highest ranking-statistics. However, other than Sage - D3 Metric, none of the other runs are good at low SNR low chirp mass signals, leading to a lower average network output at low chirp masses.

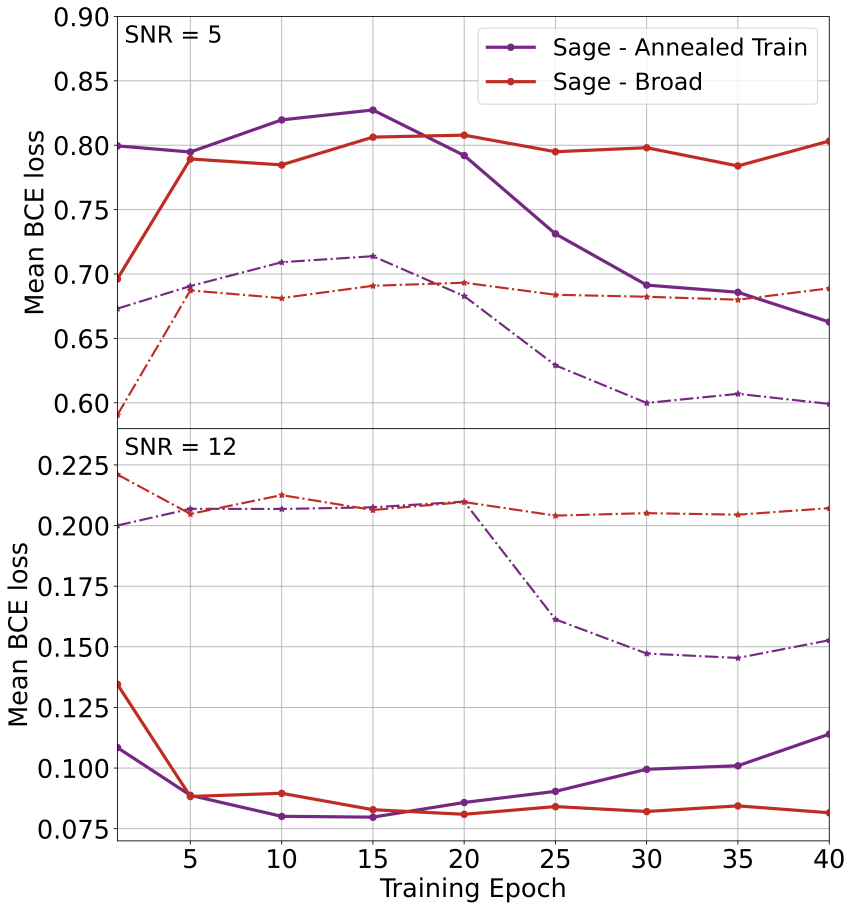


Figure 4.21: Mean BCE loss as a function of training epochs (only for the signal class). The plots compare the auxiliary dataset validation loss curves for the Sage - Broad and Sage - Annealed Train runs. The solid lines correspond to the low chirp mass dataset $\mathrm{U}[\mathcal{M}_c^{\mathrm{low}}, \mathcal{M}_c^{\mathrm{median}}]$ and the dashed lines correspond to the high chirp mass dataset $\mathrm{U}[\mathcal{M}_c^{\mathrm{median}}, \mathcal{M}_c^{\mathrm{high}}]$. As the plot text states, the top plot is in the low SNR regime (= 5), and the bottom row is in the high SNR regime (= 12).

Learning Dynamics

Here, we investigate the difference between Sage - Broad and Sage - Annealed Train to potentially bridge the gap between the two runs. We create 4 auxiliary validation sets to analyse the learning dynamics in different regions of the parameter space. We chose all unique combinations without duplicates, between the two sets of distributions: $\{\mathrm{U}[\mathcal{M}_c^{\mathrm{low}}, \mathcal{M}_c^{\mathrm{median}}], \mathrm{U}[\mathcal{M}_c^{\mathrm{median}}, \mathcal{M}_c^{\mathrm{high}}]\}$ and $\{\delta(\mathrm{SNR}{=}5), \delta(\mathrm{SNR}{=}12)\}$, where δ is the Dirac delta functions. This creates 4 datasets with different chirp mass and SNR regimes. At the end of each training epoch, we freeze the weights and run the network on each of these auxiliary validation datasets with 500,000 samples (1:1 ratio between signal and noise classes). This provides us with the learning dynamics of the network in each regime. Figure 4.21 depicts these 4 losses (obtained only for the signal class) over the course of 40 epochs of training for the runs Sage - Broad and Sage - Annealed Train.

Low SNR regime (see Fig. 4.21 (top)): For the Sage - Broad run, we see that low chirp mass signals

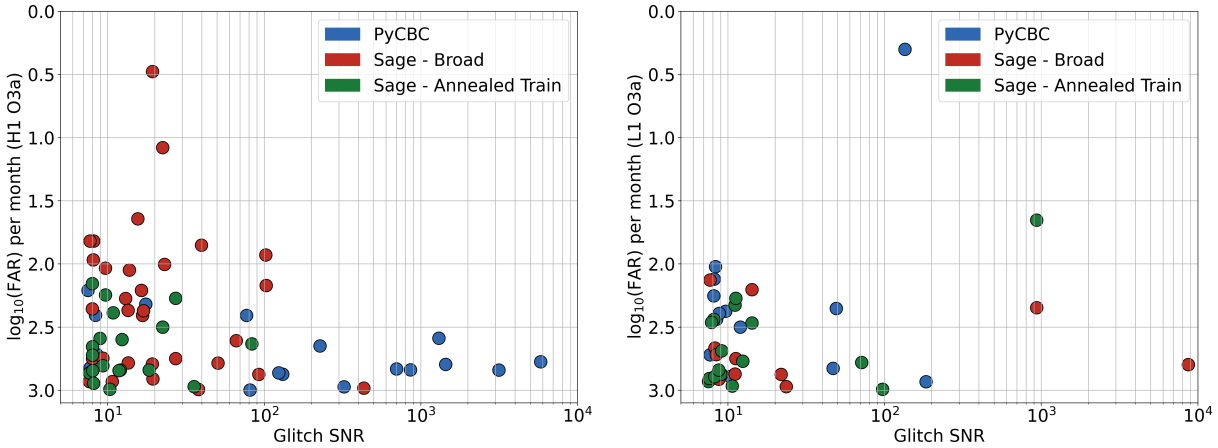


Figure 4.22: Log FAR per month as a function of glitch SNR (obtained from GravitySpy [233]) for the H1 and L1 detectors. The scatter plot shows the glitch triggers produced by different Sage runs and PyCBC from the two detectors in the O3a testing dataset noise.

are more difficult to learn compared to high chirp mass signals, given that they have higher losses. It is interesting that both losses increase with more training, contradictory to typical loss curve dynamics. This is likely due to the need to reject glitches which could be similar in morphology compared to these signals.

High SNR regime (see Fig. 4.21 (bottom)): We see a surprising trend where low chirp mass signals have much lower losses compared to high chirp mass signals, despite the expected biases against low chirp mass signals. Both losses have a decreasing trend, which is consistent with expectations since they should be easier to distinguish from non-Gaussian noise given their SNR.

Effect of annealing: The transition from $U(m_1, m_2)$ to $U(\tau_0, \tau_3)$ happens at epoch 20. Here, we see a mild increase in the loss for high SNR low chirp mass signals despite having an overdensity of low chirp mass signals, which is indicative of a contribution to noise rejection capability. We also observe that the losses pertaining to high chirp mass signals decrease steadily during the annealing procedure at both SNR regimes. However, despite having lower losses than Sage - Broad at high chirp masses, Sage - Annealed Train is not as performant. This is because we don't show the total loss function, which includes the noise class as well. The total loss of Sage - Annealed Train is higher than Sage - Broad, which explains the discrepancy.

Given the overall lower losses for the signal class in Sage - Annealed Train compared to Sage - Broad, the key to improving the detection efficiency of Sage - Annealed Train must lie in improving the noise rejection capability. This requires an in-depth investigation of the noise behaviour and its effects in the learning of different regions of the GW parameter space. We leave this study for the future.

Glitch Rejection Capability

In this section, we investigate the effect of the presence of glitches in the testing dataset. If the highest noise triggers are caused by glitches, the model would be extremely brittle to the testing dataset noise realisations and its detection efficiency at lower FARs has the potential to vary significantly with different datasets.

We get the start time and duration of all glitches reported in the Omicron trigger pipeline [263], with a glitch SNR above 7.5 and a peak frequency of the noise event between 10 Hz and 2048 Hz, for the O3a testing dataset noise segments in the H1 and L1 detectors. The triggers that lie within the start GPS time and end GPS time of each glitch are obtained from Sage for different experiments and PyCBC for the D4 submission to MLGWSC-1. We then maximise these triggers to obtain the ranking statistic of a single trigger that represents each glitch. Omicron produces a glitch SNR from the time-frequency representation of the strain and cannot be compared to the integrated signal SNR. We plot the maximised ranking statistic of triggers below an FAR of 1000/month against its corresponding glitch SNR in Fig. 4.22 for the runs Sage - Annealed Train and Sage - Broad and compare it with PyCBC.

We observe that all triggers produced by Sage - Annealed Train in H1 and L1 are above an FAR of 100/month and 1/day, respectively. Sage - Broad consistently seemed to produce a lot more triggers comparatively and at FARs below the lowest FAR observed in the Sage - Annealed Train for the H1 detector. However, both runs seem to perform similarly well in L1, with the annealed run producing the trigger with the lowest FAR. Although we expect the annealed run to have better glitch rejection capability (due to lower *bias due to limited sample representation* and, in turn, higher GW domain knowledge), we cannot conclude much from this comparison since this is a relatively small testing dataset. However, comparing our results with PyCBC, we can conclude that the performance is similar, except for PyCBC producing a trigger at an FAR of 2/month in L1.

We investigate a little further into the comparison of generated triggers for different types of glitches in the appendix (see Fig. 4.30 in appendix 4.9).

Effects of Non-stationary Noise

Since Sage and PyCBC seem to have similar glitch rejection capability, the primary cause for performance degradation in some Sage runs must be due to the differences in noise PSDs between the training and testing dataset. We investigate this possibility by comparing the results between three runs: Sage - D3 Metric, Sage - Limited and Sage - Broad. With this investigation, we test three hypotheses: [i] if there is no difference between the training and testing noise PSD distribution, the model should perform identically in both settings, [ii] a model trained with a broader noise PSD distribution will be more confident in predicting the presence of a GW signal compared to a model with a narrower distribution, and [iii] if the testing noise PSD distribution is out-of-distribution compared

to the training noise PSD distribution, the model might not be able to extrapolate to the unknown PSDs.

Parameter	Value
Approximant	IMRPhenomPv2
Mass 1	34.79
Mass 2	26.21
Right Ascension	3.11
Declination	-0.62
Inclination	0.83
Coalescence Phase	1.75
Polarisation	2.88
Time of Coalescence	11.09
Spin1x	-0.30
Spin1y	0.11
Spin1z	0.01
Spin2x	0.28
Spin2y	-0.25
Spin2z	-0.09
Chirp Mass	26.24
Mass Ratio	1.33

Table 4.3: Parameters used to produce the GW signal used in section 4.3.1. The SNR is rescaled to the required value, which dictates the rescaled distance parameter.

We create four different testing datasets (also known as a *testset* within the plots), each with 500,000 samples of noise for studying the effects of time-varying PSDs on SAGE. These datasets are described below:

Datasets 1 & 2: The first and second testing datasets contain simulated coloured Gaussian noise with PSDs from the same distribution as that described for the SAGE - D3 METRIC run. The seed for the training and testing datasets for SAGE - D3 METRIC runs determines the random choice of PSDs from the precomputed set when generating the noise samples. The first testing dataset uses the same seed as the training dataset and the second does not.

Datasets 3 & 4: The third and fourth datasets contain noise samples retrieved from O3a(Day 36 - 51) + O3b(Day 1 - 113) and O3a(Day 1 - 30) segments respectively. The inclusion or omission of O3a(Day 1 - 30) from the training or testing datasets significantly affects the variance of the noise PSDs as seen in Fig. 4.8, where the yellow (in H1) and purple (in L1) PSDs lie within this segment. The seed for these two datasets determines the random start time for each noise sample. The start time is uniformly distributed along the segments provided. The testing seed is not equal to the training seed for either of the datasets.

We then generate a single arbitrarily chosen GW signal with parameters provided in table 4.3 and

rescale its network optimal SNR across H1 and L1 to be a fixed value of 10. This signal is injected at the provided time of coalescence into half of the noise samples in each dataset, producing a 1:1 ratio between the signal and noise classes. A summary of the different runs made is provided in table 4.4. The rest of this section is dedicated to the analysis of Fig. 4.23.

Pointer	Model	Train PSD Distribution	Test PSD Distribution
Row 1 Red	Sage - D3 Metric	fixed set of O3 PSDs	same as train
Row 1 Blue	Sage - D3 Metric	fixed set of O3 PSDs	
Row 2 Orange	Sage - Limited	O3a(Day 36 - 51) + O3b(Day 1 - 113)	same as Sage - Limited
Row 2 Purple	Sage - Broad	O3a(Day 1 - 51) + O3b(Day 1 - 113)	
Row 3 Pink	Sage - Limited	O3a(Day 36 - 51) + O3b(Day 1 - 113)	O3a(Day 1 - 30)
Row 3 Green	Sage - Broad	O3a(Day 1 - 51) + O3b(Day 1 - 113)	

Table 4.4: Summary of run configurations used to make Fig. 4.23. The first column points to a histogram in a particular row of Fig. 4.23. The second column provides the name of the Sage run used to generate the histogram (details about the runs can be obtained from section 4.3.1). The third column describes the PSD distribution used to generate the training dataset samples. Sage - D3 Metric uses a fixed set of O3 PSDs as described in section 4.2.3. Sage - Broad and Sage - Limited both use noise samples from O3a(Day 36 - 51) + O3b(Day 1 - 113) for training, but the former uses PSDs from O3a(Day 1 - 30) to recolour the training samples, thus broadening the training PSD distribution. The last column describes the PSD distribution used to generate the testing dataset samples.

General description of Fig. 4.23: After evaluating a particular testing dataset with a Sage run, we histogram the network output (or ranking statistic) produced by the model for the signal and noise class. The plots on the left column correspond to the noise class and the ones on the right to the signal class. The histograms should resemble a noncentral chi-squared distribution for both classes given the behaviour of real noise PSDs. This is true for the signal class as they all contain the same signal with a fixed network optimal SNR and GW parameters. The intersection of the tails of the two classes contributes to type-I and type-II errors in the detection problem. The classification confidence of a model for a given dataset can be associated with the interclass distance (distance between average network outputs of the classes) and the reduction in type-I and type-II errors.

Row 1 in Fig. 4.23: The Sage - D3 Metric is evaluated on the first two testing datasets to produce the first row of results in Fig. 4.23. We see that the model produces similar results for both testing datasets (1 & 2). This corroborates the hypothesis that the model should perform identically in the training and testing settings, given that the PSD distribution is identical (ignoring non-Gaussian noise).

Row 2 in Fig. 4.23: Here, we test Sage - Limited and Sage - Broad on the third dataset, with noise retrieved from O3a(Day 36 - 51) + O3b(Day 1 - 113). Our naive expectation might be that Sage - Limited, which is trained on the same noise segment, should perform better than Sage - Broad as it is trained on precisely the required PSD realisations. However, we observe that Sage - Broad is more

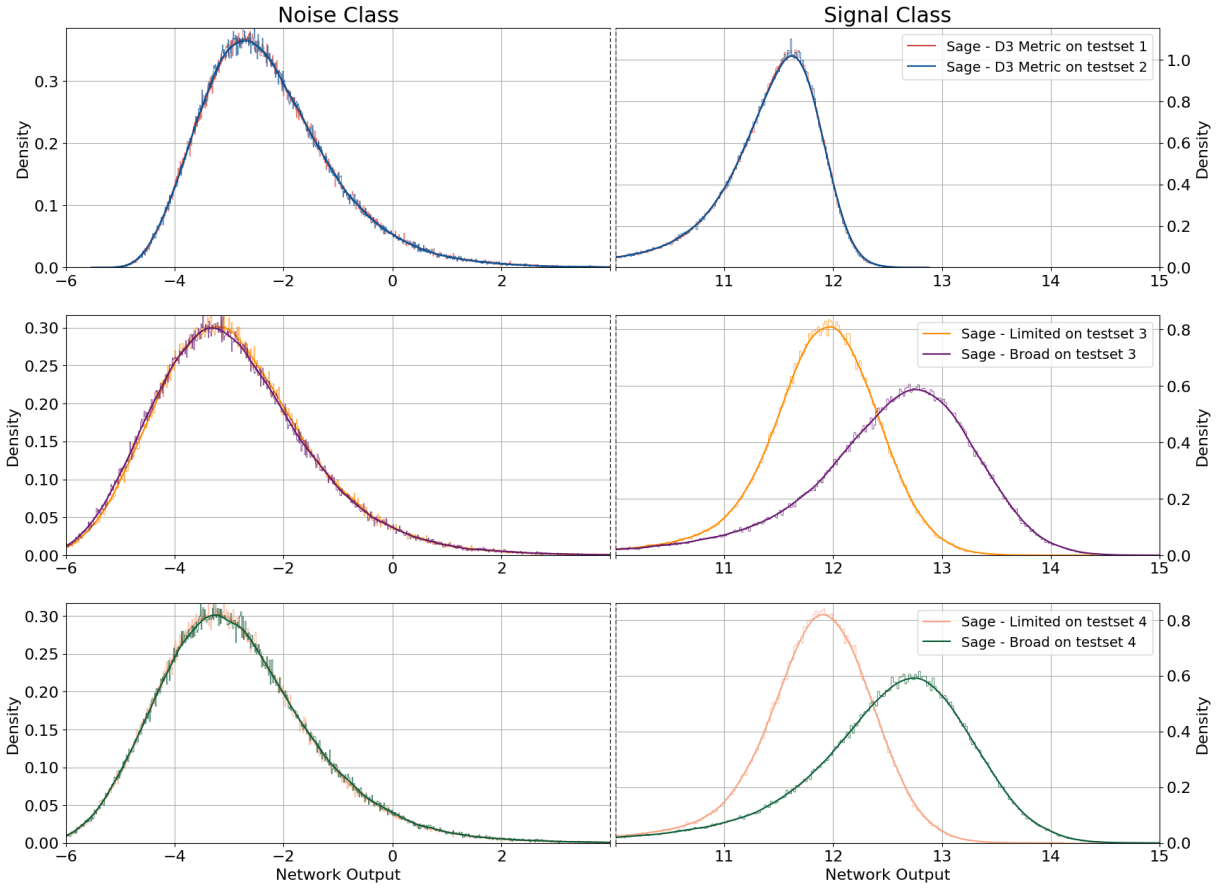


Figure 4.23: Normalised histograms of ranking statistics generated by Sage runs for samples from the signal and noise class on a particular testing dataset of 500,000 samples (1:1 ratio between the signal and noise class). A kernel density estimate of each histogram is plotted over each histogram. The left column corresponds to the noise class and the right column corresponds to the signal class. We truncate both histograms in the region of overlap between the two classes for plotting purposes.

confident in classifying a signal injected into the same noise realisations than Sage - Limited, corroborated by the larger interclass distance. Moreover, Sage - Broad also seems to perform slightly better when classifying pure noise realisations from this dataset as seen on the left plot. This corroborates the hypothesis that a model trained with a broader noise PSD distribution will be more confident in predicting the presence of a GW signal compared to a model with a narrower distribution.

Reason for classification confidence: This might not be simply due to a broader PSD distribution, but rather the inclusion of more challenging noise PSD realisations. Resolving the sources of type-I and type-II errors might lead to more confident models. The outcome of better generalisation due to the inclusion of harder training samples has been observed in many previous works [264, 265, 266], and is thus a reasonable conclusion. However, we also found that simply decreasing the lower bound of the training SNR distribution did not yield the same results. Thus, the improvement in confidence is likely due to better generalisation on noise PSDs due to addressing the *bias due to lack of variation*.

Row 3 in Fig. 4.23: We repeat the same procedure as Row 2, but using the fourth dataset, with

noise retrieved from O3a(Day 1 - 30). The behaviour of the two networks is similar to that seen in Row 2, with the exception that both networks are now equally confident about the noise class outputs. Although Sage - Broad performed better than Sage - Limited in testing dataset 3 for the noise class, this behaviour did not transfer to PSDs in testing dataset 4. The similar performance of the two models on the noise class indicates that Sage - Limited is able to successfully extrapolate to unknown PSDs with higher power. The unchanged behaviour with the signal class in Row 2 and Row 3 signifies that neither network is overfitting.

Is real noise better than Gaussian noise for training?: Comparing Row 1 to Rows 2 & 3, we see that models trained on real noise (Sage - Limited & Broad) have higher interclass distance than the model trained on simulated coloured Gaussian noise (Sage - D3 Metric). This is due to two reasons: [i] the precomputed PSDs used for Sage - D3 Metric does not vary significantly, thus leading to nearly white noise after whitening with the median PSD, [ii] absence of non-Gaussian noise artefacts. Previous works [267, 268] have established that deep networks learn patterns easily and require less network capacity to do so, but white Gaussian noise (or the lack of patterns) is not learnt but memorised and takes a large amount of network capacity. Moreover, trying to learn Gaussian noise was shown to lead to complex decision boundaries and required longer training times. Thus, the lack of patterns in noise samples hinders learning.

Implications: Data pre-processing procedures that effectively whiten the noise samples (by using an estimated PSD or glitch removal), counter-intuitively, might hinder learning for the GW detection problem. Using a broader training distribution for noise PSD might allow the model to understand and interpolate the PSD space rather than memorise it. Interpolation, in turn, leads to better generalisation and reduces the need to use larger training datasets (data efficiency).

4.3.2 Comparison with Other ML Pipelines

In this section, we compare our results with recent ML pipelines that have evaluated their work on the datasets provided in MLGWSC-1 [204]. We do a more in-depth comparison with AresGW [209] and Aframe [212] given that they exhibit the highest detection sensitivity among the pipelines considered.

Comparison with AresGW

Data/architecture differences between AresGW and Sage: The AresGW architecture is a 1D ResNet-54 trained using the curriculum learning strategy [209, 269]. For training, they use a 1-second sample length, 12 days of real O3a noise data and 38,000 different waveforms (each injected into 19 different background segments, leading to 740,000 different training samples). They use the same training mass distribution to generate waveforms as Sage - Broad.

Figure 4.24 shows the distribution of detections across different parameters and compares the number of detections between Sage - Broad and AresGW at an FAR of 1/month. We see that 2814

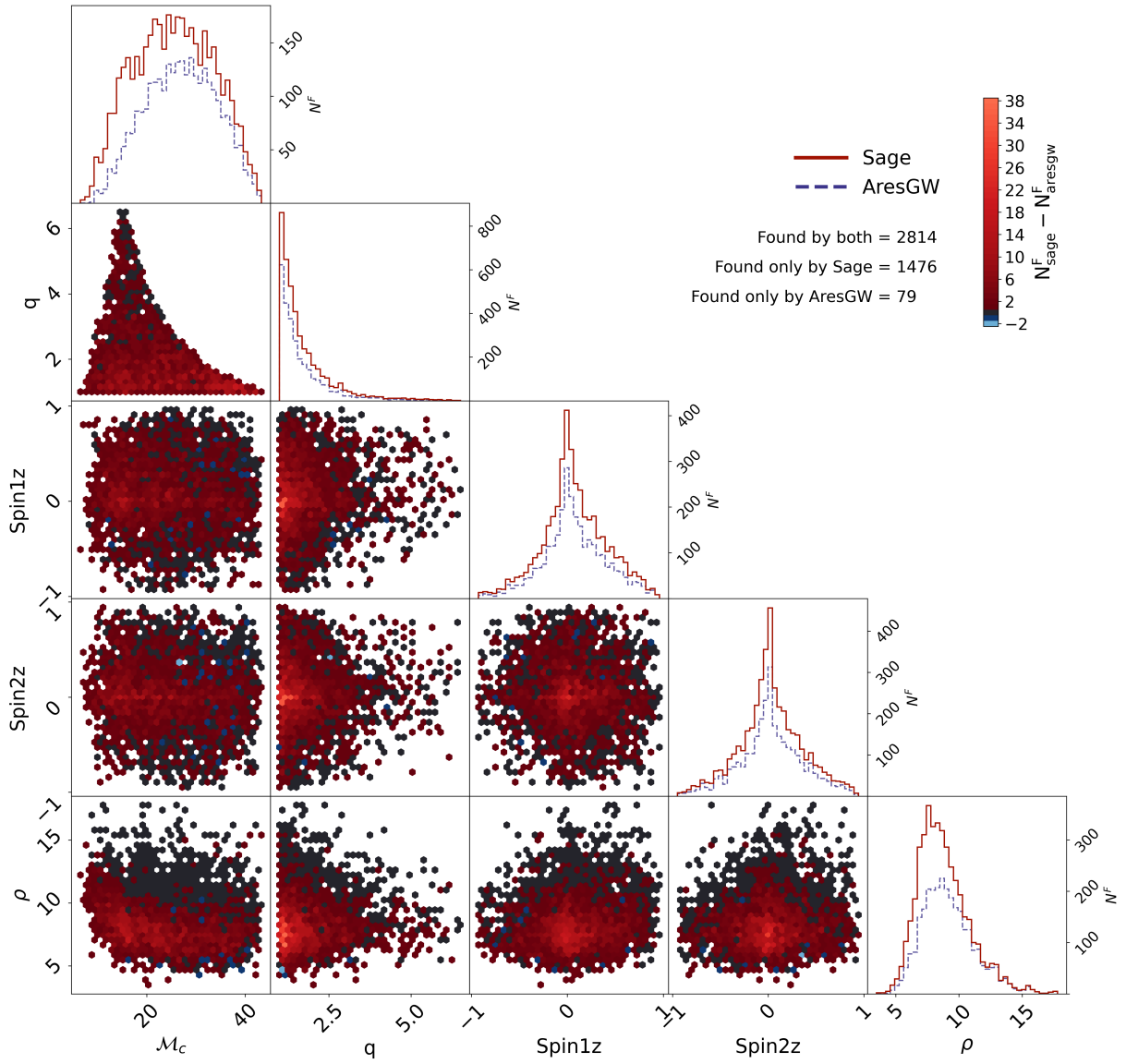


Figure 4.24: 1D and 2D histograms of detected signals for different GW parameters at an FAR of 1 per month, comparing the best Sage - Broad run and AresGW. We chose GW parameters where we expected to see learning bias. The diagonal subplots contain 1D histograms of the detected signals for a given parameter. The 2D histograms are hexagonal binned plots, where each bin is coloured using $N^F_{\text{sage}} - N^F_{\text{aresgw}}$ where N^F_{sage} and N^F_{aresgw} is the number of signals detected by Sage and AresGW respectively. We use a divergent colour map that is set to be black when $N^F_{\text{sage}} - N^F_{\text{aresgw}}$ is zero, is progressively brighter in red along the positive axis, and is progressively brighter in blue along the negative axis. So, red and blue patches can be attributed to regions where Sage and AresGW detect more signals, respectively.

signals were found by both pipelines, 1476 signals were found only by Sage and 79 signals were found only by AresGW. This is equivalent to an increase of $\approx$ 48.29% in the number of signals detected at that FAR. Sage - Broad particularly outperforms its counterpart in the lower chirp mass regime.

The AresGW pipeline was enhanced in [210] and included the following major changes: using

35 days of real O3a noise instead of 12, a low-pass filter at 350 Hz for all samples, and changes to how the triggers are generated and aggregated during testing. The relative difference in detection sensitivity between SAGE and enhanced-ARESGW should not change by much due to the additional noise data, since the latter does not mitigate the biases in section 4.1.3. This statement holds even if enhanced-ARESGW were trained with 160 days of real O3 noise, 80 million training samples with a length of 12 seconds and the same training strategies as SAGE. Justification for these statements is provided with sufficient detail in ablation study II and III (see section 4.3.3).

Comparison with AFRAME

Data/architecture differences between AFRAME and SAGE: The AFRAME architecture is a 1D ResNet-54 trained using the curriculum learning strategy [212, 269]. For training, they use a 1.5-second sample length, $\approx$ 4.5 days of real O3 noise data and 75,000 different waveforms (each injected into a random noise segment obtained from the $\approx$ 4.5 days of real noise). A key area of difference that sets apart AFRAME from the other pipelines we compare to is the training mass distribution. They use $p(m_1) \propto m_1^{-2.35}$ and $p(m_2|m_1) \propto m_2$ as per GWTC-3 [5], which places a higher emphasis on lower chirp mass or longer duration signals. Moreover, they train on a mass range between 5 and 100 $M_\odot$.

Figure 4.25 shows the distribution of detections across different parameters and compares the number of detections between SAGE - BROAD and AFRAME[7] at an FAR of 1/month. We see that 2837 signals were found by both pipelines, 1453 signals were found only by SAGE and 71 signals were found only by AFRAME. The number of signals detection is similar to ARESGW, but the distribution is expectedly different given the different training mass distribution (see Fig. 4.24 for reference). The 1D histogram of chirp mass in Fig. 4.25 shows that AFRAME addresses the *bias due to limited sample representation* of low chirp mass signals via the training mass distribution, thus detecting more longer duration signals than ARESGW. The 1D histogram of optimal SNR is similar to that of ARESGW, indicating a limited capacity to effectively detect low SNR signals.

For a more meaningful comparison, we must train SAGE on the $p(m_1) \propto m_1^{-2.35}$ distribution, which will likely detect more long duration signals than SAGE - BROAD, as indicated by the SAGE - ANNEALED TRAIN run in figure 4.17. As AFRAME uses a 1.5-second sample length, they do not address the *bias due to insufficient information*, and a testing dataset that contains significantly more long-duration signals will enunciate this bias. We leave this analysis to the future.

Consolidated comparison: We consolidate the comparison of the detection performance for different pipelines in table 4.5 and shows the sensitive distance metric of multiple ML pipelines and PYCBC at different FARs in Fig. 4.26. We see an increase of $\approx$ 4.1% in sensitive distance compared to ARESGW at an FAR of 1 per month. For the sake of completeness, we provide the sensitive distance curve of AFRAME in Fig. 4.26. The disadvantage of using a parameter-dependent metric is clearly ev-

[7] Results obtained via private communication.

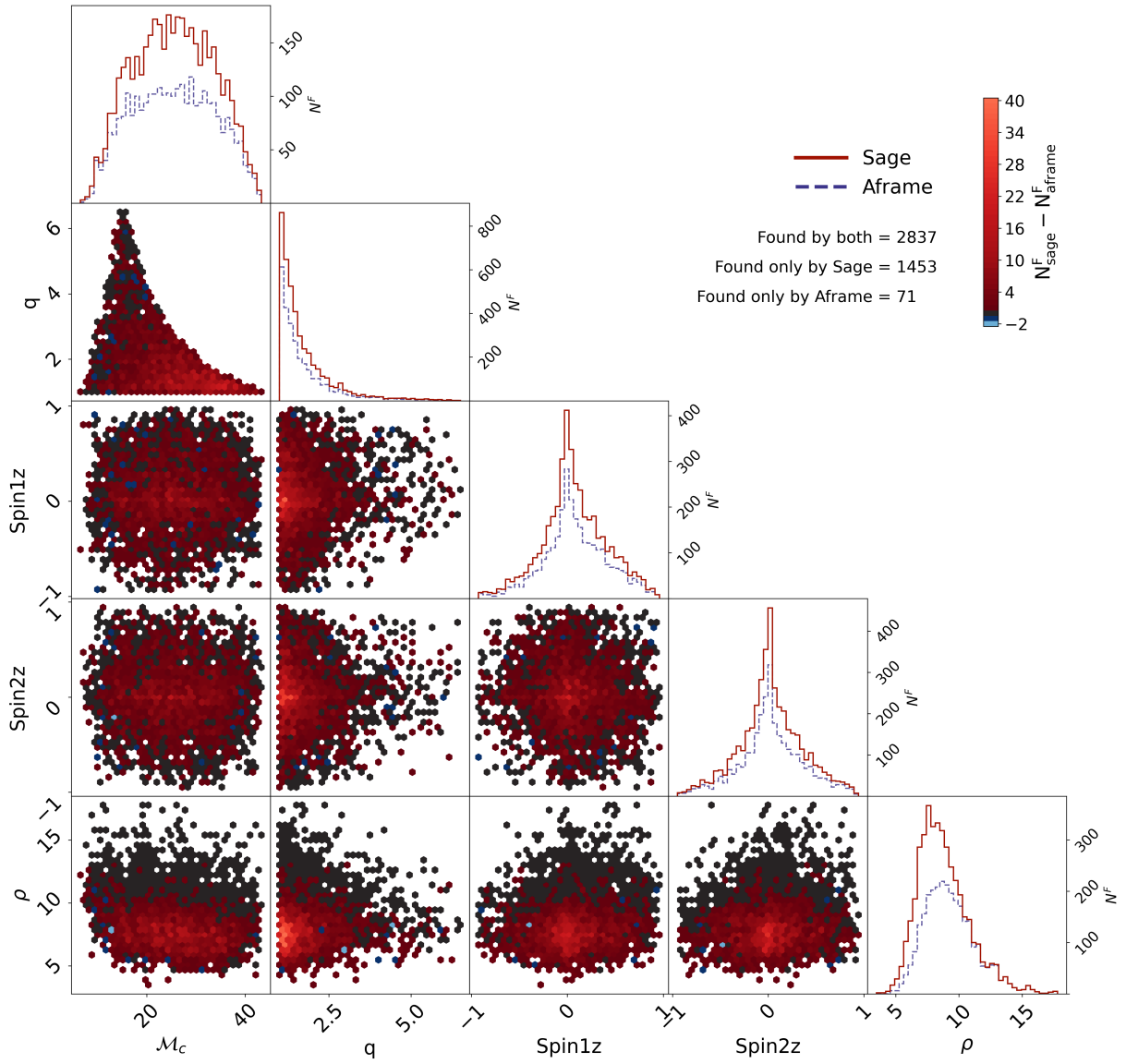


Figure 4.25: 1D and 2D histograms of detected signals for different GW parameters at an FAR of 1 per month, comparing the best Sage - Broad run and Aframe. We chose GW parameters where we expected to see learning bias. The diagonal subplots contain 1D histograms of the detected signals for a given parameter. The 2D histograms are hexagonal binned plots, where each bin is coloured using $N^F_{\text{sage}} - N^F_{\text{aframe}}$ where N^F_{sage} and N^F_{aframe} is the number of signals detected by Sage and Aframe respectively. We use a divergent colour map that is set to be black when $N^F_{\text{sage}} - N^F_{\text{aframe}}$ is zero, is progressively brighter in red along the positive axis, and is progressively brighter in blue along the negative axis. So, red and blue patches can be attributed to regions where Sage and Aframe detect more signals, respectively.

ident when comparing our Sage - Annealed Train run with AresGW. Although our run finds $\approx 15.5\%$ more signals at an FAR of 1 per month, it scores $\approx 3.5\%$ less on the sensitive distance metric, due to the chirp mass dependence.

Pipeline	FAR 100/month	FAR 10/month	FAR 1/month	$N^F_{10/\text{month}}$	$N^F_{1/\text{month}}$
PyCBC [204]	1722.008	1607.307	1538.768	4364	3857
CWB [204]	1460.617	1440.848	1364.195	–	–
Aframe* [212]	1708.237*	1570.450*	1441.291*	4252*	2908*
AresGW [209]	1817.074	1632.455	1573.914	3878	2893
TPI-FSU Jena (2024) [186]	1583.625	1424.370	1316.161	2932	1609
MFCNN [204]	1269.034	984.995	587.930	444	89
Sage - Broad	**1858.554**	**1722.352**	**1638.204**	**4760**	**4290**
Sage - Limited	1851.586	1710.573	1553.479	4731	3306
Sage - D4 Metric	1758.124	1609.436	1405.947	4179	2316
Sage - Annealed Train	1748.338	1613.982	1518.557	4280	3341

Table 4.5: Summary of sensitive distance measures and number of signals detected in the realistic testing dataset for different pipelines and Sage runs, for different FARs. $N^F_{1/\text{month}}$ and $N^F_{10/\text{month}}$ are the number of found injections at an FAR of 1 per month and 10 per month respectively. The Aframe results are marked with an asterisk (*) as their pipeline is not optimised for the D4 mass distribution in MLGWSC-1 [204].

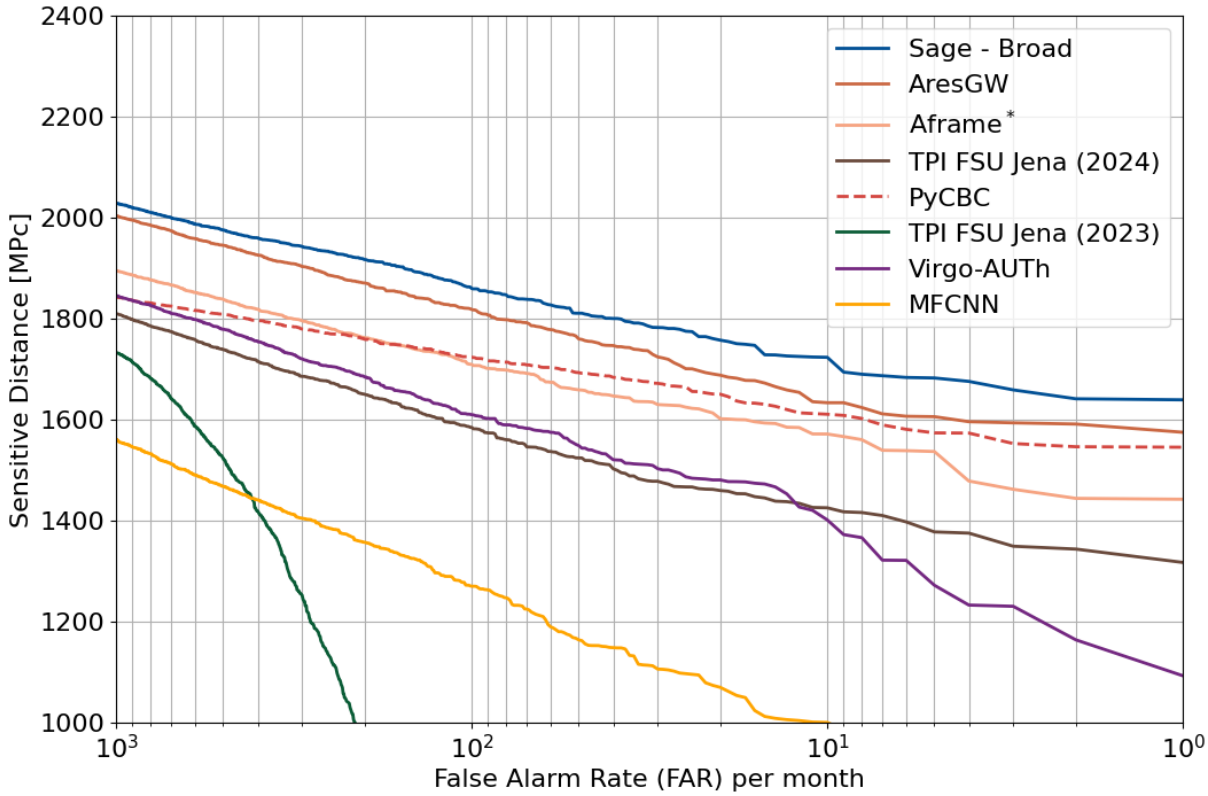


Figure 4.26: The sensitive distance metric as a function of FAR per month for Sage - Broad, other ML pipelines and PyCBC on the D4 testing dataset. The AresGW results were obtained from [209], Aframe and TPI FSU Jena (2024) results via private communication, and all other results were obtained from [204]. The Aframe results are marked with an asterisk (*) as their pipeline is not optimised for the D4 mass distribution in MLGWSC-1 [204].

4.3.3 Ablation Study on Sage

Although a common practice for pipelines in the ML literature [270], ablation studies are generally not performed in the ML for GW literature. Given the diversity of methods that can be used in ML,

it is imperative to prove the necessity of important parts of our methodology.

Although we have shown proof and explanations for the performance gains seen in SAGE compared to previous ML pipelines, the source of this performance is not apparent. We try to understand the contribution of some of the major components described in the methodology (see section 4.2) by systematically removing them and evaluating SAGE's performance on the same realistic testing dataset. The methodology used throughout the ablation study is the same as that seen in section 4.2, and any changes made are described in each study. The training seeds are fixed for all studies, so they use the same signal and noise realisations for all epochs. The studies that use a limited dataset are trained on the same pre-made dataset. All comparison plots are made at an FAR of 1/month. This ablation study is not exhaustive and only targets some of the major differences that led to significant performance gains.

Study I - Decreased Noise and Signal Realisations: For the first study, we removed several major components of SAGE that were meant to increase the variance in the signal and noise classes during training. For the signal class, we generate the plus and cross polarisations of 10^6 GW signals with the same generation settings as SAGE - BROAD. Each iteration of every epoch is given a unique seed, which is used to set the right ascension, declination, and polarisation for the antenna pattern before calculating the strain of the waveform as measured by the detector. A time shift is applied for the given detector relative to the assumed geocentric frame to obtain the strain for each of the considered detectors. We then rescale the network SNR of the system by drawing SNR values from a uniform distribution within the bounds [5, 15]. This procedure limits the GW parameter augmentation to the four parameters mentioned above. All other parameters have a fixed 10^6 realisations throughout the training procedure. For the noise class, we retrieve 10^6 samples from O3a(Day 31 - 51) noise segment (with sample overlap) and do not apply any noise recolouring.

Figure 4.27(a) shows the chirp mass histogram of the number of found injections compared to PYCBC at an FAR of 1/month. It also shows reference histograms for the PYCBC and best SAGE - BROAD results on this dataset, taken from Fig. 4.20. We only use chirp mass here since it is the parameter along which we observe the most bias. Although we use the same network architecture and fix all other settings, the detection performance is significantly reduced. We detected signs of overfitting during training, likely because many of the training hyperparameters were tuned for the original setting and higher variance in the noise and signal classes.

We fine-tune the hyperparameters of the network to work on the reduced dataset for a fairer comparison. Although the SAGE architecture reduces some of the biases, the distribution of the parameters, the amount of variation and the training strategy are paramount to achieving a good detection performance. Figure 4.27(b) shows an increase in performance due to the fine-tuning but does not compare well with our best results. All ablation studies henceforth have been fine-tuned to account for the corresponding changes made.

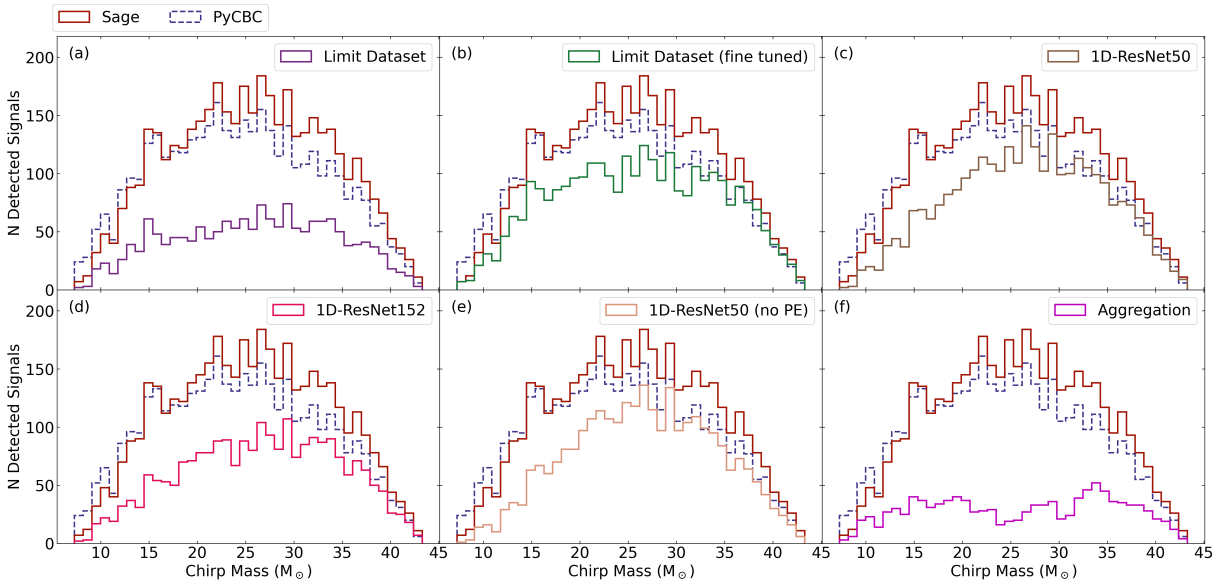


Figure 4.27: 1D chirp mass histograms of detected signals for all ablation studies compared with Sage - Broad and PyCBC at an FAR of 1 per month. Panels (a) and (b) correspond to ablation study I (Decreased Noise and Signal Realisations), panel (c) shows study II (Using a Simpler Network Architecture), panel (d) shows study III (Using a Simple (but deeper) Network Architecture), panel (e) shows study IV (Abandon Regularisation), and panel (f) shows study V (Effects of Aggregation).

Study II - Using a Simpler Network Architecture: In this study, we use a 1D ResNet-50 [243], which is comparable in size to the networks used in [212, 209, 210]. Due to the lack of relevant inductive biases (a network architecture or training strategy particularly designed for the problem) in the standard 1D ResNet, we observe in Fig. 4.27(c) that the detection sensitivity has significantly degraded compared to Sage. This study indicates that the architectures used in AresGW [209], enhanced-AresGW [210] and Aframe [212] would not experience a significant increase in detection sensitivity even if they were trained on 160 days of real O3 noise, 80 million training samples of length 12-seconds and the same training strategies as Sage. This empirically justifies the requirement of appropriate inductive biases.

Study III - Using a Simple (but deeper) Network Architecture: For the third study, we use a 1D ResNet-152 [243]. We chose this architecture for two reasons: [i] This is a fairer comparison to Sage as the 1D ResNet-50 is significantly smaller in terms of the number of trainable parameters, and [ii] deeper networks are known to learn long term dependencies better [226] which also offers an advantage over the 1D ResNet-50. However, we see in Fig. 4.27(d) that the 1D ResNet-152 performs worse than the much shallower 1D ResNet-50. This is because we fixed the number of training iterations to be the same for all ablation studies. Bigger networks are typically less data efficient than shallower networks [271], which explains the slightly worse performance in comparison to a 1D ResNet-50. However, this is *only* true when the proper inductive biases are not used. The 1D ResNet-152 and Sage have nearly the same number of trainable parameters, with the former being slightly larger. The reduction in performance seen in Fig. 4.27(d) is thus purely due to the lack of proper inductive bias. The Sage architecture allows for significantly higher data efficiency.

Study IV - Abandon Regularisation: All previous ablation studies have predicted a point estimate of the chirp mass and time of coalescence of the supposed GW signal in each sample. Section 4.2.7 of the methodology describes this point estimate as a regularisation method that provides an additional inductive bias to distinguish between the signal and noise classes. We repeat study II but remove this regularisation and see in Fig. 4.27(e) that it leads to a comparative reduction in detection performance with Fig. 4.27(c). Since the training seeds were exactly the same between studies II and IV, we can safely conclude that the difference is purely due to the lack of this regularisation feature.

Study V - Effects of Aggregation: For the final study, we emphasise the aggregating effects of supervised learning by using an architecture that is not well suited to handling glitches in the data. Instead of our CBAM ResNet-50 backend classifier, we use a Res2Net [272], a multiscale backbone to the standard ResNet architecture. We see that in Fig. 4.27(f), this relatively minor change led to a drastic reduction in detection performance. We found that certain architectures led to a completely invariant ranking statistic for all signals above a certain threshold in SNR (let us name this T_{snr}). This threshold was typically present close to the signal-noise class decision boundary. Say we have 1 month of unseen testing data. The presence of a single unseen difficult-to-classify glitch in this dataset could generate a trigger with a ranking statistic close to T_{snr} and instantly reduce the detection sensitivity at an FAR of 1 per month. This sort of *aggregation* of triggers close to the decision boundary can be mildly alleviated when using larger networks, and significantly mitigated when using networks with proper inductive biases.

4.4 Running Sage on Live Data

Section 4.3.1 (specifically subsection about the injection study, glitch rejection capability, and the effects of PSD) prove that Sage is able to achieve highly sensitive detection performance with good glitch rejection capability and limited extrapolation capability with noise PSDs. Thus, we believe that Sage can handle real-time detection of GW alongside other existing production-level pipelines. We dedicate this section to detail minor modifications to Sage that will allow for its usage on an unknown live data feed.

Say we want to run Sage on the fifth observing run (O5). Before running a search on live data, we must first train the model with noise realisations that resemble the new data, to bridge the gap due to different noise PSDs. We can obtain a small amount of noise PSDs from the new data and recolour previously known noise realisations from O3 and O4, giving us a large dataset. This data can then be used to fine-tune Sage over 1 or 2 epochs, which allows it to adapt to the new noise PSDs. To achieve a low enough FAR, we can modify Sage in one of the following ways:

Method 1: Our runs in the injection study assume a known background dataset to estimate the FAR of an event. In the case of live data, we can retain the current multi-detector architecture of Sage and perform a time-slide analysis [89] of the strain data with limited data to achieve low FARs. To perform

a time-sliding technique, the model must first be trained on a dataset that includes a signal in one detector but not the other. This will incentivise the model to produce appropriate null triggers when operating on time-slid data. However, this will use unnecessary learning capacity (since time-slid data is not of interest astrophysically) and will likely increase training times. For each time-slid data sample, the model must run in evaluation mode; to achieve an FAR of 1 per year, we must evaluate the model on 1 year of time-slid data. The primary disadvantage of this method is that going to very low FARs might become infeasible given evaluation times.

Method 2: We can run Sage on a single-detector coincident analysis setting and perform time-sliding. Background estimation can be performed similar to PyCBC Live, by storing a rolling buffer of detector triggers and performing the time-sliding technique to estimate the FAR. Shifting the triggers beyond the light travel time between detectors will negate some background triggers via non-physical coincidences. Contrary to the first method, this will lead to highly efficient time sliding of *triggers* instead of the strain itself. However, a single-detector ML model will not be able to take advantage of some of the architectural features introduced in section 4.2.7, like channel-wise attention in the backend classifier. This might reduce detection efficiency.

Running Sage on O3: Say we want to conduct an offline search on O3 data with Sage. Since Sage is trained on the entirety of O3 noise data (as labelled by current matched-filtering searches), there is the natural concern that our training dataset possibly includes as-of-yet undetected signals in the noise class. However, given that the ratio between the volume of undetected signals to genuine noise is negligible, the following biases will prevent the model from learning undetected signals as noise: [i] the *bias due to limited sample representation* and [ii] the significant imbalance between noise and signal features in the noise class (*bias due to limited feature representation*).

4.5 Limitations of Sage

Persistent biases: Although we tried to address the biases mentioned in section 4.1, our best results in Fig. 4.20 are still marginally biased against lower chirp masses. An attempt to alleviate all mentioned biases, as seen for the Sage - Annealed Train run in Fig. 4.17, leads to an $\approx$ 13.3% decrease in the number of signals detected at an FAR of 1 per month when compared to PyCBC.

Evaluation speed: A larger testing dataset would allow for performance evaluations at lower FARs. However, even at the high evaluation speed of the analysis, the current coherent strategy is limited due to the issues raised in section 4.4 *method 1*. Since evaluating on 1 month of testing data takes $\approx$ 16 hours, achieving an FAR of 1 in 10 years, for example, would require $\approx$ 80 days. Even if a model had an evaluation speed of 30 minutes for 1 month of testing data, it would still take 2.5 days. Thus, section 4.4 *method 1* becomes quickly impractical.

Mild train-test overlap: The best result we present uses noise PSDs obtained from the testing

dataset to recolour the training noise realisations. The resulting recoloured noise realisation is technically new since the testing data PSD is only an estimate and whitened noise realisation (before being recoloured) is different from testing data. Although the network does not exhibit any signs of overfitting, this method still introduces mild train-test overlap. It was shown in section 4.3.1 that the performance gain was *not* due to this overlap, but the result of a broader distribution of noise PSDs in the training dataset. We can thus alleviate the mild train-test overlap by generating artificial PSDs using a normalising flow [273] and retraining Sage once before an online search with some representative PSDs. It is possible to retrain an ML model such that it does not need further retraining throughout an observing run, as evidenced by [274].

Small testing dataset: The number of glitches, although substantial ($O(10^4)$) within the month of O3a testing noise, is not sufficient to fully analyse the glitch rejection capability of Sage. Moreover, there is no guarantee that the glitch profiles of *all* glitch types were fully learnt given the imbalance of data for different glitch types (see Fig. 4.30 in appendix 4.9).

Need for large training dataset: Our current work requires a large training dataset in order to reduce biases. Providing sufficient variation to all GW parameters and including all known noise realisations will prove difficult in the future. Thus, we have to work toward creating an information-efficient pipeline that is able to adapt to unforeseen changes in noise characteristics without the need for retraining.

4.6 Conclusion

We introduce Sage, an ML-based BBH detection pipeline, that is shown to be capable of detecting $\approx$ 11.2% and $\approx$ 48.29% more signals than the benchmark PyCBC submission in [204] and the AresGW results in [209] for the Machine Learning Gravitational Wave Search Challenge (MLGWSC-1) injection study, respectively, at an FAR of one per month in O3a noise. We achieve this performance by identifying a non-exhaustive set of 11 interconnected biases that are present in the usage of ML for the GW detection problem and create mitigation tactics and training strategies to address them concurrently. We prove that Sage is robust to OOD variation in noise power spectral densities and can reject non-Gaussian transient noise artefacts effectively. Explanations and rationale have been provided for all method choices, and an ablation study was conducted to provide evidence for our claims. Our work addresses the need for interpretability in the application of ML to GW detection.

The biases, mitigation tactics and training strategies that we provide are general, and the salient ideas are transferable to the detection of other sources of GWs and to other domains of GW data analysis. We re-iterate some of our primary empirical contributions for the application of ML to the GW detection problem:

Negating the need for PSD estimation: Contrary to previous ML-based detection pipelines oper-

ating on real detector noise [209, 212, 186, 210], we show that SAGE does not need to use a PSD estimate for whitening any input sample (see section 4.2, *Data Transformation*). We use a single approximate PSD instead for all samples.

Mitigating biases might aid in glitch rejection: In section 4.3.1, we obtain all the triggers generated by SAGE and PYCBC during glitch events in one month of O3a noise and prove that SAGE is capable of effectively rejecting glitches. We also argue about the potential benefits of mitigating biases for glitch rejection.

OOD PSDs are good for classification: In section 4.3.1, we show that using a broader range of noise PSDs for the training data, which could even be OOD to the testing dataset, greatly aids in detection performance.

Real noise is better than Gaussian noise for training: We show in section 4.3.1 that models trained with real detector noise produce more confident classifiers than models trained with simulated coloured Gaussian noise.

Parameter efficiency via inductive biases: Via an ablation study (see section 4.3.3) we show that a model that is larger than SAGE in terms of number of trainable parameters is not able to achieve nearly the same detection sensitivity as SAGE, *if* it does not have the proper inductive biases.

The investigation of machine learning biases has shown that, with appropriate mitigation strategies, reliable and robust detection pipelines can be constructed. Building on these results, we now focus on applying the developed methods to real data from the third observing run to evaluate their performance in practical GW searches.

4.7 Appendix A: Testing Dataset Parameter Distributions

Figure 4.28 shows the gravitational wave parameter distributions for the D4 testing dataset introduced in the MLGWSC-1 [204]. This was generated using the data generation `ini` file provided in the MLGWSC-1 GitHub repository for use with PYCBC (see this https url).

4.8 Appendix B: Sage-Broad Detection Sensitivity

Figure 4.29 shows the 1D histograms of detected events along different gravitational wave parameters at an FAR of 1 per month for SAGE - BROAD. Figures 4.20, 4.24, and 4.25 show the same for a limited number of parameters - chirp mass, mass ratio, spin-1z, spin-2z and optimal network SNR. Here, we provide a full comparison of SAGE - BROAD and PYCBC.

4.9 Appendix C: Network Behaviour for Different Glitch Types

We investigated a little further into the comparison of generated triggers for different types of glitches (see Fig. 4.30). We arrived at three major observations:

Observation 1: The training component mass distribution and its related learning biases determine the glitch types that the ML model finds most difficult to classify.

Observation 2: The Sage - Annealed Train run typically produced fewer glitch triggers between an FAR of 10^4 per month to 1 per month, when compared to Sage - Broad.

Observation 3: We did not observe any noticeable correlation between the difficult-to-classify glitch types and their number of occurrences in the training data. In particular, glitch types that were not significantly present in the training dataset were not any more difficult to classify than the glitch types that were present in abundance. This might be an artefact of a limited testing dataset.

4.10 Appendix D: Parameter Point Estimate

Figure 4.31 shows parity plots of true vs. predicted chirp mass and coalescence time for the validation dataset from Sage - Annealed Train and Sage - Broad, evaluated at the epoch with the lowest validation loss. These point estimates are included primarily to provide an auxiliary inductive bias during training, while also serving to passively diagnose parameter-specific training biases.

Diagnosing Biases: For both Sage - Annealed Train (Fig. 4.31, top) and Sage - Broad (Fig. 4.31, bottom), the chirp mass is typically underestimated near the upper end of its range, even at high network SNR. Two possible causes are: [i] insufficient representation of high chirp mass signals in the training data, and [ii] model limitations in capturing short-duration waveform morphology. The second is unlikely, as it would produce random errors rather than systematic bias. Supporting the first explanation, Sage - Broad, which trains on $U(m_1, m_2)$, shows reduced bias compared to Sage - Annealed Train, which trains on $U(\tau_0, \tau_3)$ and is more underrepresented at high chirp masses.

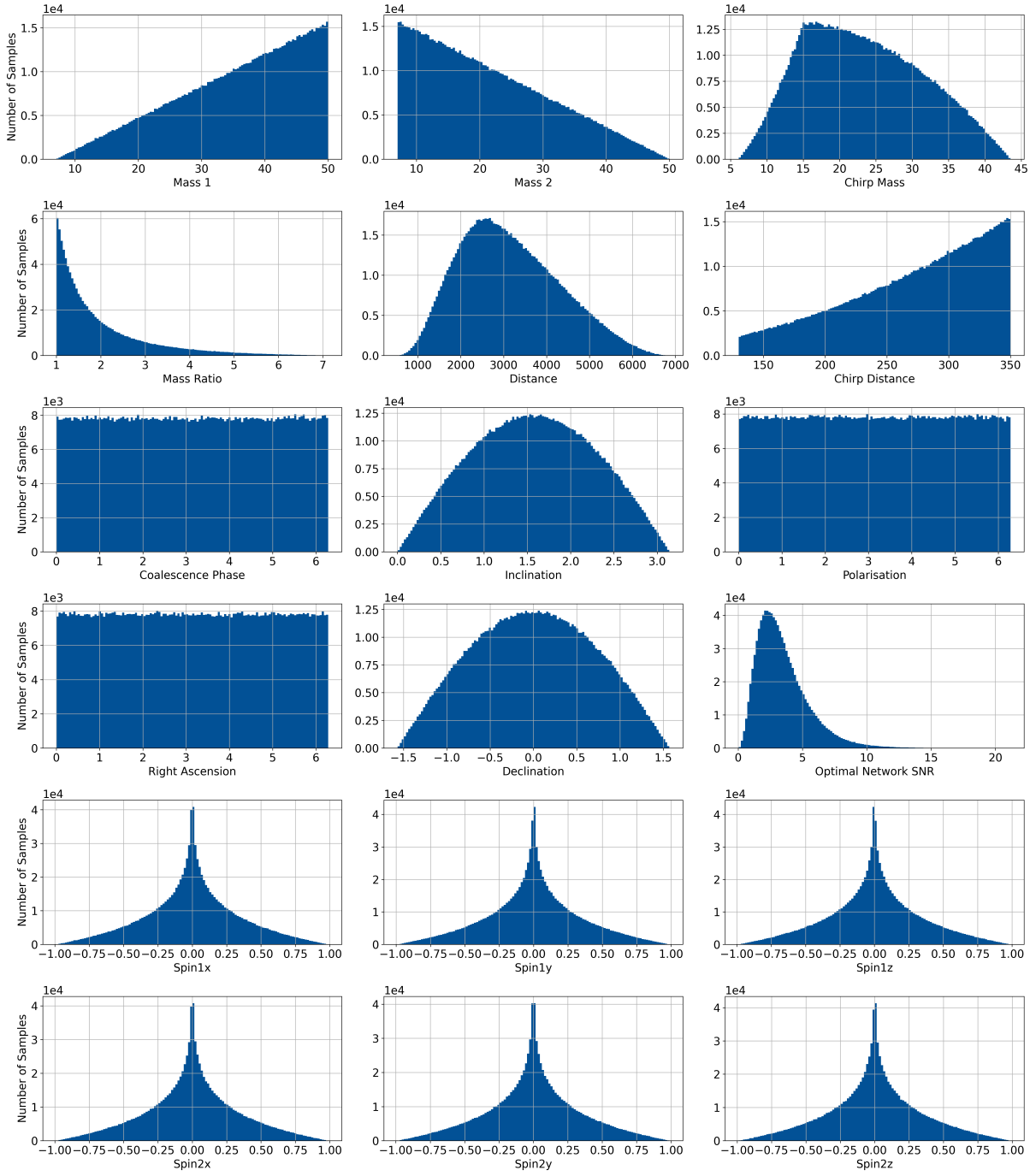


Figure 4.28: 1D histograms of GW parameters showing the parameter distributions based on the D4 testing dataset generated from 10^6 samples. The distributions are fully defined in table 4.1.

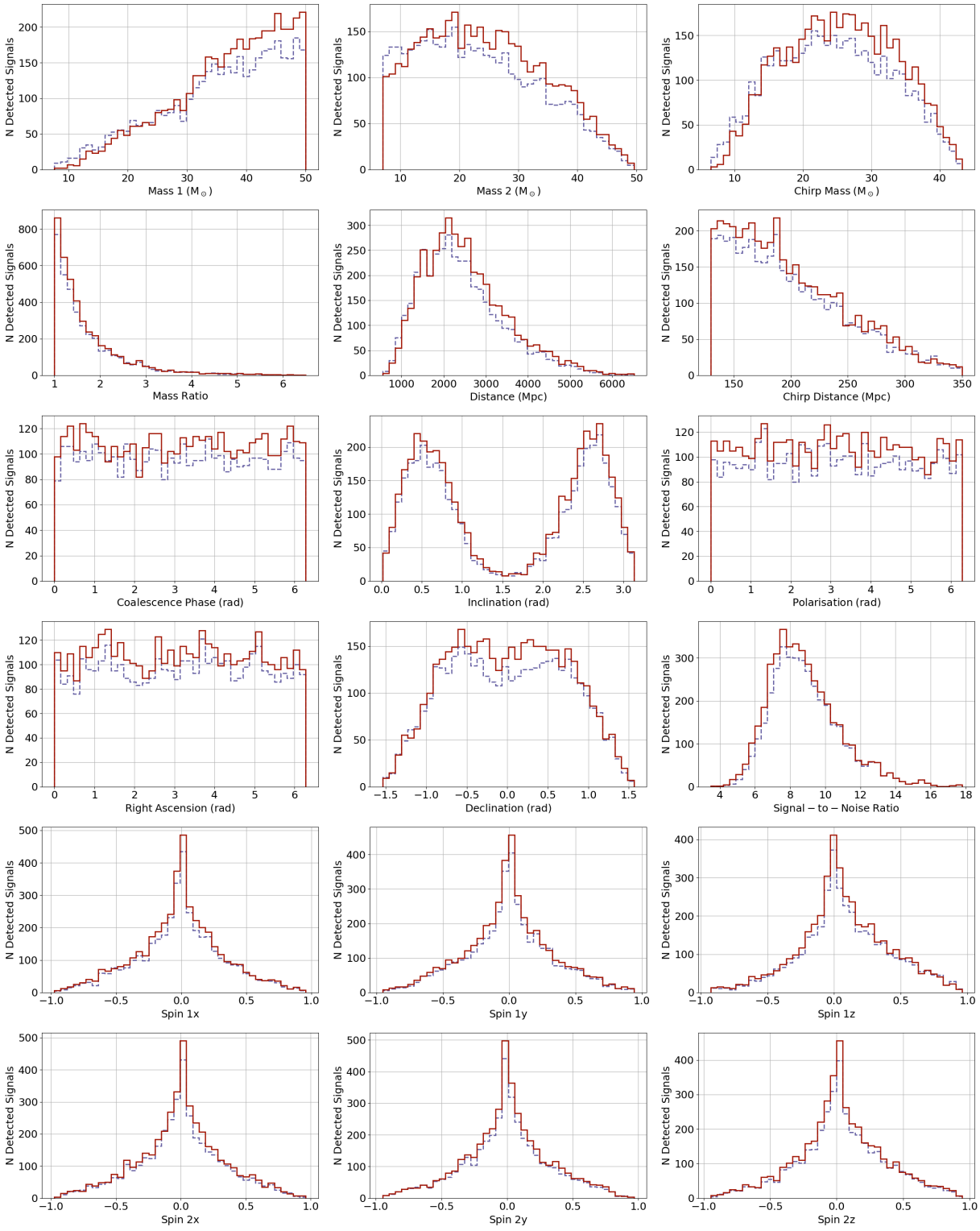


Figure 4.29: 1D chirp mass histograms of different GW parameters showing the number of detections made by SAGE - BROAD and PYCBC at an FAR of 1 per month for the D4 testing dataset.

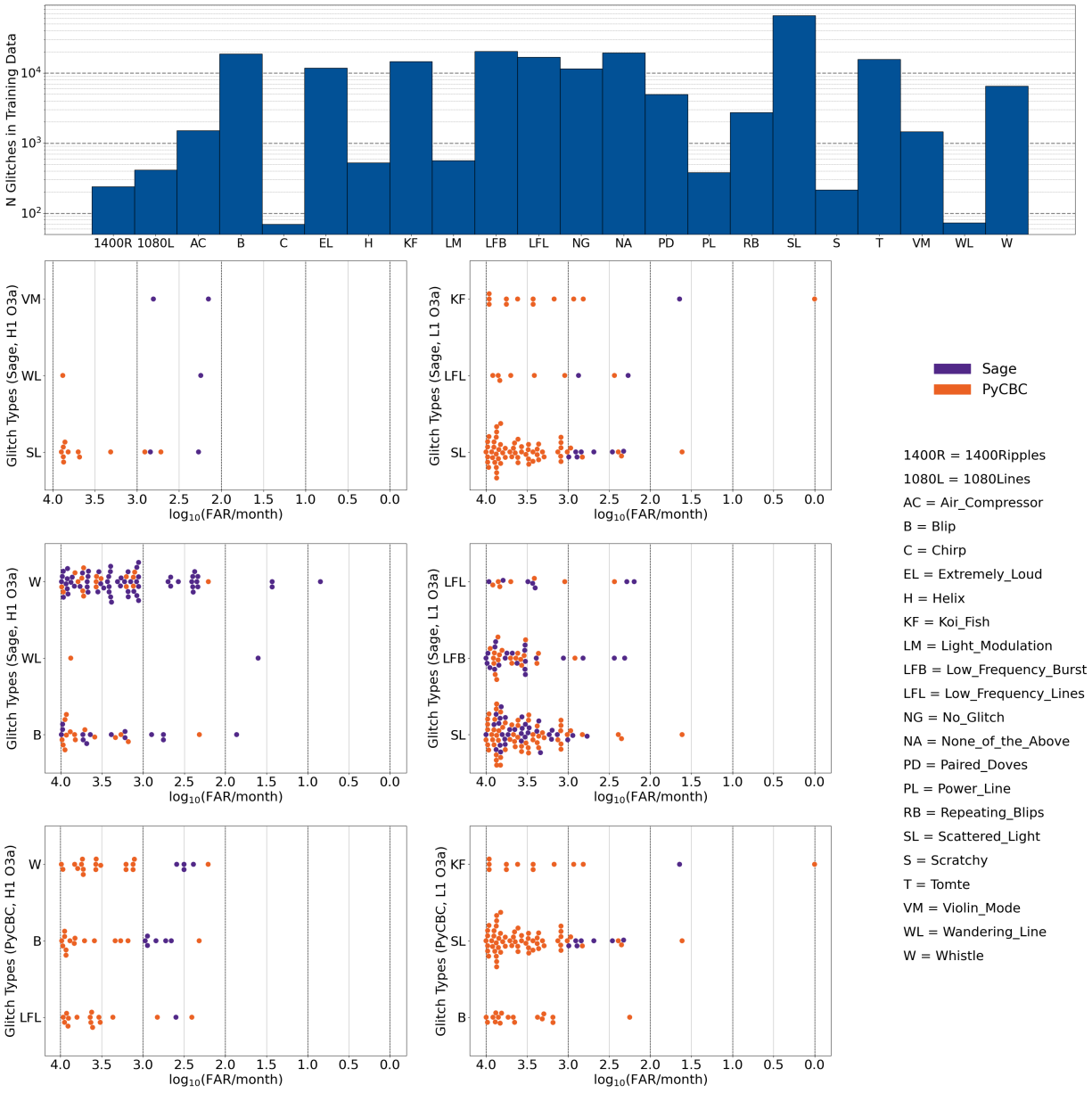


Figure 4.30: Glitch triggers from different glitch types at FARs from 10^4 per month to 1 per month for different Sage runs and PyCBC. The left and right columns correspond to the H1 and L1 detectors, respectively. The top row corresponds to the Sage - Annealed Train run, the middle row to the Sage - Broad run and the bottom row to PyCBC. The glitch types chosen for each panel produced the top three triggers for that configuration ("top" here refers to triggers with the lowest observed FARs).

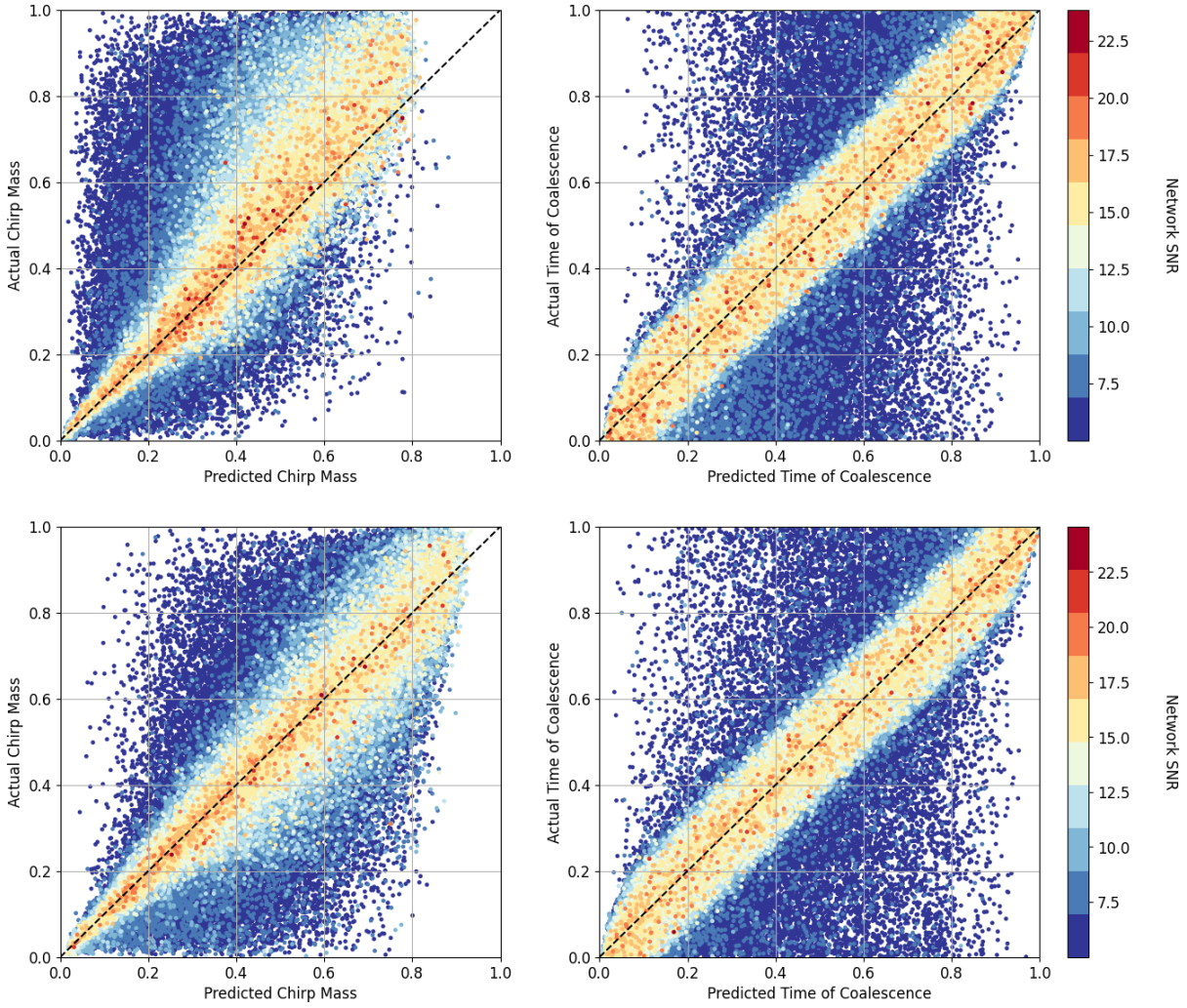


Figure 4.31: Actual value of GW parameters: chirp mass $\mathcal{M}_c$ and time of coalescence t_c, plotted against the network predicted value for a validation epoch in different SAGE runs. The points are coloured based on the network optimal SNR of their corresponding gravitational wave signal. The top row corresponds to the SAGE - ANNEALED TRAIN run and the bottom row to the SAGE - BROAD run.

Part III

Gravitational Wave Search in O3 & Closing Remarks

5

Hunting for Gravitational Waves!

"For a moment, nothing happened. Then, after a second or so, nothing continued to happen."

The Hitchhiker's Guide to the Galaxy
Douglas Adams

Optional Flavour 2

The quote shown above by Douglas Adams was the first thing that came to mind right before running the search on O3. However, we got lucky enough to not just find almost all GWs within the training distribution, but 5 entirely new GWs at an FAR $<$ 1/year. So I hope this chapter excites you, the reader, as much as it does me—not just for the results presented here but rather the remarkable potential of ML for GW science.

We report the discovery of 5 previously undetected binary black hole mergers in archival data from the third gravitational-wave observing run, identified using the machine-learning-based Sage search pipeline. Of the 5 newly identified events, all 5 have an astrophysical probability $p_{\text{astro}} > 0.5$ and are recovered at a false alarm rate less than 2 per year. In addition, we recover 21 out of 27 in-distribution (ID) events (events within the Sage search parameter space defined in chapter 4) reported in GWTC-2.1 and GWTC-3 and, surprisingly, 21 out of 38 OOD. Furthermore, we find one event (GW191204_110529-v1) that is at a FAR of 3.3 per year within GWTC at a FAR of $<$ 1/year, improving the confidence of a known event above the typical event detection threshold of 2/year. One of the new events was coincident with a sub-threshold trigger from GstLAL, making it a re-discovery. We perform Bayesian parameter estimation on all new events (including the re-discovery) to explore their physical properties.

5.1 Introduction

GW observatories have enabled direct observations of compact binary coalescences, offering unique access to the strong-field regime of gravity and compact object populations [275, 13, 14, 15, 18, 19, 20, 22, 23, 24, 2, 3, 4, 5, 276]. The discovery of new GW events with each observing run has been instrumental in advancing multi-messenger astrophysics [277, 29], cosmology [278, 279], tests of general relativity [280, 281], stellar evolution [282], nuclear astrophysics [283, 284] among many other fields of astrophysics. With increasing observations, we improve the completeness of the compact object population, get better estimates of event rates and begin to populate sparsely sampled regions of the GW parameter space.

Matched filtering is an optimal technique for detecting signals with known parameters in stationary Gaussian noise. However, GW strain is non-stationary and non-Gaussian, and searches must contend with unknown signal parameters, noise PSD estimation uncertainties, and non-Gaussian transient artefacts (glitches) that can reduce detection sensitivity [61, 285]. In recent years, ML has emerged as a compelling alternative to address these challenges [286, 287, 288, 289, 290, 291, 292, 293], but its performance remains limited by poor generalisation outside the training distribution [191, 192], unreliable glitch rejection, training data biases [294], and data-intensive requirements [194]. In our previous work [189] and in chapter 4, we introduced Sage, a BBH detection pipeline explicitly designed to address these challenges. While no approach is immune to the limitations of data-driven learning, Sage substantially improves generalisation to OOD PSDs, mitigates training-induced biases, demonstrates robust glitch rejection, and achieves significantly greater data efficiency than previous ML pipelines. On the MLGWSC-1 challenge dataset [290], we showed that Sage outperformed a benchmark PyCBC search, recovering $\approx$ 11% more signals at a FAR of 1 per month, and surpassed previous ML pipelines by $\approx$ 50%. Two of these existing ML pipelines, Aframe and AresGW, have both conducted a search on O3, with the former finding no new events at an FAR of $<$ 1/year and the latter claiming one new event. Given the significant improvement that we have seen comparing Sage to these existing pipelines in chapter 4, we expect to recover additional events and aim to establish a concrete case behind why matched filtering pipelines may have missed these events. In this work, we deploy Sage on archival O3 data, discovering binary black hole mergers missed by existing pipelines and establishing a new baseline for ML–driven GW searches.

Beyond the direct recovery of additional events, our results carry broader implications. First, they complement existing GW catalogues by providing independent detections that extend the sensitivity of current searches. Second, they highlight potential biases in existing ML pipelines, which may systematically underperform on certain classes of signals. Third, they shed light on why matched filtering pipelines may downweight or miss specific events, owing to noise artefacts, PSD estimation uncertainties, limitations in template coverage or other pipeline-specific nuances. Beyond recovering additional events, it enables characterisation of non-Gaussian noise features, assessment of generalisation

of ML detection pipelines to OOD distributions, and validation of detection sensitivity at low false alarm rates in practice. The results also demonstrate whether the pipeline's data efficiency translates into tangible advantages and establish a reproducible benchmark for future ML–based GW searches. Taken together, these findings illustrate both the opportunities and the challenges of integrating ML pipelines into the broader landscape of GW discovery.

5.2 Designing the Search

5.2.1 O3 Search Data

The third observing run of the Advanced LIGO and Virgo detectors spanned from 2019-04-01T15:00:00 (GPS=1238166018) to 2020-03-27T17:00:00 (GPS=1269363618). Data were collected from the LIGO Hanford (active time = ≈ 245.2 days; active time is the total time that the detector was collecting data), LIGO Livingston (active time = ≈ 251.9 days), and Virgo (active time = ≈ 250.3 days) detectors, with varying duty cycles across the instruments. The dataset consists of calibrated strain time series sampled at 4096 Hz, made publicly available through the Gravitational Wave Open Science Center. For this search, we make use of all the available overlapping data between at least two detectors to provide the search input for our SAGE pipeline. The current SAGE architecture works for exactly 2 detectors and no single or 3-detector search was conducted. More information about the search will be provided in section 5.2.3.

Power Spectral Densities: To study the PSDs, we construct a library of representative samples from the O3 observing run, covering all the available time coincident data from every pair of detectors. To generate a strain sample for estimating a PSD, a detector is first selected and a segment of sufficient length is chosen. Within this segment, a contiguous slice of fixed duration, $T_{\text{sample}} = 15$ s, is drawn at a uniformly random start time.

Once the time series slice is obtained, its one-sided power spectral density is estimated using the Welch method [232]. The strain is down-sampled to $f_s = 2048$ Hz, and a PSD is computed for each strain sample with the following choices for the Welch method: segment length of 4 s with median averaging to reduce the influence of transient outliers. The median averaging or median-mean method [91] splits the set of segment PSDs into odd- and even-indexed groups. For each group, it computes the median PSD across segments, correcting for the downward bias of the median (refer to [91] appendix B for more details about the bias). The final PSD estimate is then the average of the two bias-corrected medians, one from the odd group and one from the even group. The resulting PSD provides the frequency-domain representation of the chosen strain sample. This procedure is repeated independently for H1, L1 and V1. These PSDs are saved for a method of constructing the background ranking statistic distribution.

For plotting purposes, we do the following: Let $\{P_k(f)\}_{k=1}^{N}$ denote the set of estimated PSDs from

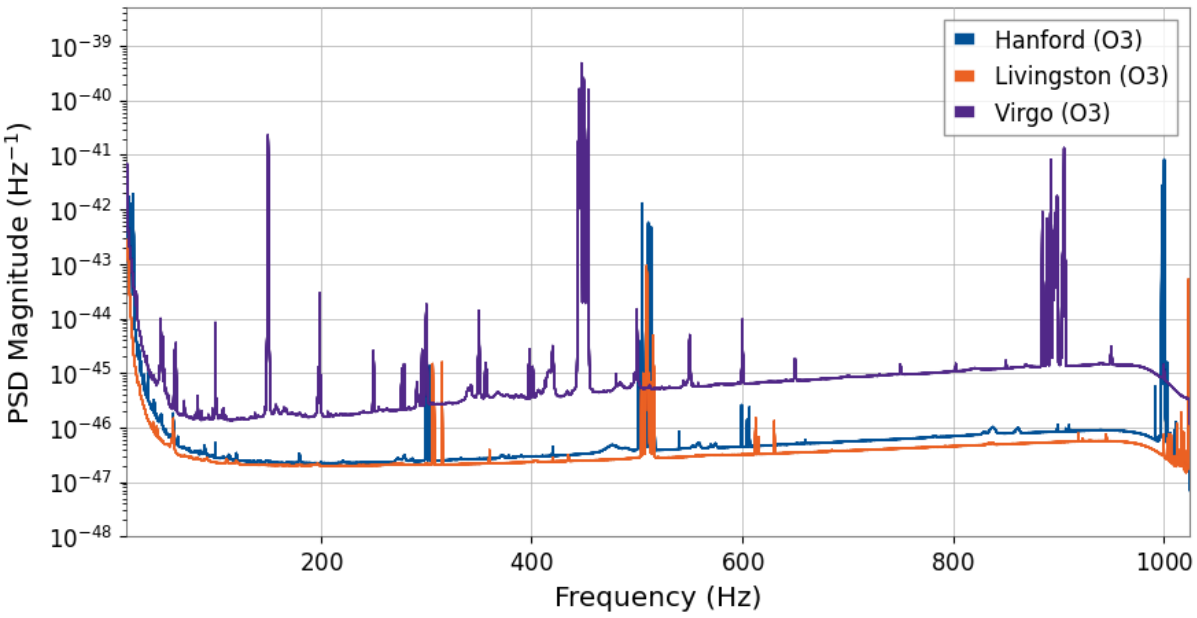


Figure 5.1: Representative power spectral density of the LIGO Hanford, LIGO Livingston, and Virgo interferometers' strain sensitivity. Strain data obtained within the time range 2019-04-01 15:00:00 UTC (GPS=1238166018) to 2020-03-27 17:00:00 UTC (GPS=1269363618). Frequencies span from 15Hz, set by the low-frequency noise cut-off, to 1024Hz, the upper bound imposed by the Nyquist limit.

N consecutive data segments. These are partitioned into $M = 30$ disjoint subsets (or chunks)

$$C_m = \{P_k(f) : k \in I_m\}, \quad m = 1, \ldots, M, \tag{5.1}$$

where I_m indexes consecutive time segments. For each chunk C_m, the representative PSD is defined as the elementwise median

$$\tilde{P}_m(f) = \text{median}\{P_k(f) \mid k \in I_m\}. \tag{5.2}$$

This procedure downweights outliers and transient noise, yielding more robust spectral estimates across time. This procedure produces a set of representative PSD curves for each detector, shown in Fig. 5.1. These curves are plotted on a logarithmic scale over the frequency range 15–1024 Hz. This allows us to analyse general data quality over an observing run and whether or not SAGE has been trained to handle it.

Non-Gaussian Noise Artefacts: We look into the data quality in O3, specifically regarding glitches using GravitySpy [233], since they are highly detrimental to search algorithms. This is useful to get an idea of how to mitigate the impact of glitches during the search. GravitySpy classifies 233981 glitches from LIGO Hanford and 379805 glitches from LIGO Livingston detectors. Given the vast amount of data samples that ML algorithms need in order to achieve learnt invariance to these glitches (see chapter 3, section 3.11.5) and the 22 different classes of glitches considered in GravitySpy, it is not unreasonable to assume that ML will struggle to fully learn to be invariant to all types of glitches. We

train Sage with spatial and channel-wise attention (as described in chapter 4), which aid in learning the difference betweeen the general structure of a signal to that of a glitch.

GWTC Search Results: GWTC-1 [2], GWTC-2.1 [3] and GWTC-3 [5] combined report a total of 90 gravitational wave candidates with a `p_astro` > 0.5 (where `p_astro` is the probability that a trigger is of astrophysical origin) providing a unique view of black holes and neutron stars. These included both matched-filter based approaches, such as GstLAL, PyCBC, and MBTA, which rely on template banks of waveform models to perform optimal filtering, as well as unmodelled searches like cWB, which identify excess power transients without assuming a specific signal morphology. Each pipeline reported candidate events with associated detection statistics, false alarm rates and `p_astro`, and cross-validation between methods was used to establish the final set of confident detections. The catalogue provides a comprehensive record of the sources observed during O3, including both high-confidence detections and marginal triggers. More recently GWTC-4 [48] increases this catalogue of merging compact objects by another 158 mergers.

5.2.2 Data Acquisition and Creating Search Samples

Segment lists for the Hanford (H1), Livingston (L1), and Virgo (V1) detectors were obtained from GWOSC at a sampling rate of $f_s = 4096$ Hz. Only intervals marked with the science "`DATA`" flag were retained, yielding contiguous blocks defined by their start and end GPS times and durations. The strain was first downsampled to $f_s = 2048$ Hz to standardise across detectors and reduce storage and computational cost. A high-pass filter with cutoff $f_{\text{low}} = 15$ Hz was then applied to suppress low-frequency noise. Segments shorter than $T_{\text{min}} = 22$ s were discarded, ensuring that after cropping 2.5,s from each end to mitigate edge artefacts introduced by filtering and decimation, only intervals suitable for analysis with Sage remained. The resulting time series were stored in compressed HDF5 files, organised by detector and observing run for efficient access in subsequent search stages.

To identify analysis-ready intervals shared across detectors, coincident times were constructed by scanning the segment lists with a cadence of 0.5 s. This ensured that any common observing periods between every 2-detector combination were captured without gaps. These coincident segments then defined the search domain, from which fixed-length samples of 17 s were extracted with a sliding window of 0.1 s, similar to the injection study seen in chapter 4 section 4.3. In total, this procedure yielded approximately $\approx$ 203.6 days of valid, preprocessed strain data across the HL detectors for O3a and O3b, 193.5 days across the LV and 186.31 days across the HV detectors. This corpus forms the foundation of the analysis presented in the following sections. Each 2-detector data sample is preprocessed and provided to Sage, in the same way as described in chapter 4 section 4.2, to obtain a ranking statistic and a point-estimate of the time of coalescence and chirp mass.

Bird's-Eye View of Data Usage: There is significant overlap between the data used for training and for the search, arising from the limited variety of PSD and glitch features in the O3 dataset.

This overlap is partly justified by the definition of a sample in Sage: each sample consists of a two-detector strain time series explicitly constructed to exclude zero-lag coincident data. Furthermore, samples from both classes are recoloured using PSDs from different times in the observing run, producing two-detector noise realisations that will never appear in a zero-lag search. Glitch features, however, are less affected by the recolouring and therefore introduce some degree of data leakage between training and search. Importantly, since the same background data are used to construct both the signal and noise classes during training, the noise class does not correspond strictly to "glitch-containing" data but rather to data without an astrophysical signal. Thus, glitches are not automatically attributed to noise; instead, the model learns the morphological differences between glitches and signals, with prior exposure to the glitch population expected during the search. In other words, the model learns what a signal is by recognising what it is not, rather than by explicitly learning what a glitch is. Moreover, given that only a limited number of glitches exist in the data, a bias due to limited data will further keep the model from memorising glitch features. Any new events found could have been a part of the training noise class, which further provides evidence toward the model not memorising.

Although the model is less likely to memorise glitches as noise, this setup is not as ideal as a fully independent training and testing dataset. For our future publication on this chapter, we plan to train Sage on O3b for a search on O3a and vice versa, thereby eliminating any data leakage. However, as shown in chapter 4, exposure to a wide variety of PSD realisations is crucial for strong detection efficiency. This can be addressed by training a time-series generative model on PSD realisations from O3 and sampling from it to obtain the PSD realisations required for the recolouring procedure during Sage training. Combined with explicitly weighting glitch samples more heavily within the independent training data, this approach would ensure similar or better performance to that demonstrated in chapter 4. To emphasise, we have *not* done this yet for this chapter, but plan to for our future publication on this work.

5.2.3 Sage for Offline Search

The Sage framework can be adapted for use in an offline search setting, where the goal is to analyse archival detector data for previously unidentified signals. Unlike real-time searches, which prioritise low latency and rapid follow-up, an offline search allows for more computationally intensive methods, extensive data conditioning, and the incorporation of additional cross-checks. This makes it possible to explore the data with improved sensitivity and robustness, while also enabling the validation of candidate events under more controlled conditions.

Design Challenges

Not Everything can be Trained Away: Before performing a full search, it is important to understand how the network responds to contaminated noise samples (noise samples that are contaminated by signal features in one or both detectors used for the search), as the background estimation procedure

inherently involves evaluating time slid data that will contain such samples. In particular, the network may encounter samples with a signal in one detector but not in the other or with signals with different intrinsic parameters in both detectors. To investigate whether the model can appropriately downweight these samples, we explicitly include contaminated noise samples in the noise class during training. The two experiments described below are designed to test the network's ability to distinguish true astrophysical signals from these contaminated noise realisations and to quantify the effect of including them in the training set on the network's output distributions.

To investigate how the network distinguishes between astrophysical signals and signal-contaminated noise (or simply contaminated noise from now) samples that will be seen during time slid background estimation, we trained a version of Sage in which the noise class was enriched with contaminated noise samples — specifically, samples containing a noisy signal in one detector and pure noise in the other. During training, 75% of the noise class was composed of these contaminated nosie samples. After training, we evaluated the network on a testing set containing 250,000 samples in each class, where the noise class consisted entirely of contaminated noise samples and the signal class contained purely astrophysical signals. For each sample, the raw network output (previously defined as the ranking statistic) was recorded. Histograms of the ranking statistics for the two classes were then computed and plotted together in Fig. 5.2 to visualise the inability of the network to separate signal from contaminated noise under these conditions. Training the network this way can be very detrimental given that there can be real situations where a signal is seen in one detector above the SNR wall and not in the other. Thus, this kind of training will confuse the model and might lower the overall detection efficiency.

To further probe the network's ability to distinguish astrophysical signals from contaminated noise, we repeated the previous experiment using a different definition of contaminated samples. In this setup, each contaminated sample consisted of two noisy signals — one in each detector — combined into a single sample. These two signals either had different intrinsic parameters or shared the same parameters but were offset more than the light travel time between the two considered detectors in coalescence time. We repeat the same experiment as above and obtain Fig. 5.3 to further illustrate the models failure.

Cost of Coherence: In a coherent search, the ranking statistic is computed jointly across all detectors, producing a single value for each candidate event. Coherent searches are searching for phase consistency between detectors and not just parameter consistency. So coherent searches care about consistent sky position, geocentric arrival time, polarisation, inclination and initial phase in addition to mass and distance. Coincident searches, on the other hand, only care about rough mass, time of arrival, and amplitude consistency. This makes the learning problem harder. Moreover, unlike coincident searches, where ranking statistics from individual detectors can be combined post hoc, a coherent search does not generate independent per-detector statistics that can later be combined. Consequently, background estimation cannot be performed simply by mixing and matching ranking statistics separated by more than the light travel time between detectors. Instead, the full strain time series $h(t)$ must

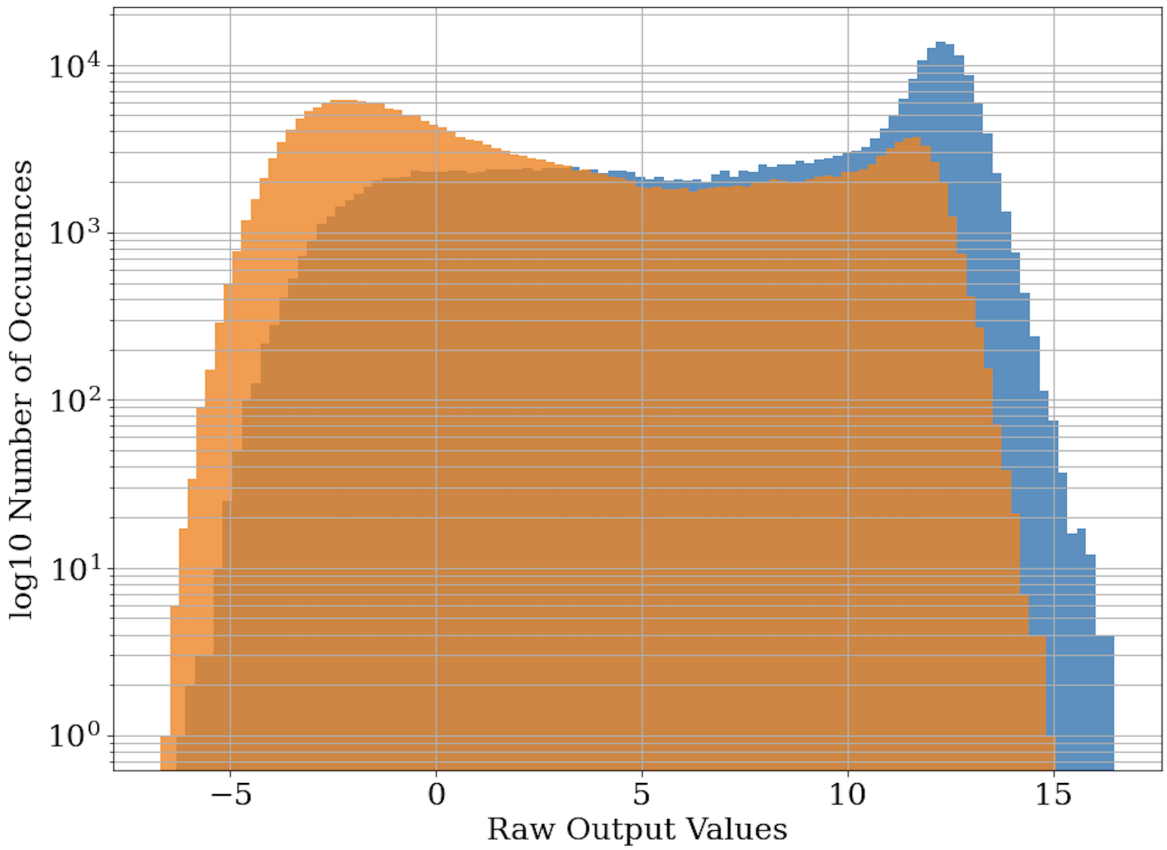


Figure 5.2: Histograms of the network output (ranking statistic) for a testing dataset containing 100% contaminated noise samples in the noise class and astrophysical signals in the signal class. The plot compares the distributions of the network output for the two classes, illustrating the network's inability to properly distinguish astrophysical signals from contaminated noise after being trained with a majority of contaminated noise samples in the noise class.

be shifted appropriately and the search rerun for each timeslide. This procedure is computationally expensive, and performing extensive background estimation on the fly is infeasible under low-latency search conditions.

To reduce this cost, one could either significantly reduce the network size, which alone is insufficient, or store intermediate, per-detector representations of the data in a compressed form—such as a latent-space encoding—that can later be combined to rapidly compute the coherent ranking statistic. In contrast, traditional matched-filtering approaches require only the storage of per-detector signal-to-noise ratios, ρ_1 and ρ_2, which can be combined via $\sqrt{\rho_1^2 + \rho_2^2}$ to obtain the final ranking statistic. No straightforward analogous shortcut exists for machine-learning-based coherent searches, highlighting the significant computational overhead inherent to such methods. In our case, since we run SAGE offline, we do not opt for a method to reduce this compute time.

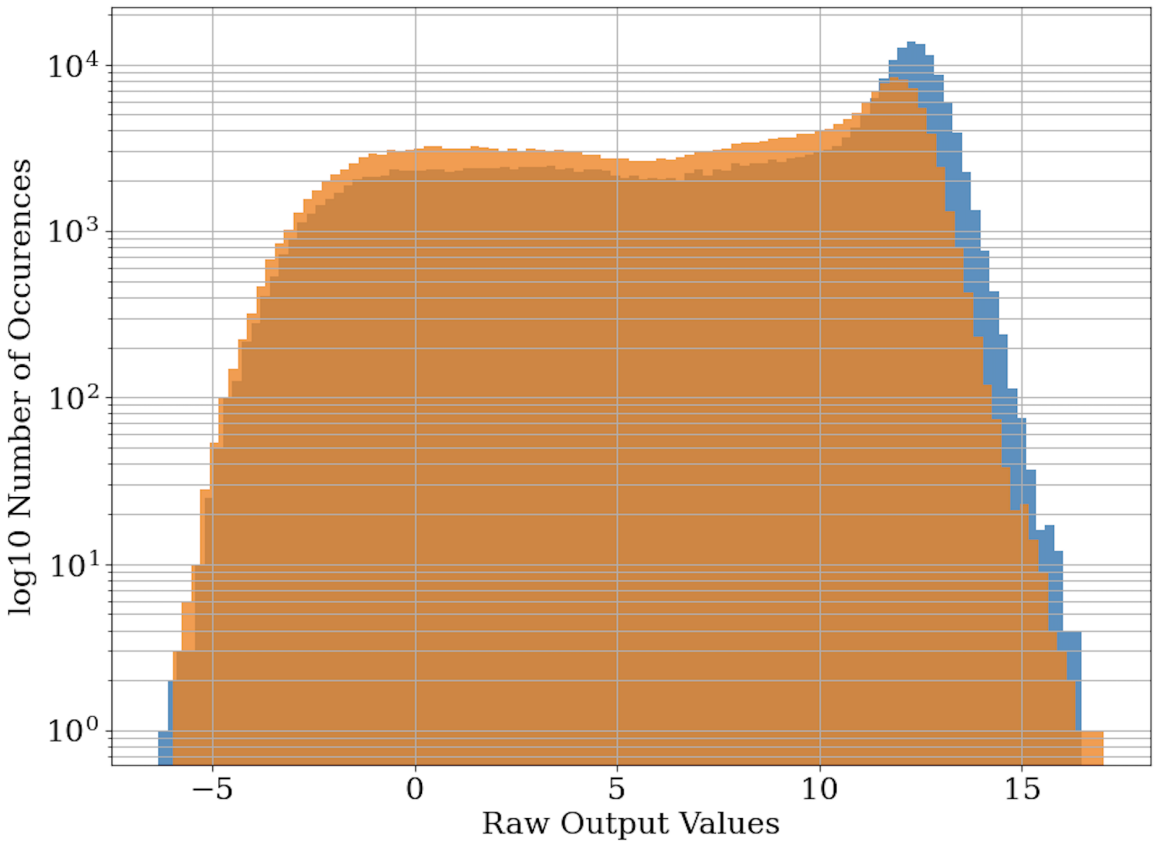


Figure 5.3: Histograms of the network output (ranking statistic) for a testing dataset in which the noise class contains 100% contaminated noise samples, defined as two noisy signals (either with differing parameters or differing coalescence times) combined into a single sample, and the signal class contains astrophysical signals within the light travel time between the two considered detectors. The plot compares the distributions of the network output for the two classes, demonstrating the network's ability to distinguish astrophysical signals from these more complex contaminated noise samples.

Training 2-ifo Combinations

Sage was initially designed for two-interferometer (IFO) operation. In this work, we trained the network on all three possible two-IFO combinations: Hanford–Livingston (HL), Hanford–Virgo (HV), and Livingston–Virgo (LV). While Sage can in principle be extended to three-IFO operation, doing so requires additional research and development to accommodate the larger input dimensionality, increased variance, and modified training strategies. To prioritise efficiency, we therefore performed three separate searches, each on a distinct two-IFO combination, rather than attempting a single three-IFO search.

5.2.4 Background Estimation

In typical gravitational-wave searches, background estimation is performed by taking approximately one week of data around a candidate event and applying time slides between detectors to produce a

very large effective background, often equivalent to tens of thousands of years. This procedure is necessary in matched-filtering pipelines to ensure that the background distribution accurately reflects the noise properties around the event, while avoiding data beyond a week–where changes in the PSD and general changes in the detector characteristics introduces a distribution shift that violates the underlying assumptions of matched filtering.

For Sage, however, this restriction is not strictly necessary. Because the network is trained to handle distributional shifts—through exposure to time-shifted and recoloured data—it can, in principle, incorporate data from longer timescales without invalidating its statistical assumptions. In the current procedure, the time-shifted data used during training are deliberately excluded from background estimation. This choice is made to maintain a strict statistical separation between the data that informs the model's learned representations and the data used to construct the null distribution for significance testing. Reusing the same time-shifted segments could introduce subtle correlations between the network's training and evaluation phases, potentially biasing the estimated false alarm rates. By separating the training and background data in this way, the procedure ensures that the detection statistics derived from the network reflect genuine performance under null-noise conditions, rather than any artefacts of the training procedure.

It is important to note, however, that there are known saturation effects in this time-sliding procedure. Beyond a certain amount of time-slid background, the error in the background estimation begins to increase due to the limited number of noise realisations with independent PSD and glitches in the dataset. In this section, we also discuss alternative approaches to background estimation for machine-learning-based searches that could potentially be adopted into Sage.

Using O3 Strain: The first background estimation method applies the standard time-slide method directly to the O3 strain data. Here, a raw strain window generated for the search from one detector is randomly selected and paired with a window from a second detector that is offset by at least 100 seconds in GPS time. This offset ensures that any true astrophysical coincidences are excluded. Since Sage is explicitly trained to handle distributional shifts in the PSD, no restrictions on the maximum separation are imposed. Using this approach, sufficient pairs were generated to produce the equivalent of 365.2422 days of background, providing an effective year-long background distribution for the search.

Figure 5.4 illustrates the distribution of the Sage ranking statistic for each 2-IFO combination over approximately one year of background data generated via timeslides. While the overall shapes are similar, differences in the high-ranking tails reflect the varying noise characteristics of each detector pair. This comparison provides a consistent reference for estimating false alarm rates and comparing detection thresholds across different detector combinations. It is interesting to see the Sage finds LV noise samples the hardest to classify compared to other 2-IFO combinations with HV being the easiest. This makes sense since we see more glitches in the Livingston detector compared to the Hanford detector. GWTC-3 [5] states that a glitch rate of $0.29\,\mathrm{min}^{-1}$ in H1 O3a and $0.32\,\mathrm{min}^{-1}$ in H1 O3b, whereas

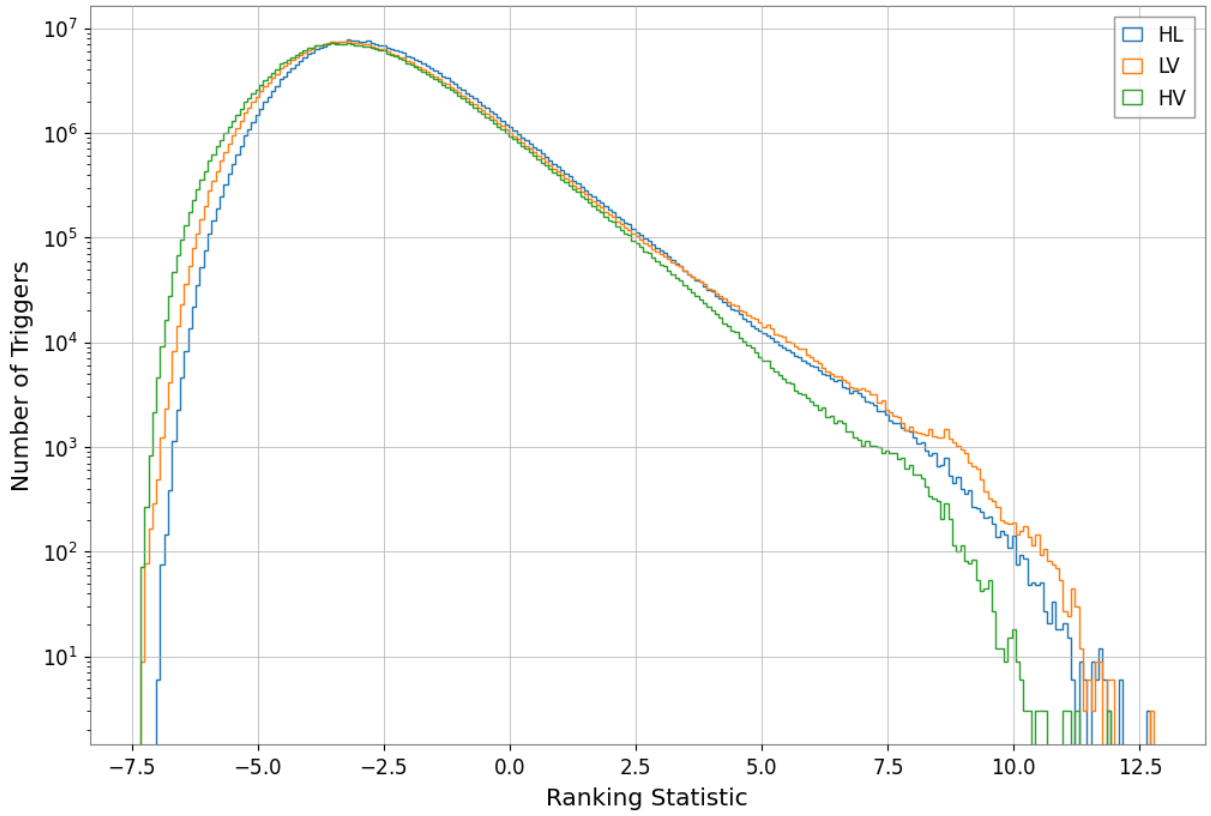


Figure 5.4: Histogram of ranking statistics obtained from background triggers using O3 strain with the time-slide method. A raw strain window from one detector is paired with a window from a second detector at least 100 seconds apart in GPS time, ensuring exclusion of astrophysical coincidences while preserving realistic noise statistics. Enough pairings were generated to yield a total background equivalent to one year of data.

1.10 min^{-1} in L1 O3a and 1.17 min^{-1} in L1 O3b. So the relative differences we see in these histograms make sense.

Using Coloured Gaussian Noise (from O3 PSDs) & Glitches: In this second method, background samples are generated using a mixture of coloured Gaussian noise and real detector glitches. A set of 250,000 PSDs is estimated for each detector from randomly selected non-overlapping 15 s stretches of O3 data. For each noise sample, a random PSD is drawn independently for both detectors, and a two-detector Gaussian realisation is generated by colouring white Gaussian noise with the chosen PSDs.

To include realistic transient noise, glitch information from Gravity Spy is incorporated. All unique glitches from O3 are considered, with their start times and durations used to determine the total glitch exposure in each detector. For H1, this corresponds to 313,609 s of glitch time, and for L1, 434,666,s. Relative to the total observing duration, this yields glitch fractions of 2.21% in H1 and 2.88% in L1. When scaled to one year of background data (315,569,520 samples), this implies approximately 6.97 million glitch-containing samples from H1 and 9.09 million from L1, resulting in ~ 16.1 million glitch

samples in total if glitches did not occur simultaneously in both detector. For our experiment, glitches are introduced independently in each detector, such that a given two-detector sample may contain a glitch in one or both detectors, depending on the random draw. In total, one year of background data is constructed by combining the glitch samples with sufficient coverage of Gaussian realisations. Each sample is then passed through Sage to obtain the background distribution of the ranking statistic.

Figure 5.5 demonstrates the contribution of different noise sources to the Sage ranking statistic. The combined histogram reflects the overall background expected when both noise types are present, providing insight into how glitches and Gaussian noise collectively shape the background for Hi–Li searches. The histogram for the pure coloured Gaussian noise case is well behaved, as expected, and is similar in shape to the histogram seen in Fig. 5.4. This tells us that Sage treats real noise and coloured Gaussian noise in a similar manner, providing evidence for robust noise handling. Including the effect of glitches to form the combined background estimate allows the final distribution to get statistically much closer to the Hi-Li distribution seen in Fig. 5.4.

To quantify the similarity, we performed the Anderson-Darling test–a test to quantify the similarity in the samples of data drawn from two distribution A and B; thus acting as a metric to tell whether A and B are equivalent up to statistical uncertainties). This test is particularly well suited for distributions that vary primarily in the tails; and has been used in the GW context before [295, 296]. However, upon comparison, we still saw that the backgrounds in Figs. 5.5 and 5.4 for Hi-Li were different in the tails. This is primarily due to the latter being significantly contaminated by signals whereas the former should be clean save any signal+glitch overlap that GravitySpy might have picked up. The background distribution obtained in Fig. 5.5 could be *better* given that we will not be overestimating the background. However, we will explore this idea in our future work to validate this claim.

Choice for Background: For the present study, we opt to use the time slide method with O3 strain samples for background estimation. While the coloured Gaussian noise combined with glitches approach is a good enough approximation, it is non-standard and would require additional investigation to ensure its reliability for inference. In contrast, the timeslid method is widely used in established matched filtering pipelines and provides the most realistic background, reflecting the true statistical properties of the search data. However, as mentioned, this distribution is contaminated with signals and we must think of a way to reduce this contamination as much as possible.

Hierarchical Trigger Removal: The background constructed from timeslides may be contaminated by astrophysical signals that appear in only one detector, or by signals present in both detectors but with different intrinsic parameters. Since time slid samples are separated by at least 100 seconds across detectors, the same signal with identical parameters will not appear in coincidence. To mitigate such contamination, we employ a hierarchical trigger removal procedure (similar to PyCBC [93]):

1. *Rank ordering of background triggers.* All timeslid background triggers are sorted in descending order of the detection statistic $\mathcal{R}_s$. The algorithm processes triggers sequentially from the

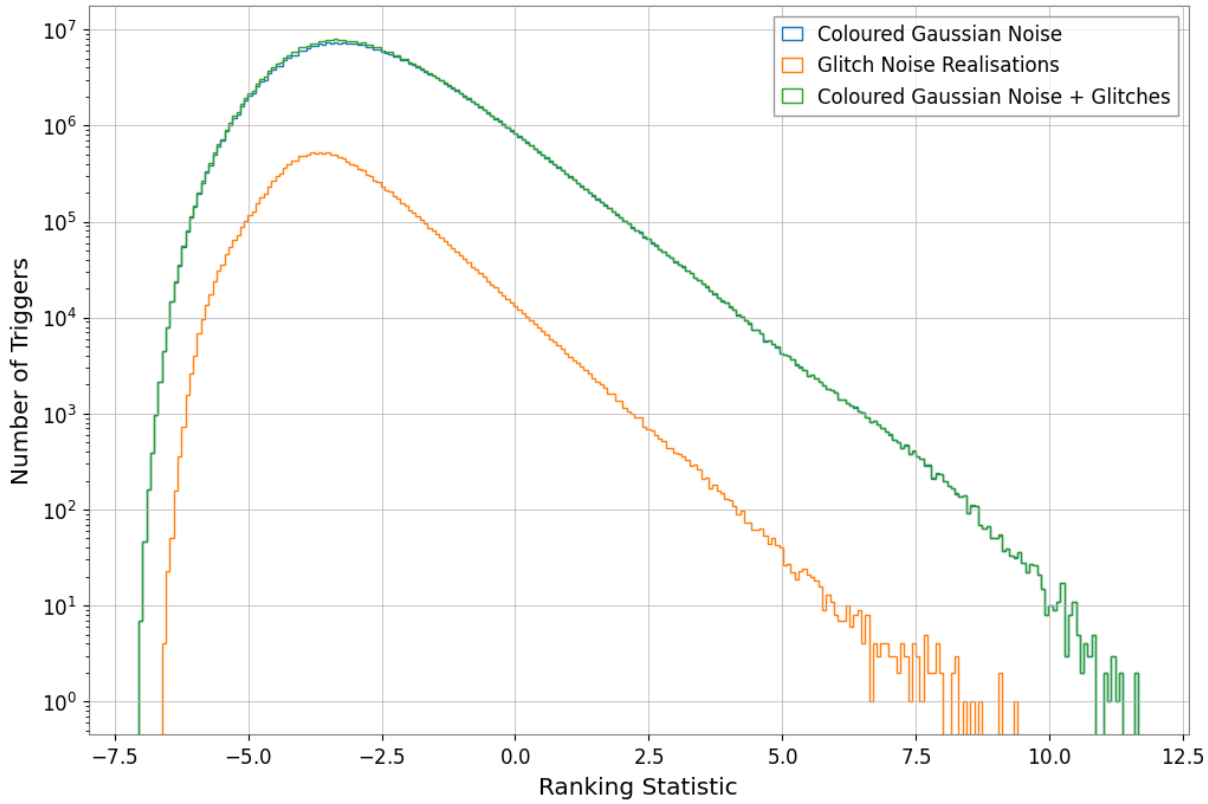


Figure 5.5: Histogram of ranking statistics obtained from background triggers using a combination of coloured Gaussian noise and real glitches from Gravity Spy (in the H1-L1 detector setup). We show three separate histograms: one of only glitch realisations, one of only coloured Gaussian noise realisation with PSDs taken from O3, and lastly the aggregate of the two. This can be used to compare the contribution due to glitches in the background.

loudest downwards.

2. *Coincidence check with zerolag triggers.* For each background trigger with ranking statistic $\mathcal{R}_{\text{S}}{}^{\text{bg}}$, we identify all zerolag triggers with $\mathcal{R}_{\text{S}} \geq \mathcal{R}_{\text{S}}{}^{\text{bg}}$. We then check whether any of these zerolag triggers occur within a predefined time window of the background trigger in either detector. If such a temporal overlap is found, the background trigger is considered potentially influenced by a real signal.

3. *Removal of contaminated triggers.* If any coincident zerolag trigger is found, the corresponding timeslid trigger is flagged as contaminated and removed from the background set, along with all triggers within the background with the same GPS time from that detector. This is under the assumption that for a genuine signal contamination, the zerolag ranking statistic would be higher than that of the background ranking statistic.

4. *Iteration and stopping.* The procedure continues iteratively, descending through the ranked list of background triggers, until either all triggers have been examined or the stopping condition is reached. If no coincident zerolag trigger, further removal is halted to avoid discarding genuine

background statistics. Removal is also stopped if more than 6 months worth of background is discarded from the background, although this is highly unlikely.

This hierarchical approach ensures that triggers in the background distribution potentially associated with astrophysical signals are removed, thereby reducing contamination in the background estimation while preserving the integrity of coincident background statistics.

Caveat: For coherent searches (including ML search pipelines), the assumption that a genuine signal will always produce a higher zerolag ranking statistic than the corresponding background trigger may not hold. For ML, in regions of parameter space where the network struggles to learn the conditional relationships between detectors, a single-detector signal may be assigned a higher ranking statistic than a coincident two-detector signal due to a bias toward learning easier samples better. As a result, some contaminated background triggers may not be removed, leading to a potential overestimation of the true background in these cases.

5.2.5 Estimating `p_astro`

The probability of astrophysical origin, commonly denoted `p_astro`, is a quantitative measure used to assess whether a given candidate trigger is likely to correspond to a genuine gravitational-wave signal rather than arising from instrumental noise or other non-astrophysical sources. In practice, it represents a ranking or confidence score for the trigger, incorporating both the background distribution of noise events and the expected signal population. High values of `p_astro` indicate a strong likelihood that the trigger is astrophysical in nature, while low values suggest it is more consistent with noise. This probability provides a critical metric for prioritising candidate events for further analysis and follow-up observations. In this section, we will describe the procedure for computing `p_astro` for our search and talk about caveats when used for machine learning.

Injection Study: To compute the probability of a trigger being astrophysical (`p_astro`), we require the signal likelihood $p(\mathcal{R}_s|\text{signal})$ where $\mathcal{R}_s$ is the ranking statistic. This is estimated from a large-scale injection study. For this purpose, we first draw source parameters from population models constrained by GWTC-3. Specifically, we use the posterior hyperparameter distributions from the `PowerLawPeak` model, and take the maximum *a posteriori* values of the hyperparameters. These are then used to generate $\approx 700,000$ simulated binary black hole systems with the following primary parameters: component mass m_1, mass ratio q, redshift z, dimensionless spins χ_1, χ_2, and spin tilt angles $\cos\theta_1$, $\cos\theta_2$.

Derived quantities are subsequently calculated to fully characterise each system. The secondary mass is obtained as

$$m_2 = q \cdot m_1. \tag{5.3}$$

The azimuthal spin angles ϕ_1 and ϕ_2 are drawn uniformly from $[0, 2\pi]$. Spin vectors are then con-

structed in Cartesian coordinates as

$$s_{1x} = \chi_1 \sin\theta_1 \cos\phi_1, \qquad s_{1y} = \chi_1 \sin\theta_1 \sin\phi_1, \qquad s_{1z} = \chi_1 \cos\theta_1,$$
$$s_{2x} = \chi_2 \sin\theta_2 \cos\phi_2, \qquad s_{2y} = \chi_2 \sin\theta_2 \sin\phi_2, \qquad s_{2z} = \chi_2 \cos\theta_2,$$

where $\sin\theta_i = \sqrt{1-\cos^2\theta_i}$ for $i = 1, 2$. Redshifts are converted to luminosity distances d_L using a `Planck15` cosmology.

Additional extrinsic parameters required for waveform generation, including sky location (RA, Dec), inclination, coalescence phase, and polarisation, are drawn from standard astrophysical distributions (this is provided in table 4.1 in chapter 4). t_c is drawn uniformly from a short window [11.0, 11.2] s to comply with the requirements for SAGE. Together, these parameters fully define each simulated injection set used to construct the signal likelihood for `p_astro` estimation. These generated waveforms are injected in O3 noise (sampled with O3 with uniform random start time) and the ranking statistic is obtained from SAGE. We then perform 1D bounded kernel density estimation to obtain the signal likelihood for a given 2-IFO search.

Noise Likelihood: This is simply the 1D bounded kernel density estimation of the background distributions obtained in section 5.2.4 for each 2-IFO search.

Computing `p_astro`: We follow the FGMC formalism proposed in [297] to compute the total number of background and foreground triggers expected given the data collected above.

We consider a dataset of N events with ranking statistics $\{x_i\}$, each arising from either a foreground or background process. Let Λ_f and Λ_b denote the expected counts of foreground and background events, and let $\hat{f}(x, \theta)$ and $\hat{b}(x, \theta)$ be the likelihood of obtaining a trigger with ranking statistic x under the foreground or background distribution, parameterised by θ. All parameters of the rate distributions except Λ_f and Λ_b are denoting using θ. The total (unnormalised) rates are then

$$f(x, \theta) = \Lambda_f \hat{f}(x, \theta), \quad b(x, \theta) = \Lambda_b \hat{b}(x, \theta). \tag{5.4}$$

where $f(x, \theta)$ and $b(x, \theta)$ are differential rates assuming that the foreground and background events are samples from an inhomogeneous Poisson process:

$$\frac{dN_f}{dx} = f(x, \theta), \quad \frac{dN_b}{dx} = b(x, \theta). \tag{5.5}$$

where N_f and N_b are the number of foreground and background events respectively with $N = N_f + N_b$.

We introduce latent state flags $g_i \in \{0, 1\}$ indicating whether event i is signal ($g_i = 1$) or noise ($g_i = 0$) for a dataset $d = x_i | i = 1, \ldots, n$. Using Bayes theorem, we can write the posterior probability

of $\{g_i\}, \Lambda_f, \Lambda_b$ and θ as,

$$p\left(\{g_i\}, \Lambda_f, \Lambda_b, \theta | d_{\text{to}}, N\right) = \frac{p\left(d_{\text{to}} | \{g_i\}, N, \Lambda_f, \Lambda_b, \theta\right) p\left(\{g_i\}, N, \Lambda_f, \Lambda_b, \theta\right)}{p(d_{\text{to}}, N)}. \tag{5.6}$$

Assuming the events are independent and that the dataset is ordered in time as d_{to}, the likelihood of observing the data d_{to} is

$$p(d_{\text{to}} | \{g_i\}, \Lambda_f, \Lambda_b, N, \theta) = \prod_{i:g_i=1} \hat{f}(x_i, \theta) \prod_{i:g_i=0} \hat{b}(x_i, \theta). \tag{5.7}$$

The prior distribution can be factorized as

$$\begin{aligned} p\left(\{g_i\}, N, \Lambda_f, \Lambda_b, \theta\right) &= p\left(\{g_i\} | N, \Lambda_f, \Lambda_b\right) p\left(N | \Lambda_f, \Lambda_b\right) p\left(\Lambda_f, \Lambda_b, \theta\right) && (5.8) \\ &= p\left(\{g_i\}, N | \Lambda_f, \Lambda_b\right) p\left(\Lambda_f, \Lambda_b, \theta\right). && (5.9) \end{aligned}$$

where the factors from the first line can be defined as follows:

$$p\left(\{g_i\} | N, \Lambda_f, \Lambda_b\right) = \left(\frac{\Lambda_f}{\Lambda_f + \Lambda_b}\right)^{N_f} \left(\frac{\Lambda_b}{\Lambda_f + \Lambda_b}\right)^{N_b}, \tag{5.10}$$

and assuming a Poisson process,

$$p\left(N | \Lambda_f, \Lambda_b\right) = \frac{\left(\Lambda_f + \Lambda_b\right)^N}{N!} e^{-(\Lambda_f + \Lambda_b)}. \tag{5.11}$$

Combining equations 5.10 and 5.11, the prior on the flags and counts can be written as

$$p(\{g_i\}, N | \Lambda_f, \Lambda_b) = \frac{\Lambda_f^{N_f} \Lambda_b^{N_b}}{N!} e^{-(\Lambda_f + \Lambda_b)}, \tag{5.12}$$

The rate prior can be chosen as the Jeffreys prior to be as uninformative as possible,

$$p(\Lambda_f, \Lambda_b, \theta) \propto \frac{p(\theta)}{\sqrt{\Lambda_f \Lambda_b}}. \tag{5.13}$$

Combining likelihood and prior, the joint posterior for flags, counts, and shape parameters is

$$p(\{g_i\}, \Lambda_f, \Lambda_b, \theta | \{x_i\}, N) \propto \left[\prod_{i:g_i=1} \Lambda_f \hat{f}(x_i, \theta) \prod_{i:g_i=0} \Lambda_b \hat{b}(x_i, \theta)\right] \times e^{-(\Lambda_f + \Lambda_b)} \frac{p(\theta)}{\sqrt{\Lambda_f \Lambda_b}}. \tag{5.14}$$

Marginalising over the unknown flags yields the posterior on rates and distribution parameters alone:

$$p(\Lambda_f, \Lambda_b, \theta|\{x_i\}, N) \propto \prod_{i=1}^{N} \left[\Lambda_f \hat{f}(x_i, \theta) + \Lambda_b \hat{b}(x_i, \theta)\right] e^{-(\Lambda_f+\Lambda_b)} \frac{p(\theta)}{\sqrt{\Lambda_f \Lambda_b}}. \tag{5.15}$$

This formalism provides a fully Bayesian, self-consistent estimate of the foreground and background rates while accounting for the uncertainty in event provenance, and naturally reduces to standard Poisson counting in the case of perfect classification. For more details regarding the derivation, please refer to [297].

We can use this formalism to now calculate the `p_astro`. We follow the derivation provided in [298] and obtain the following definition. The astrophysical probability `p_astro` quantifies the likelihood that a given trigger with detection statistic x arises from a signal rather than noise. Now, we can be more specific for the GW detection problem, and we redefine the notion of foreground and background to the signal and noise hypotheses $\mathcal{S}$ and $\varnothing$, respectively. Conditioning on priors π_s, π_n, Bayes' theorem gives

$$p_{\text{astro}}(x) = \frac{\pi_s p(x|\mathcal{S})}{\pi_s p(x|\mathcal{S}) + \pi_n p(x|\varnothing)}. \tag{5.16}$$

We define the prior terms π_s, π_n for the signal and noise as,

$$\pi_s = \frac{\Lambda_s}{\Lambda_s + \Lambda_n}, \ \pi_n = \frac{\Lambda_n}{\Lambda_s + \Lambda_n}. \tag{5.17}$$

From equation 5.15 of the FGMC formalism, the posterior on the counts, assuming that there are no additional shape parameter to be taken into account, is

$$p(\Lambda_s, \Lambda_n|\{x_i\}, N) \propto \prod_{i=1}^{N} \left[\Lambda_s p(x_i|\mathcal{S}) + \Lambda_n p(x_i|\varnothing)\right] e^{-(\Lambda_s+\Lambda_n)} \pi(\Lambda_s, \Lambda_n). \tag{5.18}$$

Finally, `p_astro` marginalises over the uncertainty in the mean counts,

$$p_{\text{astro}}(x) = \int d\Lambda_s \, d\Lambda_n \, \frac{\Lambda_s p(x|\mathcal{S}) \; p(\Lambda_s, \Lambda_n|\{x_i\}, N)}{\Lambda_s p(x|\mathcal{S}) + \Lambda_n p(x|\varnothing)}. \tag{5.19}$$

This framework connects the trigger-level probability to the underlying Poisson mixture model and forms the basis of all current gravitational-wave search pipelines.

Reparameterisation: In practice, directly estimating the mean counts Λ_s and Λ_n from a large set of triggers can be numerically challenging. The ratio of noise to signal triggers is typically very high, so including all triggers in the FGMC formalism may overwhelm the posterior with noise, potentially yielding a zero count for Λ_s (this is due to numerical precision and not a fundamental problem with

the formalism). To mitigate this, we reparameterise the problem in terms of the *total trigger rate* Λ and the *signal fraction* r, defined as

$$\Lambda = \Lambda_s + \Lambda_n, \qquad r = \frac{\Lambda_s}{\Lambda}. \tag{5.20}$$

In this parameterisation, the signal and noise counts are given by

$$\Lambda_s = r\,\Lambda, \qquad \Lambda_n = (1-r)\,\Lambda. \tag{5.21}$$

Substituting into the posterior yields,

$$p(\Lambda, r|\{x_i\}, N) \propto \prod_{i=1}^{N} \left[r\,\Lambda\, p(x_i|\mathcal{S}) + (1-r)\,\Lambda\, p(x_i|\varnothing) \right] \cdot e^{-\Lambda} \cdot \pi(\Lambda, r). \tag{5.22}$$

With the change of variables from (Λ_s, Λ_n) to (Λ, r), the prior probability distribution $\pi(\Lambda, r)$ must account for the transformation and is of the form

$$\pi(\Lambda, r) = \pi(\Lambda_s, \Lambda_n) \left| \frac{\partial(\Lambda_s, \Lambda_n)}{\partial(\Lambda, r)} \right| \tag{5.23}$$

$$= \frac{\Lambda}{\sqrt{r\Lambda \cdot (1-r)\Lambda}} \tag{5.24}$$

$$= \frac{1}{\sqrt{r(1-r)}}. \tag{5.25}$$

Now we can rewrite the `p_astro` as:

$$p_{\text{astro}}(x) = \int d\Lambda\, dr\, \frac{r\,\Lambda\, p(x|\mathcal{S})\; p(\Lambda, r|\{x_i\}, N)}{r\,\Lambda\, p(x|\mathcal{S}) + (1-r)\Lambda\, p(x|\varnothing)}. \tag{5.26}$$

This completes the reparameterisation and allows for stable and efficient computation of `p_astro` even in datasets dominated by noise triggers.

Practical Implementation: Now, we can apply this to our method to our data to compute the `p_astro`. First, we obtain the noise and signal likelihoods from the injection study and background estimation trigger data, respectively.

To construct likelihood functions for both signal and noise distributions, we employ a 1D bounded kernel density estimate evaluated on a fine grid within the bounds $[a, b]$ that define the domain of the ranking statistic, with appropriate normalisation and optionally thresholding (and renormalisation for the new thresholds). Thresholding is typically done for the likelihood and the triggers used for the analysis at some ranking statistic such that we do not have to deal with the computational burden of hundreds of millions of triggers. However, this procedure comes with a set of serious caveats in the case of ML.

Optional Thresholding: If a threshold is specified, the probability density below this value is set to zero, ensuring that only triggers above the threshold contribute to the likelihood. The truncated distribution is then renormalised by computing the integral of the surviving density.

Caveats: When computing `p_astro` using machine learning classifiers, extra care is required in how the likelihoods are defined and evaluated. In particular, an important difference arises in the low ranking statistic regime. For matched filtering pipelines, signals that cannot be recovered or that fall outside the template bank typically contribute at low SNR, and their ranking statistic distribution resembles that of noise. This ensures that the signal likelihood remains below or resembles that of the noise likelihood in the bulk of the low-statistic region, and astrophysical information is instead concentrated in the high-statistic tail.

In contrast, machine learning models are capable of extrapolation beyond their training distribution. As a result, a high SNR out-of-distribution signal may still be recognised by the model as "something astrophysical", but because of its OOD nature it can be assigned a low ranking statistic. These events form a distinct bulk distribution in the low-statistic regime, which does not necessarily resemble noise. Consequently, depending on the injection study, the signal likelihood may exceed the noise likelihood even at low ranking statistic. This effect introduces significant difficulties when evaluating `p_astro`, as the inferred signal fraction from the posterior becomes highly sensitive and unstable in parts of the low-statistic region.

A direct consequence is that the choice of ranking statistic threshold, which is typically introduced for computational efficiency, becomes critical. Changing the threshold alters the relative weight between the bulk and the tail of the likelihood, and therefore directly affects the computed `p_astro`. For machine learning based pipelines, the value of `p_astro` can therefore vary substantially with the placement of this cut, highlighting the need for careful treatment of likelihood normalisation and thresholding. A simpler injection study that uses the Sage training distribution might provide a useful metric that does not suffer from this issue, but this is technically not `p_astro` since the training distribution is not astrophysical.

Compromise: Since thresholding is not a stable option for now and further research is needed to find a solution for this issue, we resort to showing the results only from the signal likelihood built from the Sage training distribution for the corresponding 2-IFO network. We do this without the SNR rescaling procedure mentioned in chapter 4 that was used to place all signals above an optimal SNR threshold of 5. Figure 5.6 shows the kernel density estimates obtained from the Sage training distribution and background estimation of the LV search. We also show the one from the injection study which shows the clear bump of the signal likelihood over the noise likelihood at low ranking statistics.

A two-dimensional posterior distribution is then computed over the total rate Λ and the signal fraction r. We include $> 99\%$ of the posterior within this grid and ensure that the grid is fine enough

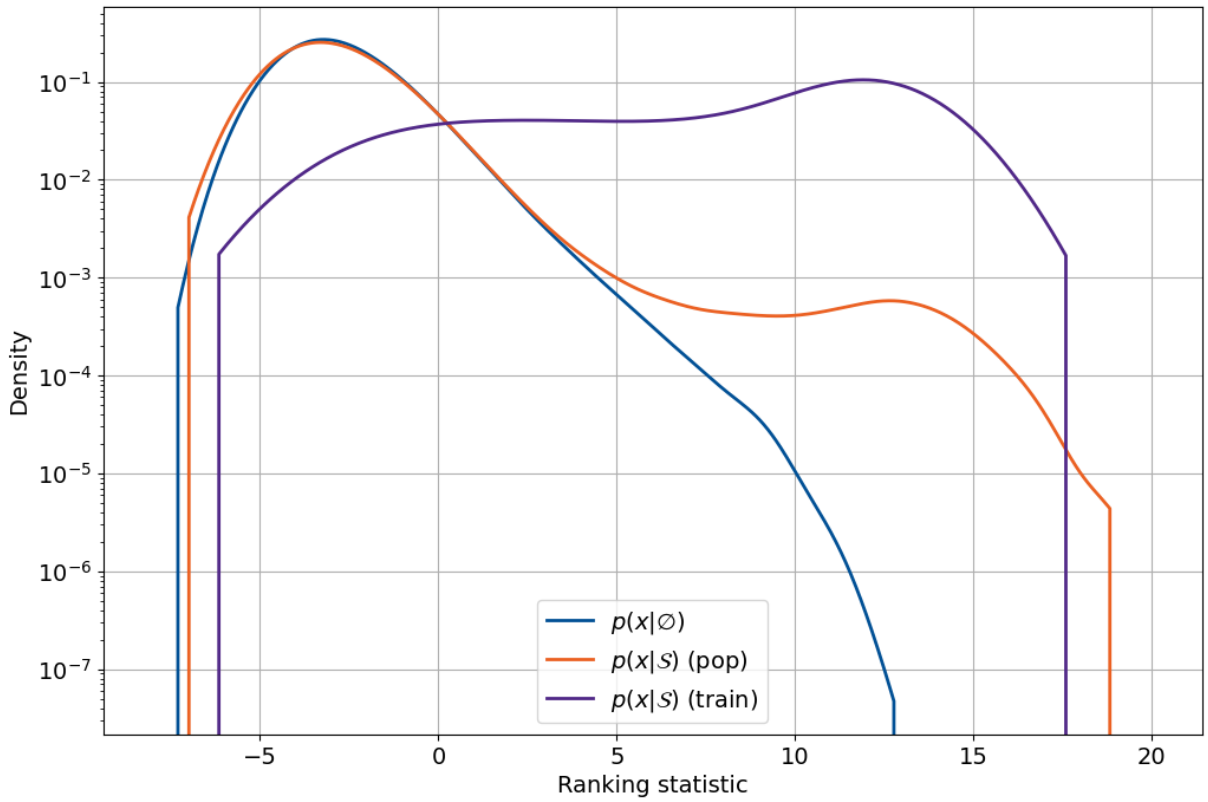


Figure 5.6: Kernel density estimates of the ranking statistic distributions for noise and signal triggers, both normalised to unit area. The comparison highlights the separation between the two populations, with the signal distribution shifted towards higher ranking statistic values relative to the noise.

for the final `p_astro` result to not change significantly for a chosen zerolag trigger with some delta change in the grid resolution. Now, we have all the ingredients for computing `p_astro` from equation 5.26. Although this formalism works fine for matched filtering, there are some extra considerations that have to be made for this to work for machine learning.

5.3 Catalogue of BBH events in O3

To construct a catalogue of BBH mergers from the third observing run, we apply the SAGE search pipeline to the strain data from the LIGO and Virgo detectors. The search is performed in the zero-lag configuration, meaning that data streams from different detectors are analysed without artificial time shifts, so that coincident triggers can correspond to genuine GW events. Candidate triggers are then ranked using the pipeline's detection statistic, and those with a FAR < 2/year are identified as potential gravitational-wave events. This procedure yields the set of BBH events that form the basis of the O3 catalogue considered in this work. Before we open the box [1], let us first review the GWTC events and determine which ones are inside the SAGE training distribution, and thus label what we can reasonably expect and not expect to detect.

5.3.1 Expected to Re-discover

To assess the ability of SAGE to recover known BBH events, we first define the search region based on the training distribution used for the network. We consider component masses in the range $7\,M_\odot \leq m_1, m_2 \leq 50\,M_\odot$, with the requirement that $m_2 \leq m_1$. This region reflects the parameter space over which the network was trained and is the domain in which we expect reliable detection performance.

We obtain the posterior samples of GWTC events from O3 for processing[2]. We classify an event as OOD if the majority of its posterior samples lie outside this defined search region. Concretely, for each event, we compute the fraction of posterior samples that satisfy the search-region constraints. If more than 50% of the samples lie outside the SAGE search region, the event is flagged as OOD. This procedure allows us to distinguish between events that the network is expected to detect reliably and those whose parameters extend beyond the training domain, where performance may be degraded. For this study, we restrict our analysis to BBH events reported in GWTC O3a and O3b [3, 5] with FARs less than 2 per year and `p_astro` > 0.5 (no GWTC marginal events were included; only confident events), enabling a focused evaluation of SAGE's sensitivity across the defined mass range.

In order to visualise the relation between the training-domain search region of SAGE and the population of BBH events from O3, we plot the component-mass posterior samples of all considered events over the (m_1, m_2) plane in Fig. 5.7 for O3a and Fig. 5.8 for O3b. Each event is represented by a kernel density estimate of its posterior, and a 98.9% credible-level contour is shown. Events whose posteriors predominantly lie outside the SAGE search region are flagged as out-of-distribution and drawn in light grey, while events within the search region (meaning at least 50% of the posterior samples are within the search region) are coloured. The figure also highlights the bounds of the SAGE search region, corresponding to $7, M_\odot \leq m_1, m_2 \leq 50, M_\odot$ and $m_2 \leq m_1$, with diagonal lines indicating $q = 1$ and

[1] In GW searches, "open the box" refers to unblinding the zerolag search after all analysis choices are fixed, by analogy with Pandora's box, as this step irreversibly reveals any candidate signals and their consequences.

[2] Refer to https://zenodo.org/records/8177023 for more details (last accessed on Sept 2025)

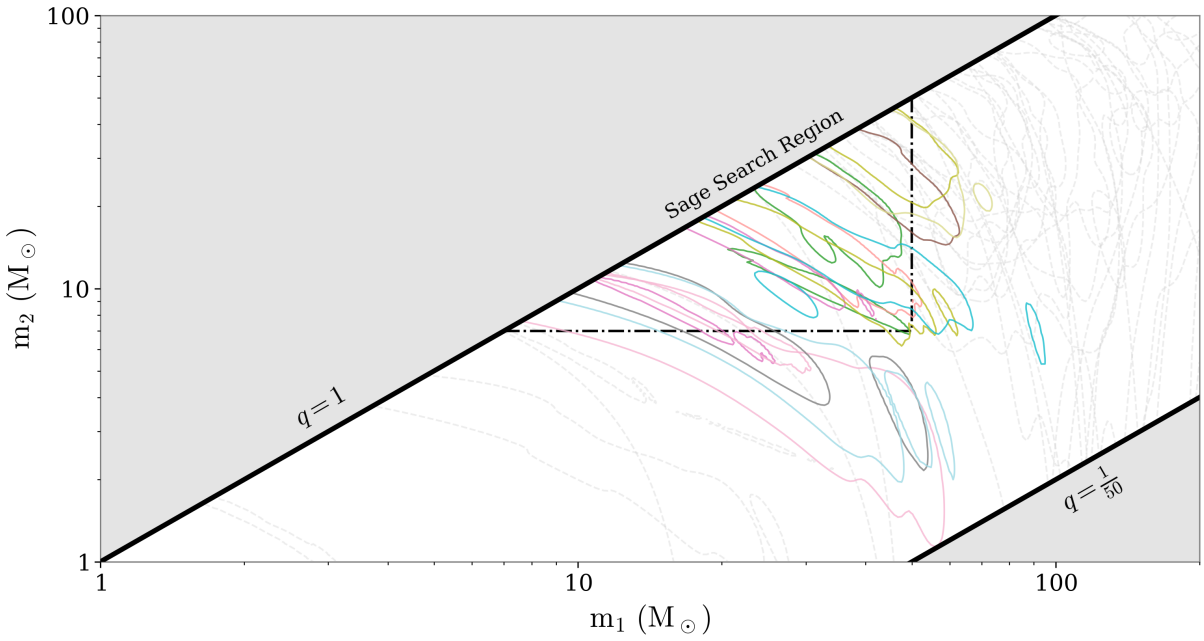


Figure 5.7: Posterior distributions of BBH component masses in O3a, visualised using kernel density estimate contours at the 99.8% credible level. Events lying predominantly outside the Sage search region are shown in light grey to indicate out-of-distribution status, while events within the search bounds are coloured.

$q = 1/50$ for reference. A dashed triangle further demarcates the allowed region for the search. This visualisation demonstrates which events the Sage pipeline is expected to detect reliably, and identifies those that lie outside the network's training domain. In total, there are 27 unique candidates that are defined as being within the training distribution, within the GWTC-3 catalogue and have a FAR less than 2 per year. There are 38 unique candidates outside the training distribution, within GWTC-3 with a FAR less than 2 per year. If our definition of OOD is correct, we can expect to find 27 candidates from our search below a FAR of 2 year pear. Some BBH events were dismissed since their FARs in the GWTC catalogue were > 2/year.

5.3.2 Opening the Box

We now compare the zero-lag triggers obtained from our Sage searches to the background distributions estimated from time slides for each of the three search configurations. The zero-lag triggers correspond to coincident events in the real detector data, whereas the background distributions represent the expected triggers from purely noise fluctuations. By overlaying the zero-lag triggers on the corresponding background distributions, we can visually identify which triggers are likely of astrophysical origin by looking at the high ranking statistic tail.

Each search configuration produces a distinct set of triggers due to differences in the trained parameter distributions and network sensitivity. We quantify the total number of unique events recovered across all searches, taking care to account for redundant detections. This allows us to assess the com-

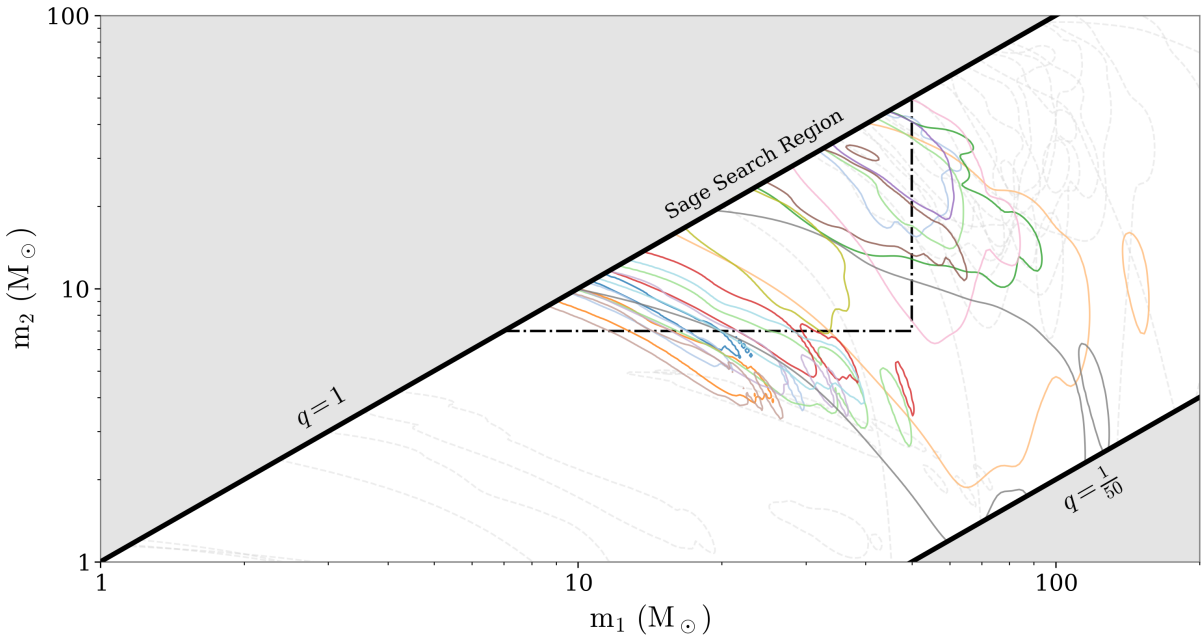


Figure 5.8: Posterior distributions of BBH component masses in O3b, visualised using kernel density estimate contours at the 99.8% credible level. Events lying predominantly outside the Sage search region are shown in light grey to indicate out-of-distribution status, while events within the search bounds are coloured.

bined coverage of the Sage searches and identify events that would be missed by individual search configurations alone.

We compare the zero-lag triggers identified by the three Sage search configurations—HL, LV, and HV—to their respective background distributions. The zero-lag triggers represent candidate GW events in real data, while the background distributions are obtained from time-slide analyses and represent the expected noise fluctuations. In total, HL produced 38 unique candidates, LV 13, and HV 19. Across all searches, there are 70 candidate triggers, corresponding to 52 unique events. All candidates correspond to a false alarm rate of less than 2 per year. This means we found an excess of 25 candidates than expected. Although this is exciting, these candidates could be one of many things: [i] events within GWTC and outside the training distribution, [ii] previously identified events that lie outside the GWTC catalogue and either within or outside the Sage training distribution [299], [iii] previously identified GWTC subthreshold events[3], [iv] non-Gaussian noise artefacts, and if we are lucky [v] new gravitational wave events.

Figure 5.9 shows the distribution of ranking statistics for both zero-lag triggers and background events. Each configuration is plotted separately, with zero-lag triggers in solid lines and backgrounds in dashed lines. This comparison allows visual identification of events that significantly exceed the expected noise distribution and illustrates the relative sensitivity of each search configuration.

[3]Refer to the GWTC data release documentation (https://gwosc.org/eventapi/html). Last accessed on Sept 2025.

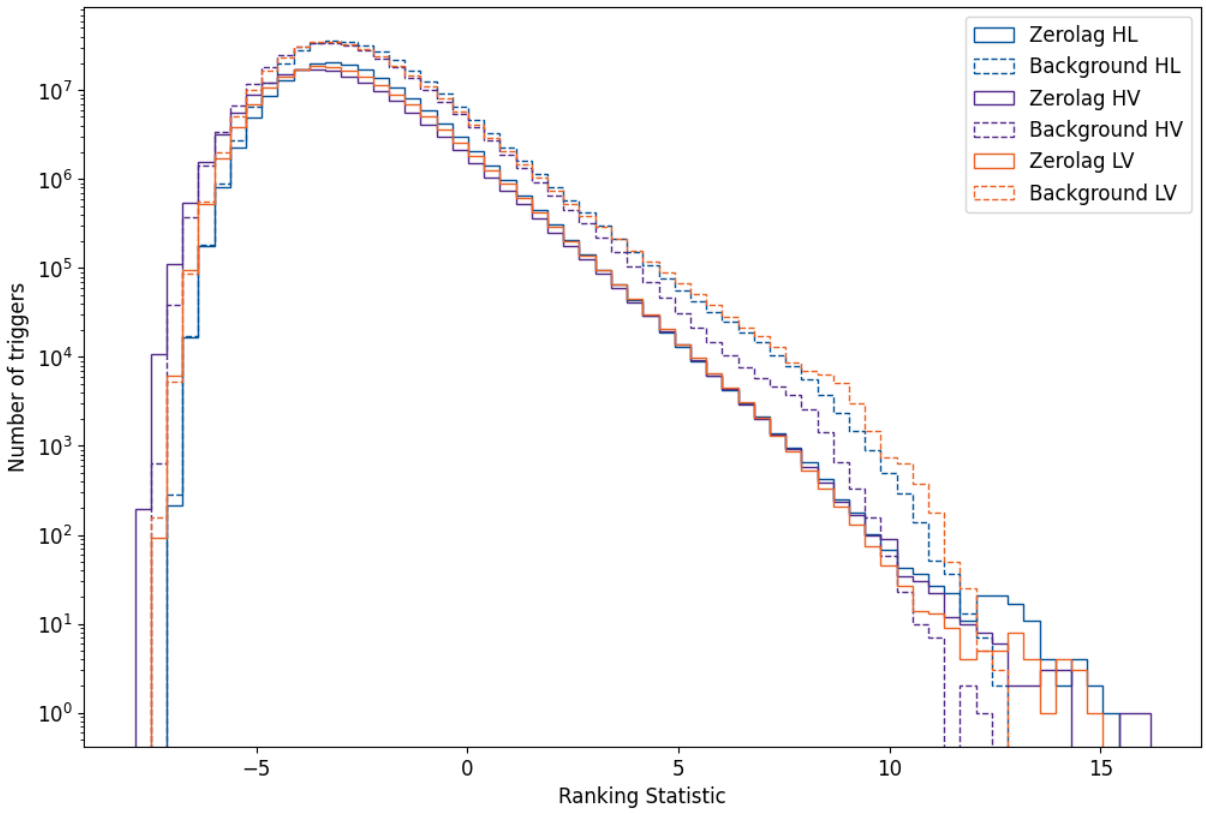


Figure 5.9: Histograms of ranking statistics for zero-lag triggers and background events for the three SAGE search configurations: HL (blue), LV (orange), and HV (purple). Solid lines show the zero-lag triggers, while dashed lines show the background distributions estimated from time slides.

5.4 Catalogue Coincidence

In this section, we examine the correspondence between the candidate events identified by the SAGE searches and existing gravitational-wave catalogues. We assess which triggers coincide with known events in GWTC, which are subthreshold detections not included in the published catalogues, and which lie within external catalogues. Additionally, we discuss events that were missed by the searches and analyse possible reasons for their non-detection, including ranking statistic placement, learning biases, signal morphology, and out-of-distribution effects relative to the SAGE search region.

5.4.1 Coincidence with GWTC, Subthreshold and Community Catalogues

We find a total of 43 events that coincide with GWTC, plus 1 subthreshold candidate flagged by GSTLAL and find no coincidence with unique events presented in external catalogues that are not present in GWTC as provided in [299]. From external (non-GWTC) catalogues there are 5 candidates that lie within our training distribution; 3 of those 5 were reported in their respective external catalogues with FAR less than 2 per year, yet none of those three were recovered by SAGE. Focusing on in/out-of-distribution recovery within the GWTC catalogue: of the events that we classify as in-distribution,

we miss 6 out of 27; for out-of-distribution events, we miss 17 out of 38 (finding 42 GWTC events that were reported with a FAR < 2/year; we will discuss the 1 remaining event soon). These results imply that our in/out-of-distribution test is not perfectly predictive of recoverability — the shortfall is plausibly driven by learning biases in the network and by its extrapolation behaviour outside the training manifold, rather than by a simple boundary effect alone.

Tables 5.4.1 and 5.4.1 summarises the coincidence study between our search and the events reported in GWTC. We separate the binary black hole events into in-distribution and out-of-distribution categories according to the training support of our model. For each event, we list its GPS time, false-alarm rate, astrophysical probability, network matched filter SNR, and source-frame chirp mass (all values are obtained from the GWOSC event list and not obtained from Sage). The Sage FARs for all events marked as found are < 2/year. The final column indicates whether the event was recovered in our search. Rows are highlighted in green for recovered events and in red for missed events. This table illustrates both the successful recoveries and the limitations due to model biases.

Table 5.1: Catalogue coincidence showing in-distribution BBH events from GWTC in O3.

Event	GPS Time	FAR [1/year]	p_astro	SNR	$\mathcal{M}_c^{\mathrm{src}}$ [$M_\odot$]	Found
GW190408_181802-v2	1238782700.2	1e-05	1.0	14.6	$18.5^{1.9}_{-1.2}$	Yes
GW190412_053044-v4	1239082262.1	1e-05	1.0	19.8	$13.3^{0.5}_{-0.5}$	Yes
GW190512_180714-v2	1241719652.4	1e-05	1.0	12.7	$14.6^{1.4}_{-0.9}$	Yes
GW190630_185205-v2	1245955943.1	1e-05	1.0	16.4	$25.1^{2.2}_{-2.1}$	Yes
GW190707_093326-v2	1246527224.1	1e-05	1.0	13.1	$8.4^{0.6}_{-0.4}$	Yes
GW190708_232457-v2	1246663515.3	0.00031	0.99965	13.4	$13.1^{0.9}_{-0.6}$	Yes
GW190720_000836-v2	1247616534.7	1e-05	1.0	10.9	$9.0^{0.4}_{-0.8}$	No
GW190725_174728-v1	1248112066.4	0.46	0.96021	9.1	$7.4^{0.5}_{-0.5}$	No
GW190728_064510-v2	1248331528.5	1e-05	1.0	13.1	$8.6^{0.6}_{-0.3}$	Yes
GW190828_063405-v2	1251009263.7	1e-05	1.0	16.5	$24.6^{3.6}_{-2.0}$	Yes
GW190828_065509-v2	1251010527.8	3.5e-05	0.99998	10.2	$13.4^{1.4}_{-1.0}$	Yes
GW190915_235702-v2	1252627040.6	1e-05	1.0	13.1	$24.4^{3.0}_{-2.3}$	Yes
GW190925_232845-v1	1253489343.1	0.0072	0.98849	9.7	$15.6^{1.1}_{-1.1}$	No
GW190930_133541-v2	1253885759.2	0.012	0.99785	9.7	$8.5^{0.5}_{-0.4}$	No
GW191204_171526-v1	1259514944.0	1e-05	0.99	17.4	$8.56^{0.41}_{-0.28}$	Yes
GW200311_115853-v1	1267963151.3	1e-05	0.99	17.8	$26.6^{2.4}_{-2.0}$	Yes
GW191216_213338-v1	1260567236.4	1e-05	0.99	18.6	$8.33^{0.22}_{-0.19}$	Yes
GW191129_134029-v1	1259070047.1	1e-05	0.99	13.1	$7.31^{0.43}_{-0.28}$	Yes
GW191103_012549-v1	1256779567.5	0.46	0.94318	8.9	$8.34^{0.66}_{-0.57}$	Yes
GW200129_065458-v1	1264316116.4	1e-05	0.99	26.8	$27.2^{2.1}_{-2.3}$	Yes
GW200112_155838-v1	1262879936.0	1e-05	0.99	19.8	$27.4^{2.6}_{-2.1}$	Yes
GW191105_143521-v1	1256999739.9	0.012	0.99	9.7	$7.82^{0.61}_{-0.45}$	Yes
GW191215_223052-v1	1260484270.3	1e-05	0.99	11.2	$18.4^{2.2}_{-1.7}$	Yes
GW200202_154313-v1	1264693411.5	1e-05	0.99	10.8	$7.49^{0.24}_{-0.2}$	No
GW200302_015811-v1	1267149509.5	0.11	0.90643	10.8	$23.4^{4.7}_{-3.0}$	Yes
GW200225_060421-v1	1266645879.3	1.1e-05	0.99	12.5	$14.2^{1.5}_{-1.4}$	Yes
GW200316_215756-v1	1268431094.1	1e-05	0.99	10.3	$8.75^{0.62}_{-0.55}$	No

Table 5.2: Catalogue coincidence showing out-of-distribution BBH events from GWTC in O3.

Event	GPS Time	FAR [1/year]	p_astro	SNR	$\mathcal{M}_c^{\rm src}$ [$M_\odot$]	Found
GW190413_052954-v2	1239168612.5	0.82	0.9275	9.0	$24.5^{4.6}_{-3.4}$	Yes
GW190413_134308-v2	1239198206.7	0.18	0.99058	10.6	$33.3^{7.8}_{-6.3}$	Yes
GW190421_213856-v2	1239917954.2	0.0028	0.99792	10.7	$31.4^{6.0}_{-4.6}$	Yes
GW190425_081805-v3	1240215503.0	0.034	0.78349	12.4	$1.44^{0.02}_{-0.02}$	No
GW190503_185404-v2	1240944862.2	1e-05	1.0	12.2	$29.3^{4.5}_{-4.4}$	Yes
GW190513_205428-v2	1241816086.7	1.3e-05	0.99998	12.5	$21.8^{3.8}_{-2.2}$	Yes
GW190517_055101-v2	1242107479.8	0.00035	1.0	10.8	$26.5^{4.0}_{-4.2}$	Yes
GW190519_153544-v2	1242315362.3	1e-05	1.0	15.9	$44.3^{6.8}_{-7.5}$	Yes
GW190521_030229-v4	1242442967.4	0.0013	0.99989	14.3	$63.3^{19.6}_{-14.6}$	No
GW190521_074359-v2	1242459857.4	1e-05	1.0	25.9	$32.8^{3.2}_{-2.8}$	Yes
GW190527_092055-v2	1242984073.7	0.23	0.84784	8.0	$23.9^{6.8}_{-3.9}$	Yes
GW190602_175927-v2	1243533585.0	1e-05	1.0	13.2	$48.0^{9.5}_{-9.7}$	No
GW190620_030421-v2	1245035079.3	0.011	0.98748	12.1	$38.1^{8.5}_{-7.2}$	No
GW190701_203306-v2	1246048404.5	0.0057	0.99986	11.2	$40.2^{5.4}_{-5.0}$	No
GW190706_222641-v2	1246487219.3	5e-05	1.0	13.4	$45.6^{13.0}_{-9.1}$	Yes
GW190719_215514-v2	1247608532.9	0.63	0.91976	7.9	$22.8^{8.3}_{-4.3}$	Yes
GW190727_060333-v2	1248242631.9	1e-05	1.0	11.7	$29.4^{4.6}_{-3.7}$	Yes
GW190731_140936-v2	1248617394.6	0.33	0.83427	8.8	$29.7^{7.4}_{-5.3}$	No
GW190803_022701-v2	1248834439.8	0.073	0.97383	9.3	$27.6^{5.4}_{-3.8}$	Yes
GW190805_211137-v1	1249074715.3	0.63	0.95372	8.1	$31.9^{8.8}_{-6.3}$	No
GW190814_211039-v3	1249852257.0	1e-05	1.0	25.3	$6.11^{0.06}_{-0.05}$	Yes
GW190910_112807-v2	1252150105.3	0.0029	0.99669	14.5	$33.5^{4.2}_{-4.1}$	No
GW190917_114630-v1	1252756008.0	0.66	0.76643	8.3	$3.7^{0.2}_{-0.2}$	No
GW190924_021846-v2	1253326744.8	1e-05	1.0	12.0	$5.8^{0.2}_{-0.2}$	No
GW190926_050336-v1	1253509434.0	1.1	0.54334	8.1	$24.4^{9.0}_{-4.9}$	No
GW190929_012149-v2	1253755327.4	0.16	0.87042	9.7	$35.6^{10.2}_{-7.4}$	No
GW200210_092254-v1	1265361792.9	1.2	0.54462	8.4	$6.56^{0.38}_{-0.4}$	No
GW191127_050227-v1	1258866165.5	0.25	0.73996	9.2	$29.9^{11.7}_{-9.1}$	Yes
GW200208_130117-v1	1265202095.9	0.00031	0.99	10.8	$27.7^{3.7}_{-3.1}$	Yes
GW200128_022011-v1	1264213229.9	0.0043	0.99	10.6	$32.0^{7.5}_{-5.5}$	Yes
GW200224_222234-v1	1266618172.4	1e-05	0.99	20.0	$31.1^{3.3}_{-2.7}$	Yes
GW200219_094415-v1	1266140673.1	0.00099	0.99	10.7	$27.6^{5.6}_{-3.8}$	Yes
GW191109_010717-v1	1257296855.2	0.00018	0.99	17.3	$47.5^{9.6}_{-7.5}$	No
GW191230_180458-v1	1261764316.4	0.05	0.96159	10.4	$36.5^{8.2}_{-5.6}$	No
GW200209_085452-v1	1265273710.1	0.046	0.97236	9.6	$26.7^{6.0}_{-4.2}$	Yes
GW200115_042309-v2	1263097407.7	1e-05	0.99	11.3	$2.43^{0.05}_{-0.07}$	No
GW200216_220804-v1	1265926102.8	0.35	0.77277	8.1	$32.9^{9.3}_{-8.5}$	No
GW191222_033537-v1	1261020955.1	1e-05	0.99	12.5	$33.8^{7.1}_{-5.0}$	Yes

Improved FAR for GW191204_110529-v1: In addition to the 42 GWTC events that were *reported* at a FAR of < 2/year and *found* at a Sage FAR of < 2/year, we find one GWTC event that was reported at a FAR of 3.3/year with a Sage FAR of < 1/year. GW191204_110529-v1 was reported in GWTC with a $p_{\rm astro}$ of 0.74239, a network matched filter SNR 8.9 and a chirp mass of $19.8^{+3.6}_{-3.2}$ $M_\odot$. This showcases the potential of Sage in detecting candidates at a higher confidence than matched filtering pipelines that were used to run the O3 search, especially in the low SNR regime. With this we

have accounted for all 43 events that were in-distribution, within GWTC and found by Sage below the FAR threshold of 2/year.

Subthreshold Coincidence: Here, we report one new event that was present in the subthreshold catalogue, making this a re-discovery. A single subthreshold candidate reported by the GstLAL All-Sky search [100] with a combined false-alarm rate of 1.2×10^{-5} per year was detected by Sage at a FAR of < 2/year. The parameters of this event presented in this section were all provided with the GstLAL All-Sky search subthreshold candidate list. The event occurs at GPS time 1251502439.848 and is observed in coincidence across H1 and V1. The inferred component masses are approximately $m_1 \approx 43.4\,M_\odot$ and $m_2 \approx 22.6\,M_\odot$, corresponding to a chirp mass of $\mathcal{M}_c \approx 26.9\,M_\odot$. The network SNR is 7.97, with the individual detectors reporting SNRs of 6.83 (H1) and 4.10 (V1). While the false-alarm rate is sufficiently low to suggest astrophysical plausibility, the candidate was ultimately rejected. The most likely explanations are that the SNR fell below the canonical threshold of 8 or that the inferred GstLAL p_{astro} was too low to pass catalogue inclusion criteria. More details about this candidate is provided in table 5.3. The estimated maximum a posteriori (MAP) value of the component masses for this event is safely within the Sage search region. We will refer back to the details of this table after we perform parameter estimation on this candidate.

Table 5.3: Details of the subthreshold candidate identified by GstLAL. The same source parameters are reported in both H1 and V1 detectors, while the recovered SNR and χ^2 differ. We distinguish the spin parameter and the chi-squared statistic parameter by using χ^2_{stat} for the latter.

IFO	m_1 [$M_\odot$]	m_2 [$M_\odot$]	$\mathcal{M}_c$ [$M_\odot$]	$\chi_{\text{eff},1}$	$\chi_{\text{eff},2}$	SNR	χ^2_{stat}	End Time (s)	Coa. Phase
H1	43.36	22.56	26.95	-0.60	0.55	6.83	0.83	1251502439.8	0.41
V1	43.36	22.56	26.95	-0.60	0.55	4.10	0.89	1251502439.8	-2.50

5.4.2 Missed Events

In addition to the recovered events, our search also failed to identify a number of GW candidates. Within GWTC, six in-distribution BBH events were missed, and a further three in-distribution candidates from the external catalogues were not recovered. To investigate the underlying cause, we examined the posterior distributions of these missed events and plotted their component masses in the m_1–m_2 plane. The resulting distribution shows that the missed events predominantly occupy the region of low m_1, low m_2, and consequently low chirp mass. This behaviour is consistent with the learning biases inherent in this particular Sage model. This behaviour is similar to that seen in chapter 4 for the `Sage-Broad` model. Figure 5.10 shows the posteriors in the $m_1 - m_2$ space for the missed events from within GWTC at FARs less than 2 per year. The same is depicted for in distribution events from external catalogues in Fig. 5.11. It is clear to see that most of the missed events by Sage on both Figs. 5.10 and 5.11 are in the low chirp mass region, indicated a bias. However, there are two events that seem to be in the prime mass space for detection, but are yet not detected. Let us now discuss the possible reasons for these missed events.

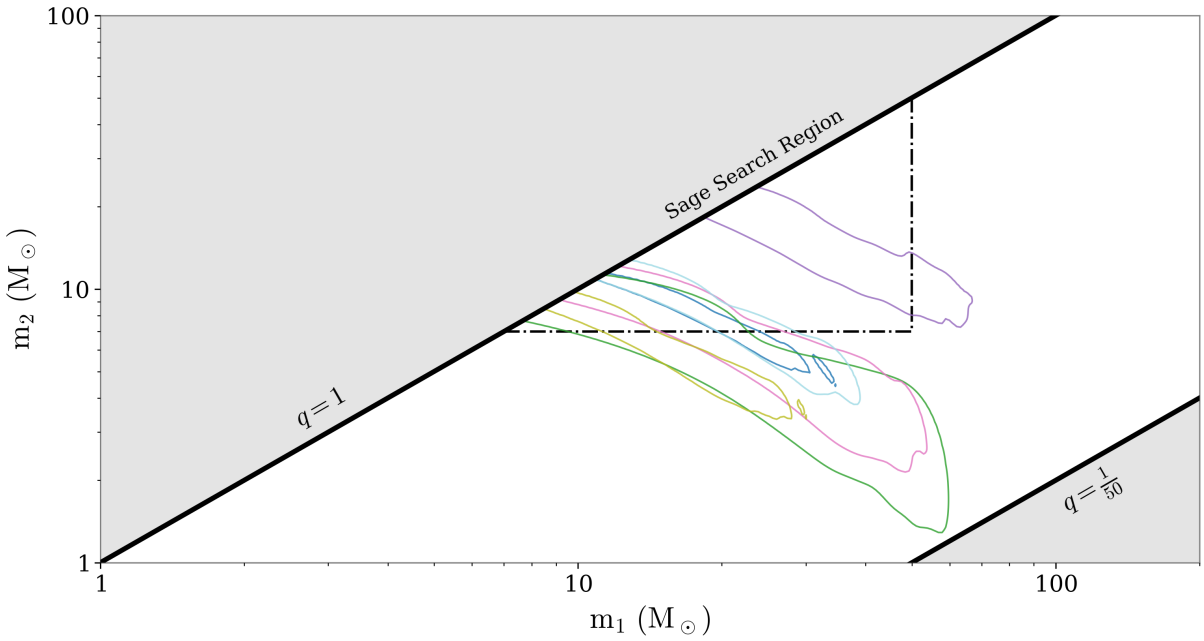


Figure 5.10: Posterior distributions of BBH component masses of in-distribution events that were missed in O3 from the **GWTC catalogue** at a FAR of less than 2 per year, visualised using kernel density estimate contours at the 99.8% credible level.

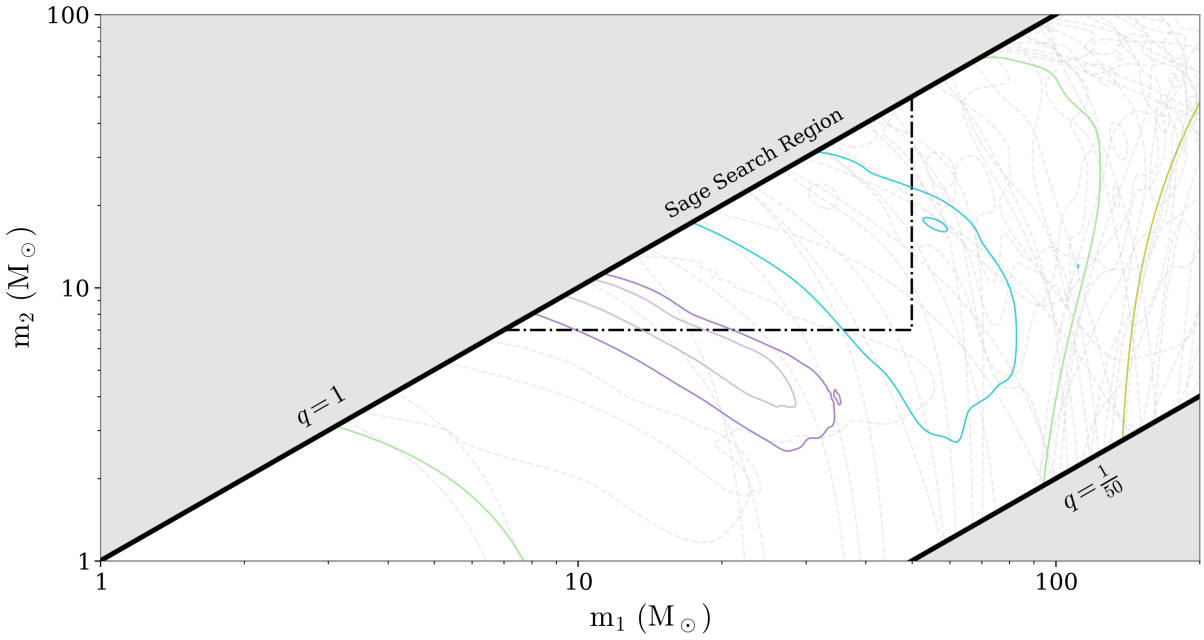


Figure 5.11: Posterior distributions of BBH component masses of in-distribution events that were missed in O3 from **external catalogues** at a FAR of less than 2 per year, visualised using kernel density estimate contours at the 99.8% credible level.

Why did we miss these events?

Search Sensitivity and Residual Biases: Although our search is very sensitive, there remain indications of residual biases even after applying physics-informed representation learning and bias mitigation (as seen in chapter 4). These residual biases are particularly noticeable at low chirp masses, corresponding to longer-duration signals, which slightly reduces our sensitivity in this regime (as is the case with the `Sage-Broad` model used in this chapter for the search). This effect is evident in our observation that we predominantly miss low chirp mass events (see Figs. 5.10 and 5.11). The one slightly higher chirp mass event that was missed in Fig. 5.10 and possibly one or more in Fig. 5.11 (the blue posterior satisfies this case; but given the broadness two of the posteriors here, it is not possible to accurately classify them) is likely attributable to either low SNR or anomalous noise characteristics around the event, causing Sage to downrank it. This interpretation also depends on the background estimation procedure. Importantly, while these events were missed, they were not associated with extremely high FAR values, and were only modestly downranked.

Glitches and Non-Gaussian Noise Artefacts: Some missed events could be attributed to the presence of nearby glitches. Sage analyses 12-second-long segments, unlike other machine learning pipelines that use much shorter windows ($O(1)$ seconds), making our search more susceptible to glitches or local noise fluctuations that may downweight candidates. To maintain a standalone approach in rejecting glitches, we did not rely on external pipelines for glitch mitigation. If any low-FAR glitches were present, they were left in the background distribution rather than removed, which could reduce detection efficiency for nearby true signals.

Differences in Ranking Statistics: Traditional matched filtering pipelines combine multiple statistics to ensure consistency with astrophysical expectations. These include, for example, the χ^2 test to confirm the signal morphology is GW-like and cross-detector consistency measures to improve ranking. While machine learning is designed to approximate these behaviors, such consistency is not guaranteed, and it can vary across the parameter space, particularly in regions affected by biases. As a result, some events that would pass traditional ranking may be downranked by Sage.

Data Quality Flags: In some cases, the data may exhibit poor quality or anomalous PSD features. Although our network is trained on a wide range of PSDs and should typically be robust, certain pathological segments may lie outside the network's learned distribution. In these cases, Sage may downrank an event simply because it cannot reliably extrapolate to such data, resulting in missed detections.

Effects of learning biases: Although Sage operates on a constrained search parameter space, this can occasionally contribute to missed detections. Compared to traditional matched filtering, our smaller parameter space reduces the false alarm threshold for a given FAR, but residual biases—especially in low chirp mass regions or in the presence of longer signals—can further limit sensitivity. Interestingly, other ML pipelines are generally more biased and thus effectively operate on an even more constrained parameter space. In principle, this should allow them to find more signals than Sage, but our approach

to noise rejection and bias mitigation enables SAGE to outperform other pipelines. This likely explains why we miss the one new claimed event by ARESGW in [210].

We plan to investigate each missed event in detail for our publication to obtain a fuller picture and place a concrete reason for missing them. We can hopefully improve SAGE with this future analysis.

5.5 Discovering New Events in O3!

Having established the recovery performance of our search on events already catalogued in GWTC, we now turn to the question of new discoveries. The O3 run contains a large number of triggers beyond those included in public catalogues, many of which originate from instrumental artefacts or background noise. Our aim here is to examine the population of previously unreported candidates identified by our pipeline, assess their statistical significance, and evaluate whether any can be regarded as genuine GW events. In doing so, we seek to determine whether machine learning–based methods can extend beyond reproducing existing catalogues to provide new astrophysical discoveries.

5.5.1 Events with False Alarm Rate < 2 per year

In total, we identify seven candidate events with a false alarm rate below two per year that are not present in the GWTC catalogues, in external catalogues, or among subthreshold events. Furthermore, none of these candidates coincide with Gravity Spy glitch times, suggesting that they are not trivially explained by known instrumental artefacts. In this section, we examine these candidates in detail to assess their astrophysical plausibility and determine which, if any, could constitute previously undiscovered gravitational-wave events.

Table 5.5.1 summarises the new candidate GW events identified by the SAGE search in O3 that are not present in GWTC, external catalogues, or subthreshold lists. The event names are in the following format: SC (SAGE Candidate) followed by GPS Time of merger (in ISO8601). In total, nine candidates are found with a false alarm rate of less than 2 per year (the event detection threshold chosen for this work; however all new candidates in table 5.5.1 were found at FARs < 1/year). The 2-IFO network association, SAGE ranking statistic $\mathcal{R}_{\&}$, the loudest background trigger in the corresponding 2-IFO background $\mathcal{R}_{\&,\mathrm{bg}}^{2/\mathrm{year}}$, and SAGE predicted point-estimate of source-frame chirp masses $\mathcal{M}_c^{\mathrm{pred}}$ are listed for each candidate. We also provide the `p_astro` values estimated from one method: estimated from the SAGE training distribution (denoted as train). In the future, with more work, we will also compute this using an injection study built from the GWTC-3 astrophysical signal distribution.

Likely to be Glitches: During our examination of the new candidates, we identified one trigger whose SNR and ranking statistic suggested a potential GW signal. To characterise its origin, we produced a time–frequency representation using a Q-transform. A quick visual inspection reveals that SAGE candidate SC190617_193742 is most definitely a "Whistle" type glitch (see Fig. 5.12 for confirma-

Table 5.4: New candidate events identified by Sage in O3. Not all of them might be GW. We will move these candidates to an event status once we complete parameter estimation and create a separate table summarising the new events.

Event	FAR [1/yr]	$p_{\text{astro}}^{\text{train}}$	$\mathcal{R}_s$	$\mathcal{R}_{\text{bg}}^{2/\text{year}}$	$\mathcal{M}_c^{\text{pred}}$ [$M_\odot$]	2-ifo
SC200103_214329	< 1	> 0.99	13.078	12.438	35.41	LV
SC190421_153038	< 1	> 0.99	12.312	11.883	12.45	HV
SC190502_113616	< 1	> 0.99	11.961	11.883	7.25	HV
SC190522_120915	< 1	> 0.99	12.179	11.883	30.95	HV
SC190902_233341	< 1	> 0.99	12.094	11.883	31.36	HV
SC190929_095627	< 1	> 0.99	12.164	11.883	31.20	HV
SC191105_051752	< 1	> 0.99	11.914	11.883	34.50	HV
SC200205_085012	< 1	> 0.99	11.992	11.883	33.19	HV
SC190617_193742	< 1	> 0.99	12.266	12.188	33.78	HL

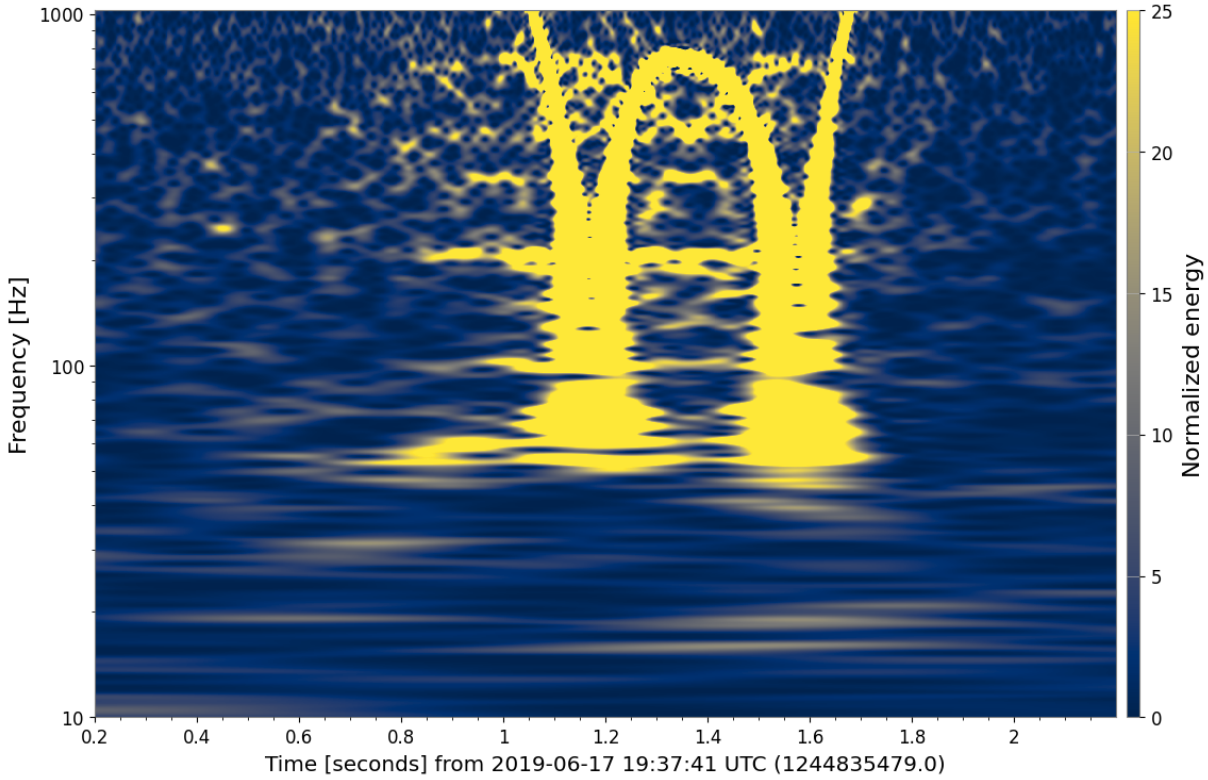


Figure 5.12: Time–frequency representation of the `SC190617_193742` candidate event identified by Sage using a Q-transform. The characteristic whistle pattern indicates that this trigger corresponds to an instrumental glitch rather than an astrophysical GW signal.

tion), a known instrumental or environmental transient. This demonstrates that despite its apparent statistical significance, this candidate is not astrophysical in origin but corresponds to a GW detector glitch. Such visual inspections remain crucial for distinguishing genuine GW signals from instrumental artefacts, particularly for low SNR events or candidates outside previously catalogued populations.

5.5.2 Parameter Estimation

Important Note: The parameter estimation runs were made by Dr Daniel Williams of the University of Glasgow, who will be co-author on our future publication.

We perform Bayesian parameter estimation to infer posterior probability distributions over the source parameters for each new candidate event, using a segment of data surrounding the trigger. The analysis assumes models of GW emission and stationary, Gaussian noise that is uncorrelated across detectors. Default priors are chosen to be agnostic and sufficiently broad to encompass the regions of parameter space supported by the posteriors. Specifically, we adopt uniform priors in the redshifted component masses, uniform spin magnitudes, isotropic spin orientations, isotropic binary orientation, uniform merger time and coalescence phase, isotropic sky location, and a distance prior corresponding to a uniform merger rate in comoving volume and time (similar to that done for GWTC-3 [5]). We use the `IMRPhenomXPHM-SpinTaylor` waveform model to perform all parameter estimation runs. Below, we provide a note on each new candidate (except `SC190617_193742` which was visually confirmed to be a Whistle glitch) based on the results from parameter estimation:

`SC190421_153038`: Refer to Fig. 5.13 for a parameter estimation summary of the candidate event. We see a positive χ_{eff} and χ_p, which are the effective and precessing spins respectively. Concretely, χ_{eff} can be defined as:

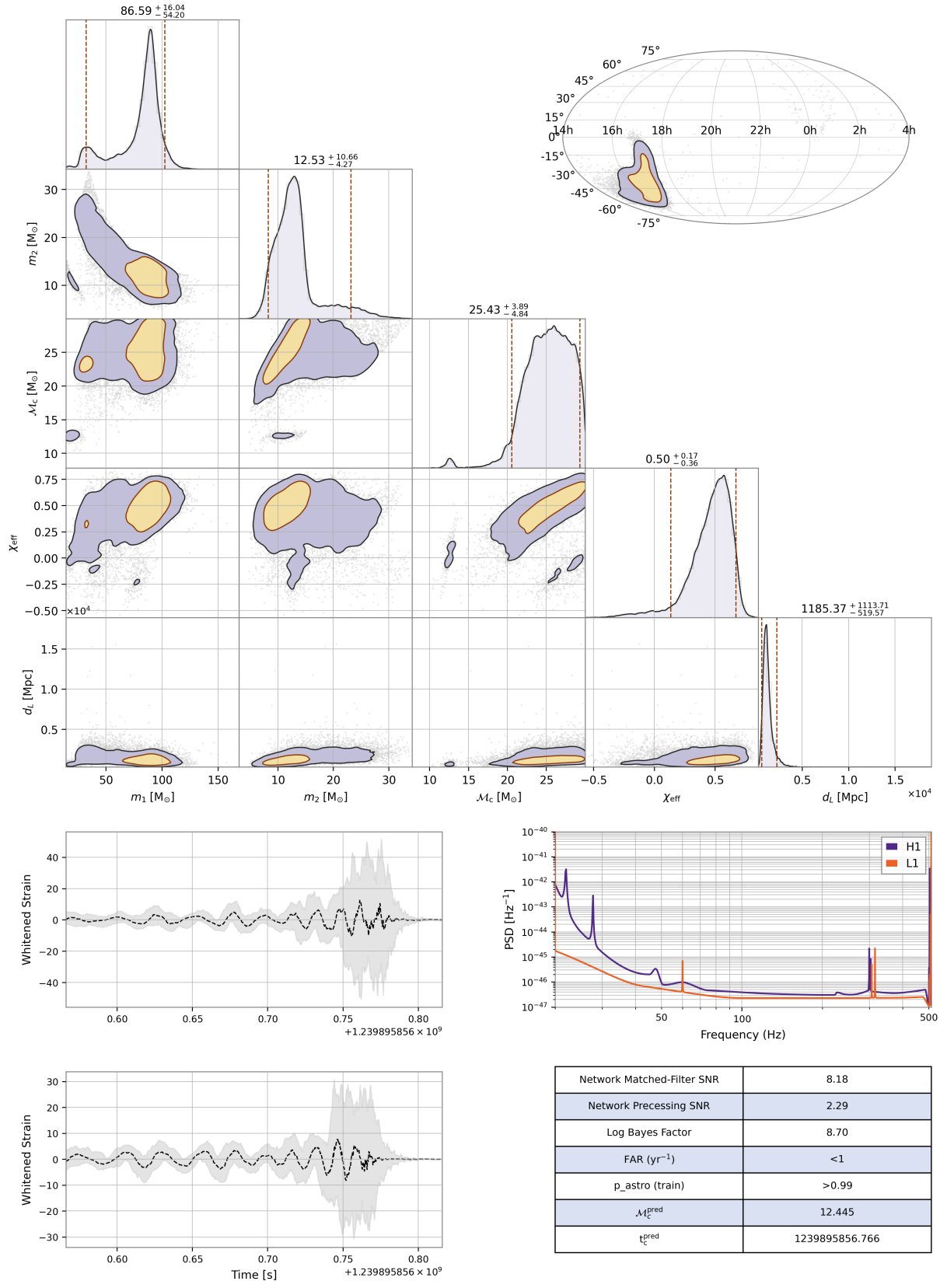
$$\chi_{\text{eff}} = \frac{m_1 \chi_1 \cos\theta_1 + m_2 \chi_2 \cos\theta_2}{m_1 + m_2} \tag{5.27}$$

where m_1 and m_2 are the two component masses, χ_1 and χ_2 are two dimensionless spin magnitudes (which are also represented using symbols a_1 and a_2 as seen in the summary plot), and θ_1 and θ_2 are the tilts. The definition of precessing spin requires a level of detail that is outside the scope of this chapter; for a full definition of precessing spin, please refer to [300] and [301].

The corner plot shows that the spin component a_1 is very high, which could be real but this behaviour could also be caused by the presence of glitch-like features. The posterior on the masses and chirp mass show some bimodality, which needs further investigation. This could be evidence for an overlapping glitch alongside the signal. Re-running the summary on all three detectors might provide a better picture.

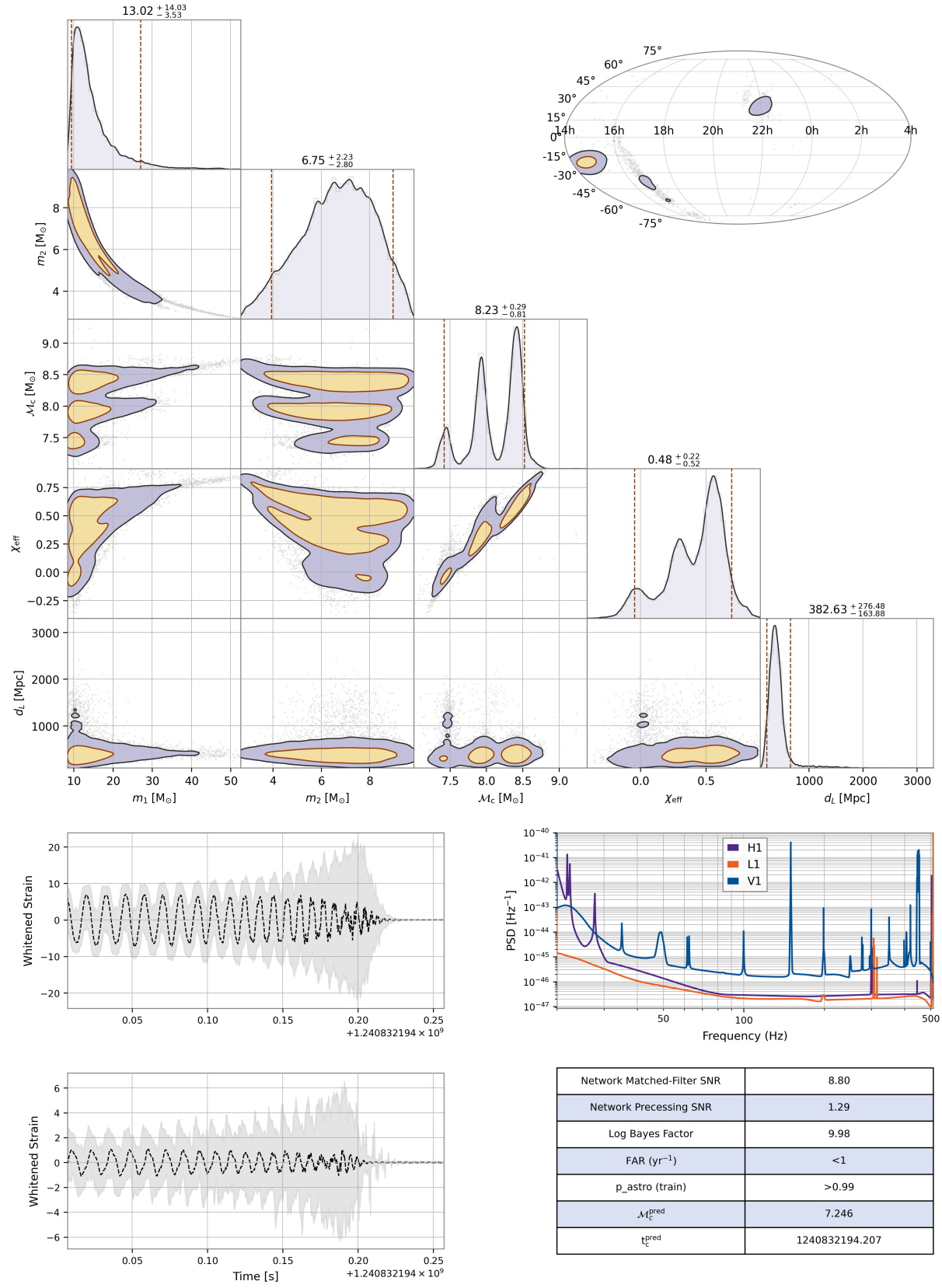
`SC190502_113616`: Refer to Fig. 5.14 for a parameter estimation summary of the candidate event. Chirp mass is clearly multimodal and spin behaviour is non-standard. The Virgo detector PSD behaviour is anomalous, which might cause some unforeseen behaviour as well. This has the lowest chirp mass predicted of all the SAGE candidates. Given that `Sage-Broad`, the version of SAGE used for this search (refer to chapter 4 section 4.3 for more details) has some bias against low chirp mass signals, the trustworthiness of this candidate further reduces. Evidence points to this candidate very likely being a glitch.

`SC190522_130915`: Refer to Fig. 5.15 for a parameter estimation summary of the candidate event.



Network Matched-Filter SNR	8.18
Network Precessing SNR	2.29
Log Bayes Factor	8.70
FAR (yr^{-1})	<1
p_astro (train)	>0.99
$\mathcal{M}_c^{pred}$	12.445
t_c^{pred}	1239895856.766

Figure 5.13: Parameter estimation summary for SAGE candidate `SC190421_153038`. Corner plot shows posteriors for m_1, m_2, χ_{eff}, $\mathcal{M}_c$ and d_L, alongside sky localisation. Bottom-left: `IMRPhenomXPHM` waveforms from 32 posterior samples (median in dashed purple). PSDs for all detectors and key metrics are also shown.



Network Matched-Filter SNR	8.80
Network Precessing SNR	1.29
Log Bayes Factor	9.98
FAR (yr^{-1})	<1
p_astro (train)	>0.99
$\mathcal{M}_c^{\rm pred}$	7.246
$t_c^{\rm pred}$	1240832194.207

Figure 5.14: Parameter estimation summary for SAGE candidate `SC190502_113616`. Corner plot shows posteriors for m_1, m_2, $\chi_{\rm eff}$, $\mathcal{M}_c$ and d_L, alongside sky localisation. Bottom-left: `IMRPhenomXPHM` waveforms from 32 posterior samples (median in dashed purple). PSDs for all detectors and key metrics are also shown.

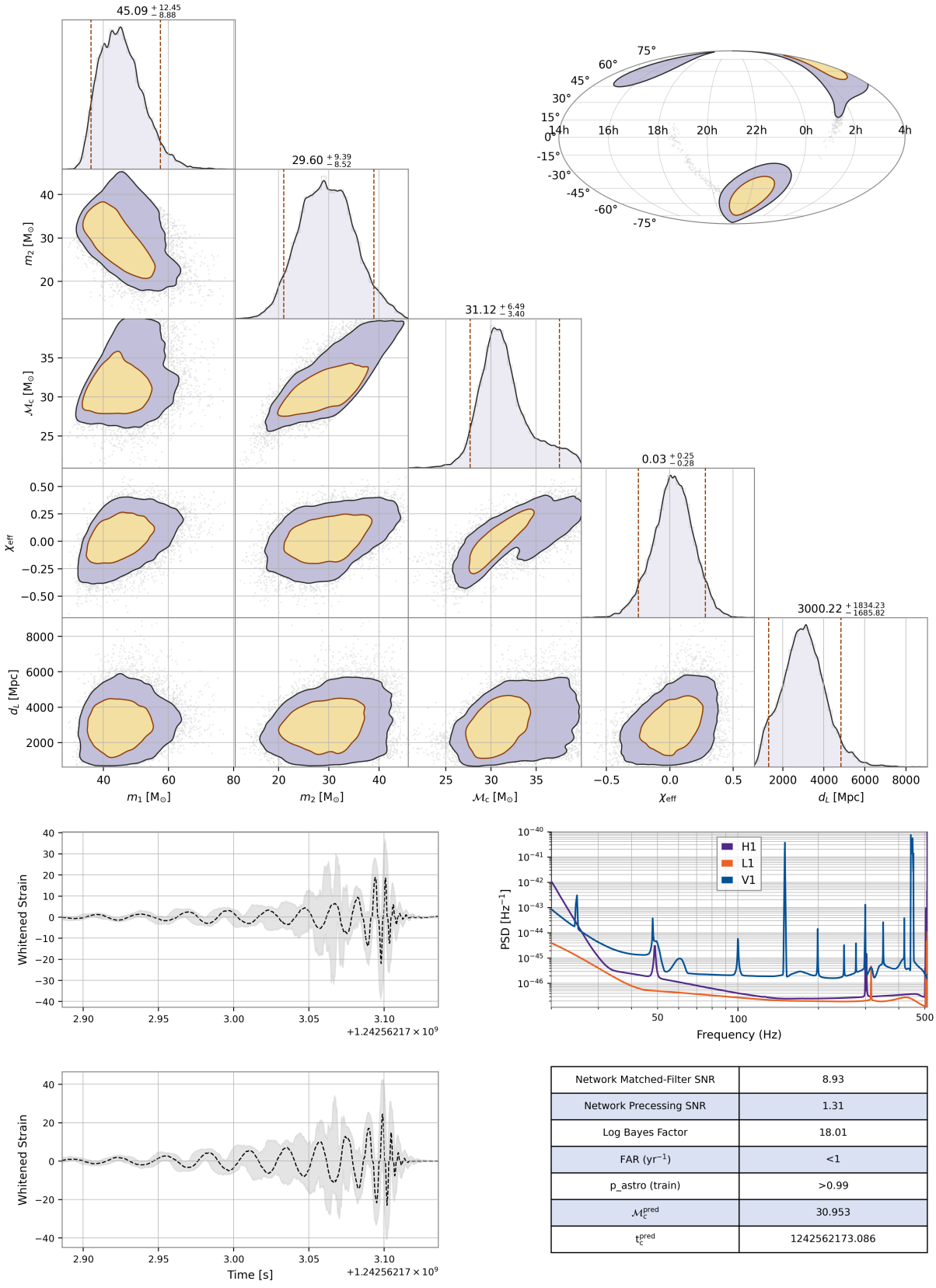
This run is fine with the exception of some weird behaviour from the Virgo PSD. Mild railing was seen in the upper chirp mass, likely due to relatively low network matched filter SNR; this was later fixed with a re-rerun on an expanded chirp mass space. This candidate has the highest log Bayes factor so far of 18.01, which provides a bit more confidence in its significance. Very standard behaviour overall for a typical BBH and thus we claim this as our first new SAGE GW event at an FAR of less than 2 per year: `SGW190522_130915`!

`SC190902_232241`: Refer to Fig. 5.16 for a parameter estimation summary of the candidate event. This run too does not show any anomalous behaviour. We see that the event favours a low $\chi_{\rm p} = 0.35^{+0.40}_{-0.25}$, which is interesting (see Fig. 5.17); but the precessing SNR is still quite low. Similar to `SGW190522_130915`, we initially saw railing in the upper end of the chirp mass; later mitigated with a re-rerun. This candidate has an even higher log Bayes factor of 27.89. Overall, its behaviour is standard, typical of a BBH of this chirp mass and thus it becomes our second new claim of a GW at a FAR less than 2 per year: `SGW190902_232241`. However, we cannot fully claim this as a new discovery since GSTLAL has already marked this candidate as one of their sub-threshold events (see section 5.4.2). Although we find this candidate at a lower FAR than GSTLAL, we can only claim this one as a re-discovery; but a new GW event nonetheless!

`SC190929_095627`: Refer to Fig. 5.18 for a parameter estimation summary of the candidate event. The summary is mostly typical of a standard BBH, but given the lower network matched filter SNR comapared to the previous two events and the lower log Bayes factor of 8.55, the posteriors were broader and required multiple re-runs to avoid railing issues in the upper end of the chirp mass. Other behaviour matched a vanilla BBH and we claim this as our third new event at a FAR less than 2 per year: `SGW190929_095627`!

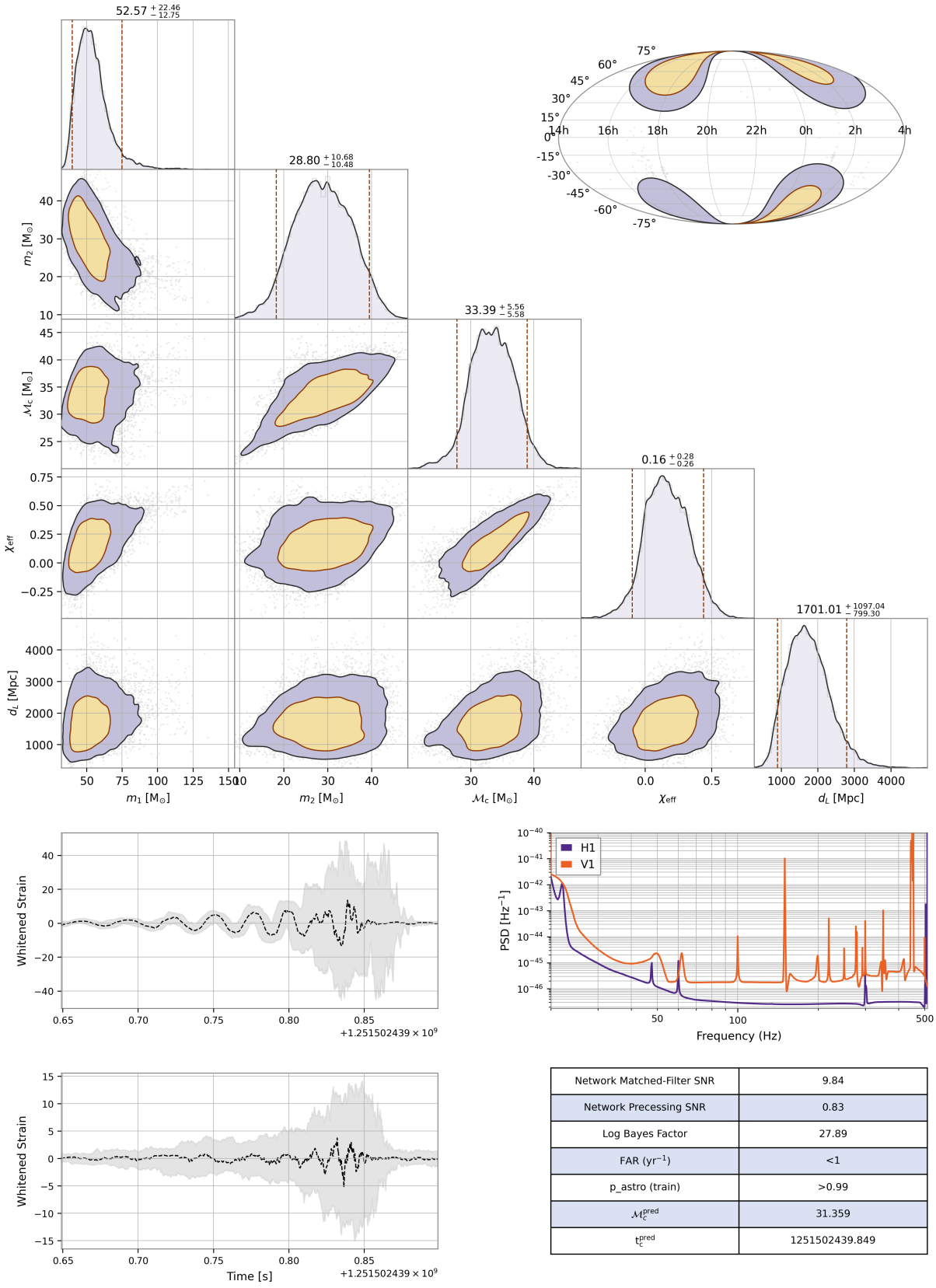
`SC191105_051753`: Refer to Fig. 5.19 for a parameter estimation summary of the candidate event. We see a clear bimodal m_1 and m_2; indicative of a glitch, but possibly alongside a signal. Further analysis of this candidate is required, likely a parameter estimation run along a glitch characterisation and subtraction procedure. We plan on investigating this further for the upcoming publication. The ability to identify weak signal that are at close proximity to glitches would be a highlight of the capabilities of ML given that matched filtering would very likely downrank such scenarios.

`SC200103_214330`: Refer to Fig. 5.20 for a parameter estimation summary of the candidate event. This candidate did not flag any real issues other than the presence of a slightly messy Virgo data around the merger time. This is our most confident candidate with a log Bayes factor of 31.67 and a network matched filter SNR of 10.11 (which is still low, but comfortably above the SNR threshold for science). As expected, the SAGE ranking statistic was also the highest for this candidate comparatively indicating SAGE's capability in distinguishing between candidates of varying Bayes factor. So far, our new candidates have all been very close in vicinity in the chirp mass space, which might hint at a potential mild inadequacy of matched filtering pipelines in capturing candidates in this region. We claim this



Network Matched-Filter SNR	8.93
Network Precessing SNR	1.31
Log Bayes Factor	18.01
FAR (yr^{-1})	<1
p_astro (train)	>0.99
$\mathcal{M}_c^{pred}$	30.953
t_c^{pred}	1242562173.086

Figure 5.15: Parameter estimation summary for SAGE candidate `SC190522_130915`. Corner plot shows posteriors for m_1, m_2, χ_{eff}, $\mathcal{M}_c$ and d_L, alongside sky localisation. Bottom-left: `IMRPhenomXPHM` waveforms from 32 posterior samples (median in dashed purple). PSDs for all detectors and key metrics are also shown.



Network Matched-Filter SNR	9.84
Network Precessing SNR	0.83
Log Bayes Factor	27.89
FAR (yr^{-1})	<1
p_astro (train)	>0.99
$\mathcal{M}_c^{\rm pred}$	31.359
$t_c^{\rm pred}$	1251502439.849

Figure 5.16: Parameter estimation summary for SAGE candidate `SC190902_232241`. Corner plot shows posteriors for m_1, m_2, $\chi_{\rm eff}$, $\mathcal{M}_c$ and d_L, alongside sky localisation. Bottom-left: `IMRPhenomXPHM` waveforms from 32 posterior samples (median in dashed purple). PSDs for all detectors and key metrics are also shown.

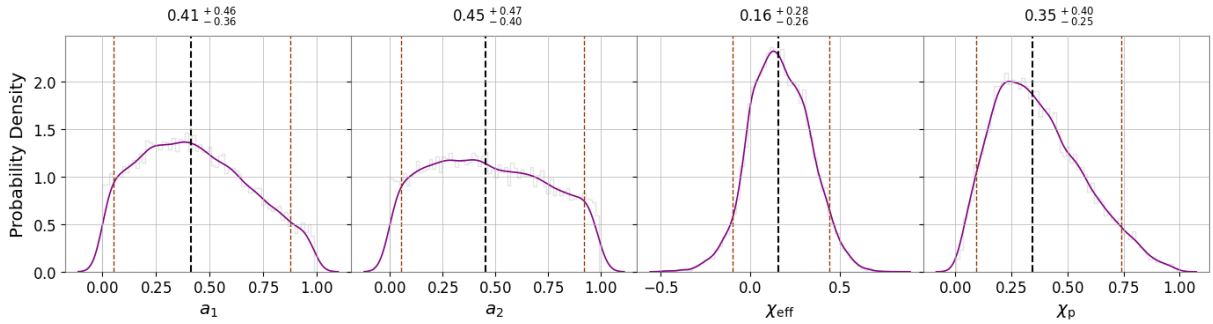


Figure 5.17: 1D histograms on the spin components a_1, a_2, the effective spin $\chi_{\rm eff}$ and the precessing spin $\chi_{\rm p}$.

candidate as our fourth and most confident event: SGW200103_214330!

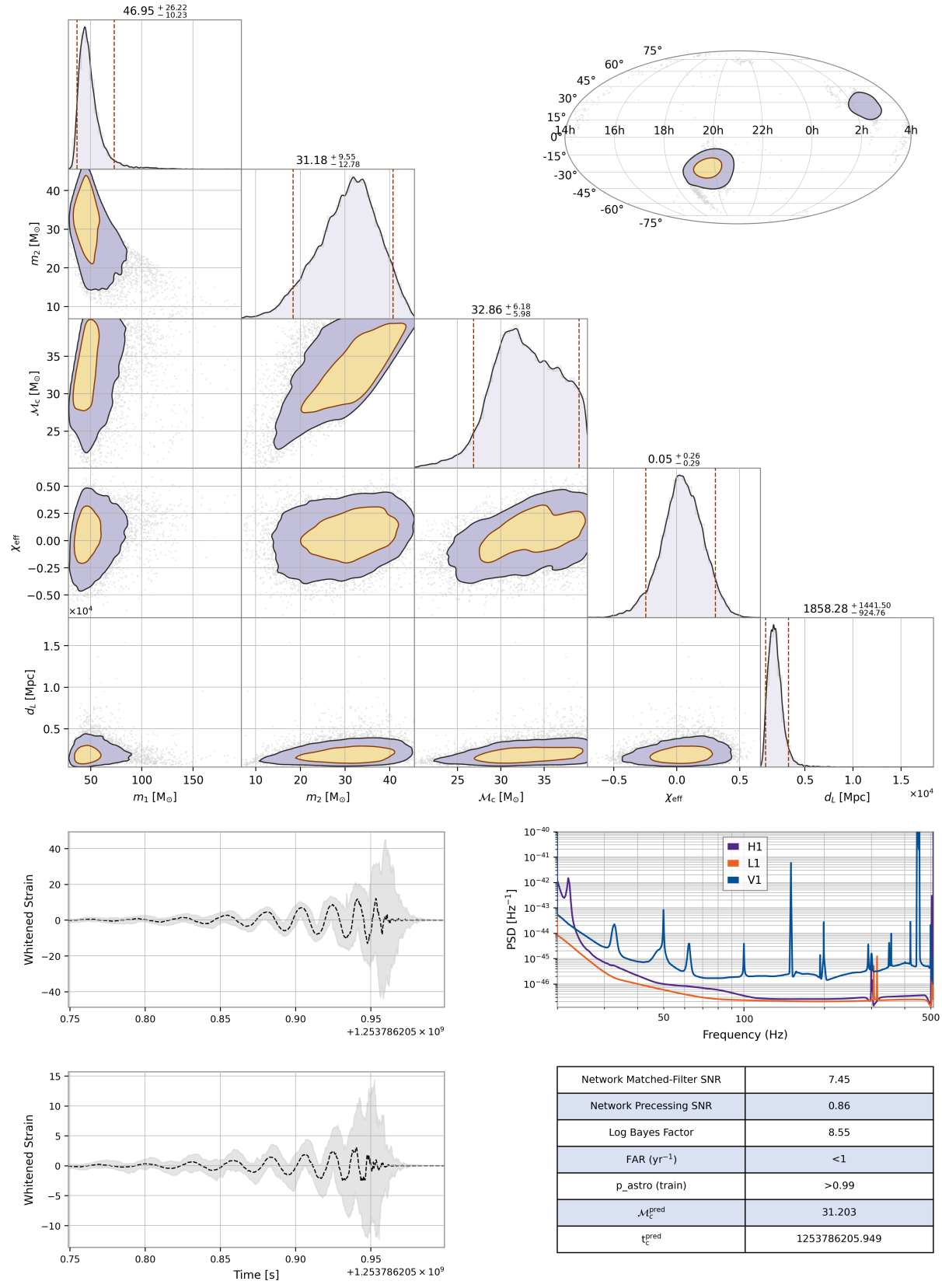
SC200205_085013: Refer to Fig. 5.21 for a parameter estimation summary of the candidate event. This is our last and final candidate with a FAR less than 2 per year. From the summary, we see that this too has a high chance of being a vanilla BBH with standard behaviour. We noticed that the time delay between the H1 and L1 detectors were slightly larger than usual, but this is likely due to the particular noise realisation and the relatively low SNR of the candidate. We claim this final candidate as our fifth and final event: SGW200205_085013!

Event Summary: In total we claim 5 new GW events as a FAR of less than 2 per year. This result showcases the strength of our work and the potential science impact that SAGE will provide to the GW search community. Overall, these five candidates demonstrate the sensitivity and robustness of the SAGE search, highlighting its capability to identify astrophysical BBH events even in challenging noise conditions. Table 5.5.2 shows a summary of all the newly claimed events by SAGE.

Table 5.5: New SAGE GW events in O3. Final curated set of GW events at FAR < 2/year.

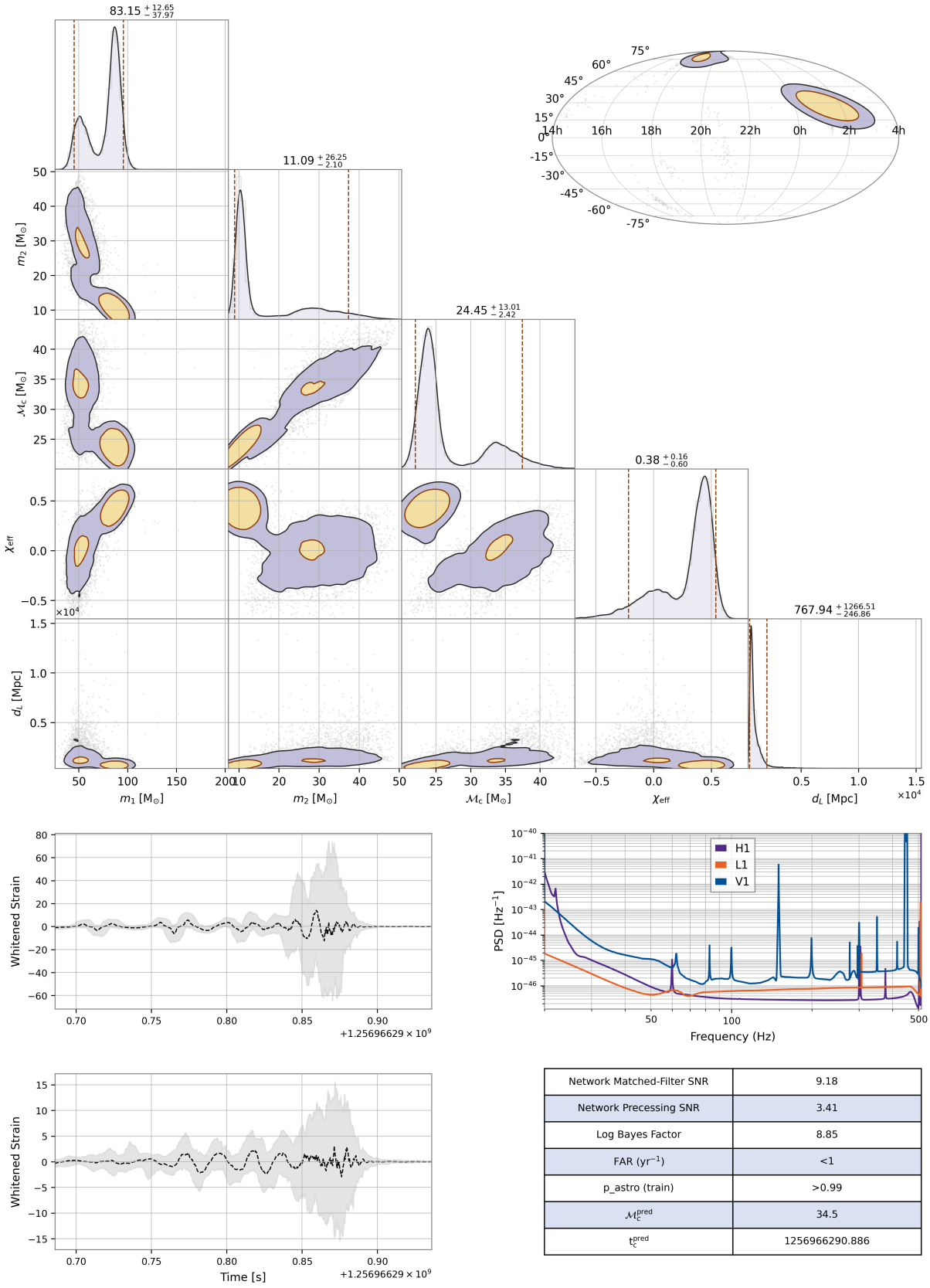
Event	FAR [1/yr]	$p^{\rm train}_{\rm astro}$	$\mathcal{R}_{\rm s}$	$\mathcal{R}_{\rm bg}^{\rm 2/year}$	$\mathcal{M}_c^{\rm pred}$ [$M_\odot$]	2-ifo
SGW200103_214329	< 1	> 0.99	13.078	12.438	35.41	LV
SGW190522_120915	< 1	> 0.99	12.179	11.883	30.95	HV
SGW190902_233341	< 1	> 0.99	12.094	11.883	31.36	HV
SGW190929_095627	< 1	> 0.99	12.164	11.883	31.20	HV
SGW200205_085012	< 1	> 0.99	11.992	11.883	33.19	HV

Interesting Subthreshold Event: So far, we have only looked at triggers with a FAR < 2/year. If we allow for the candidate threshold to be higher, say at 1/month, we get 13 undiscovered candidates in the LV search, 52 in the HV search and 3 in the HL search. Given time and computational constraints, we were not able to run parameter estimation on all of these events. Within the subset of candidates that we were able to analyse (27 candidates analysed excluding all events with FAR < 2/year), we found several vanilla BBH candidates and one candidate among them was interesting in particular. SC190623_192541 had considerable support for positive $\chi_{\rm eff}$, some evidence of precession and



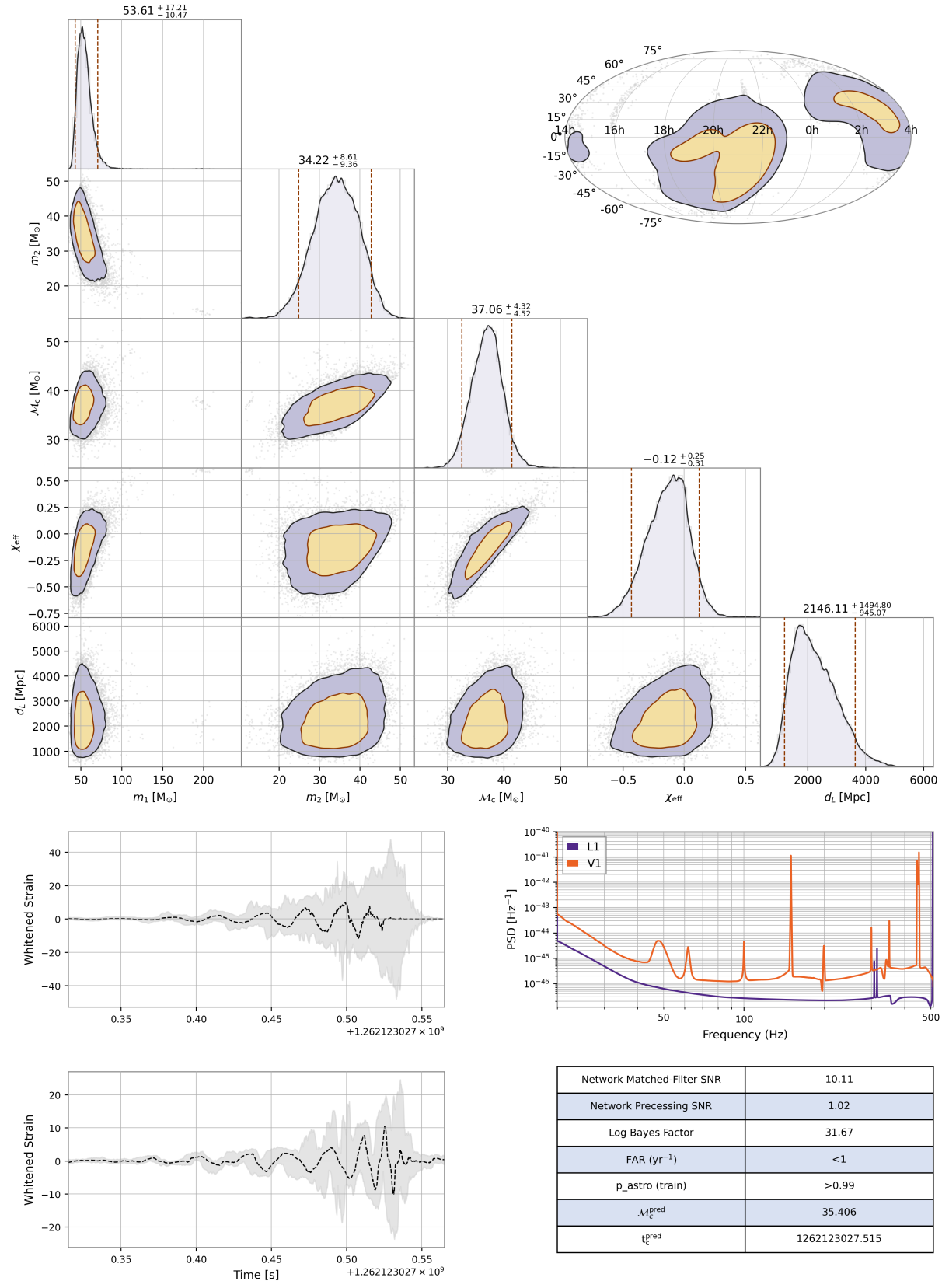
Network Matched-Filter SNR	7.45
Network Precessing SNR	0.86
Log Bayes Factor	8.55
FAR (yr^{-1})	<1
p_astro (train)	>0.99
$\mathcal{M}_c^{pred}$	31.203
t_c^{pred}	1253786205.949

Figure 5.18: Parameter estimation summary for SAGE candidate `SC190929_095627`. Corner plot shows posteriors for m_1, m_2, χ_{eff}, $\mathcal{M}_c$ and d_L, alongside sky localisation. Bottom-left: `IMRPhenomXPHM` waveforms from 32 posterior samples (median in dashed purple). PSDs for all detectors and key metrics are also shown.



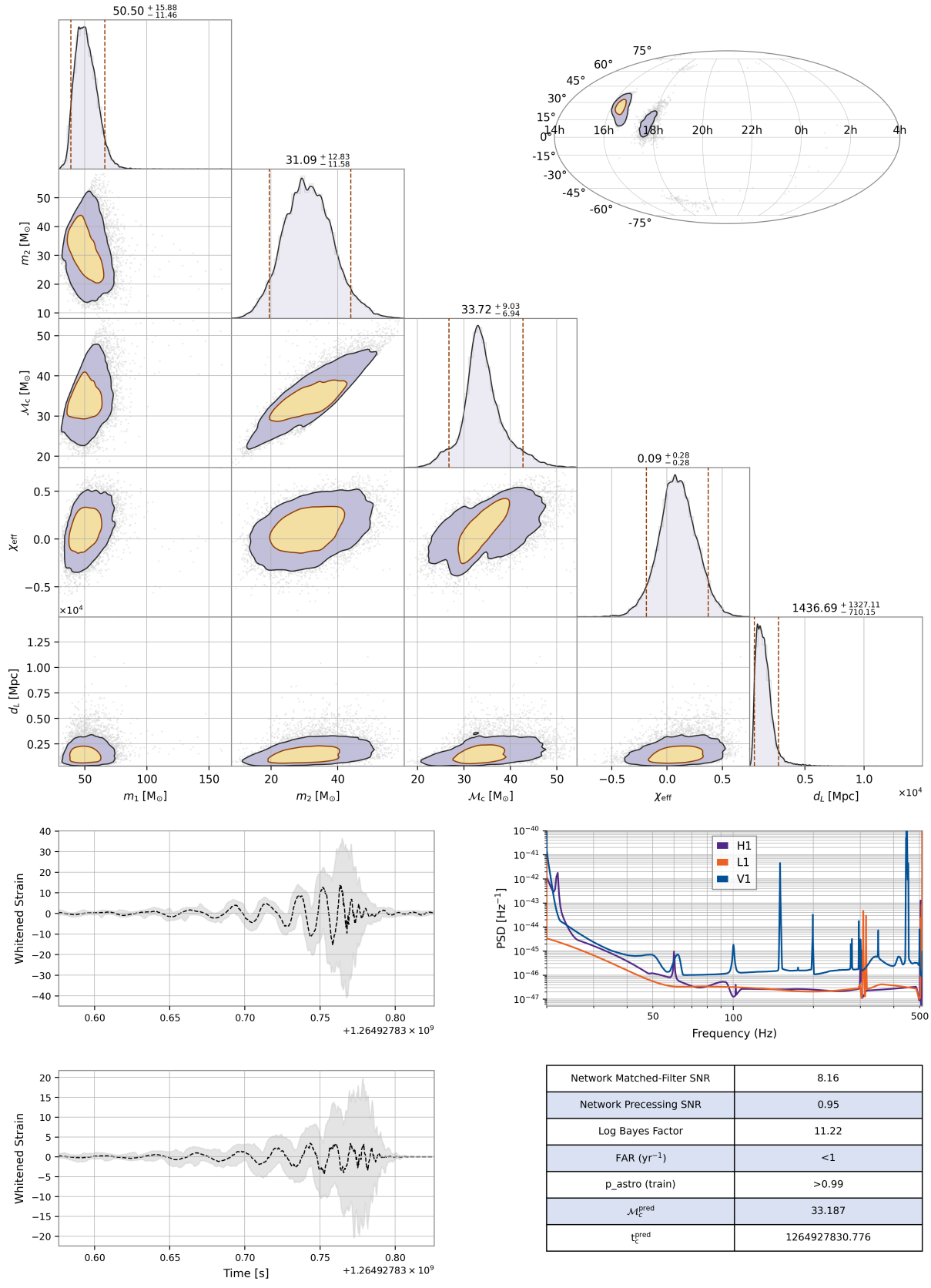
Network Matched-Filter SNR	9.18
Network Precessing SNR	3.41
Log Bayes Factor	8.85
FAR (yr^{-1})	<1
p_astro (train)	>0.99
$\mathcal{M}_c^{pred}$	34.5
t_c^{pred}	1256966290.886

Figure 5.19: Parameter estimation summary for SAGE candidate `SC191105_051753`. Corner plot shows posteriors for m_1, m_2, χ_{eff}, $\mathcal{M}_c$ and d_L, alongside sky localisation. Bottom-left: `IMRPhenomXPHM` waveforms from 32 posterior samples (median in dashed purple). PSDs for all detectors and key metrics are also shown.



Network Matched-Filter SNR	10.11
Network Precessing SNR	1.02
Log Bayes Factor	31.67
FAR (yr^{-1})	<1
p_astro (train)	>0.99
$\mathcal{M}_c^{pred}$	35.406
t_c^{pred}	1262123027.515

Figure 5.20: Parameter estimation summary for SAGE candidate SC200103_214330. Corner plot shows posteriors for m_1, m_2, χ_{eff}, $\mathcal{M}_c$ and d_L, alongside sky localisation. Bottom-left: IMRPhenomXPHM waveforms from 32 posterior samples (median in dashed purple). PSDs for all detectors and key metrics are also shown.



Network Matched-Filter SNR	8.16
Network Precessing SNR	0.95
Log Bayes Factor	11.22
FAR (yr^{-1})	<1
p_astro (train)	>0.99
$\mathcal{M}_c^{pred}$	33.187
t_c^{pred}	1264927830.776

Figure 5.21: Parameter estimation summary for SAGE candidate `SC200205_085013`. Corner plot shows posteriors for m_1, m_2, χ_{eff}, $\mathcal{M}_c$ and d_L, alongside sky localisation. Bottom-left: `IMRPhenomXPHM` waveforms from 32 posterior samples (median in dashed purple). PSDs for all detectors and key metrics are also shown.

strong evidence for asymmetrical masses. We provide a full summary of the analysis in Fig. 5.22. With a log Bayes factor of 9.97, this candidate is similar in significance to the ones in our claimed events list. With well behaved PSDs in both considered detectors and with an SNR comparable to already detected events, there are two possible biases in Sage that might contribute to the downranking of this trigger: [i] candidate has relatively low chirp mass of 16.74 $M_\odot$ and known biases plague this region of the parameter space, and [ii] the spin distribution is biased toward zero and the precessing SNR did not actually contribute to the detection. These reasons provide potential avenues of improvement for Sage.

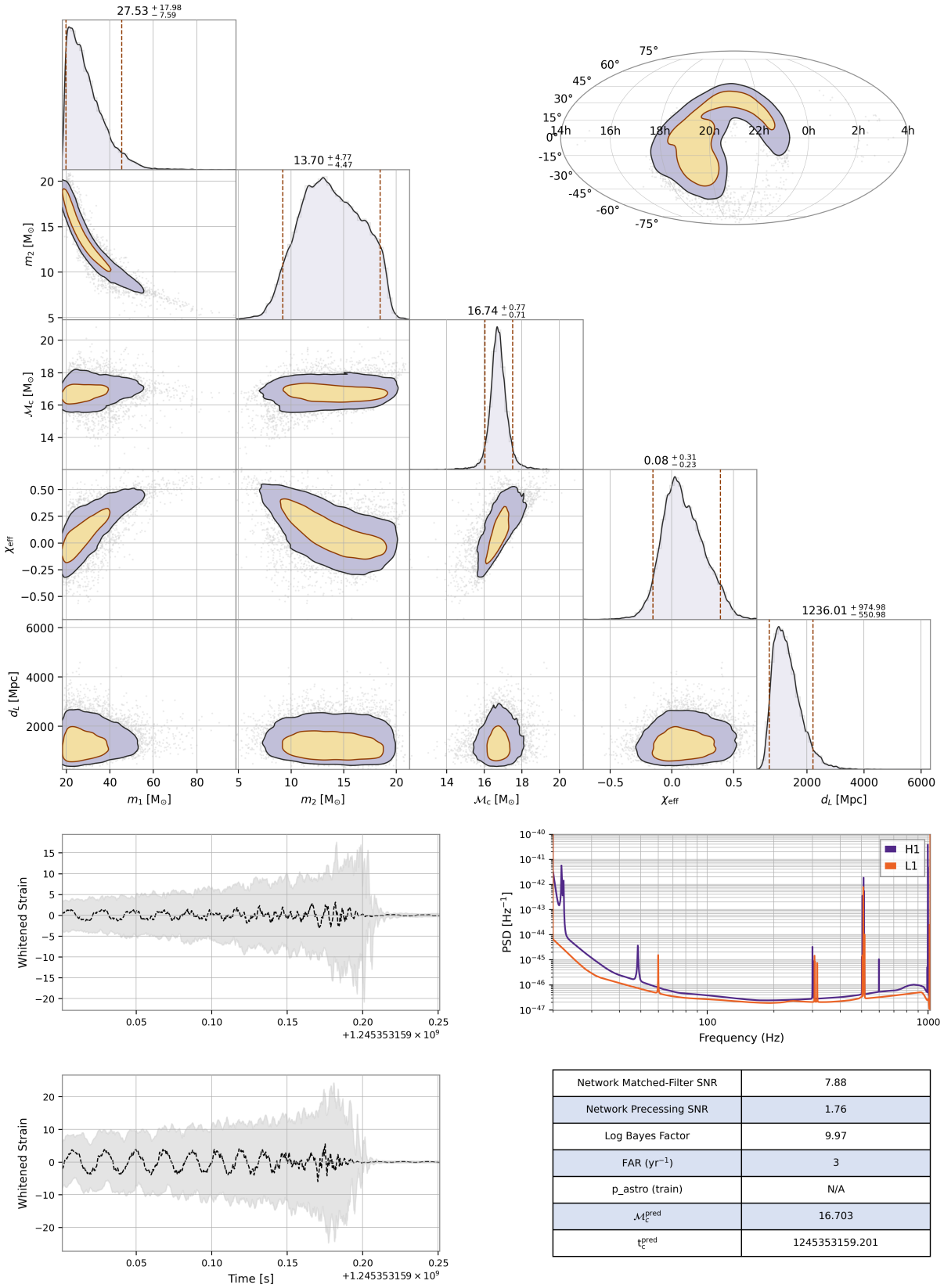
Addressing discrepancy is number of new events: When comparing the GW candidates identified by Sage across different detector pairs, we observe a notable variation in the number of new events relative to the GWTC catalogue at a FAR of 1/month: 3 in HL, 13 in LV, and 52 in HV. This discrepancy raises questions about the statistical plausibility of the HV results. We emphasise that the analysis procedure was identical across all detector pairs, and we have verified the integrity of the pipeline. One possible contributing factor is that traditional matched-filtering searches may exhibit subtle biases with respect to the Virgo detector, potentially underreporting events that are more easily detected in combination with Hanford or Livingston. While this hypothesis remains untested, it highlights the importance of considering detector-specific sensitivity and selection effects when interpreting the apparent excess of HV candidates, and motivates further investigation to assess their astrophysical credibility.

Visualising all new events: In Fig. 5.23 we retrieve publicly available GW strain data from the Gravitational Wave Open Science Center for all newly claimed events at each available detector. For each detector, a short segment of strain data centred on the candidate coalescence time is downloaded. A Q-transform is then computed over a bandpass filter within a [15, 512]Hz region to visualise excess power associated with the candidate signal. For each event, the resulting time–frequency representations from multiple interferometers are plotted, allowing direct visual comparison across detectors. Subplots corresponding to unavailable data are explicitly blanked and labelled. Each panel is annotated with the event time, detector-specific matched-filter SNR, and a classification flag indicating whether the candidate is considered astrophysical. We see characteristic chirps of the claimed events along detectors with high enough SNR.

Visualising rest of the candidates: The Q-transforms for all other candidates that have either been disregarded as noise or marked as potential candidates worth further investigation are provided in Fig. 5.24. These plots are produced in the same way as the Q-transforms of the newly claimed events.

5.5.3 Why did we find new events?

Pipeline Strengths: The strengths of Sage, discussed in previous chapters (chapters 3 and 4), are largely responsible for detecting new GW candidates. Its design incorporates physics-informed representation learning and bias mitigation strategies, allowing the search sensitivity to approach that of



Network Matched-Filter SNR	7.88
Network Precessing SNR	1.76
Log Bayes Factor	9.97
FAR (yr^{-1})	3
p_astro (train)	N/A
$\mathcal{M}_c^{\text{pred}}$	16.703
t_c^{pred}	1245353159.201

Figure 5.22: Parameter estimation summary for SAGE candidate `SC190623_192541`. Corner plot shows posteriors for m_1, m_2, χ_{eff}, $\mathcal{M}_c$ and d_L, alongside sky localisation. Bottom-left: `IMRPhenomXPHM` waveforms from 32 posterior samples (median in dashed purple). PSDs for all detectors and key metrics are also shown.

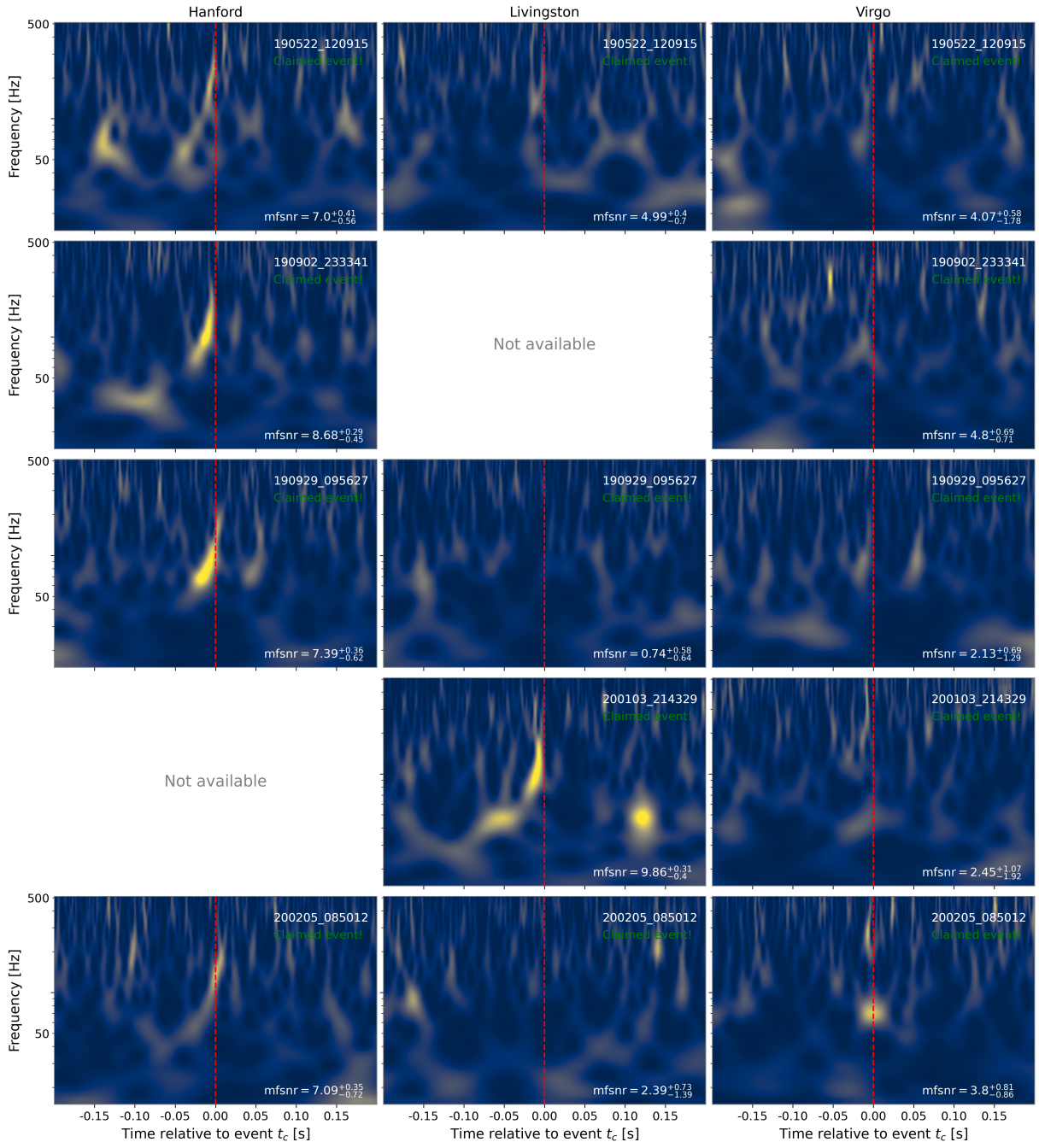


Figure 5.23: Q-transforms of all new events shown in Table 5.5.2 for each available detector. We band-pass the data between 15 Hz and 512 Hz followed by whitening before creating the Q-transform. Please refer to the supplementary code for more details. The subplots provide the event ID and the detector matched filtering SNR for all available detectors.

traditional matched filtering under ideal conditions. In non-ideal cases, such as in the presence of non-Gaussian noise artefacts or poorly-behaved PSDs, SAGE can even outperform matched filtering. Importantly, the network is trained on all new signals found so far, *labelled as noise* in the training set, demonstrating that label noise is effectively handled. This provides insight into why glitches are not memorised despite training on most observed data (with time-shifted samples). Machine learning

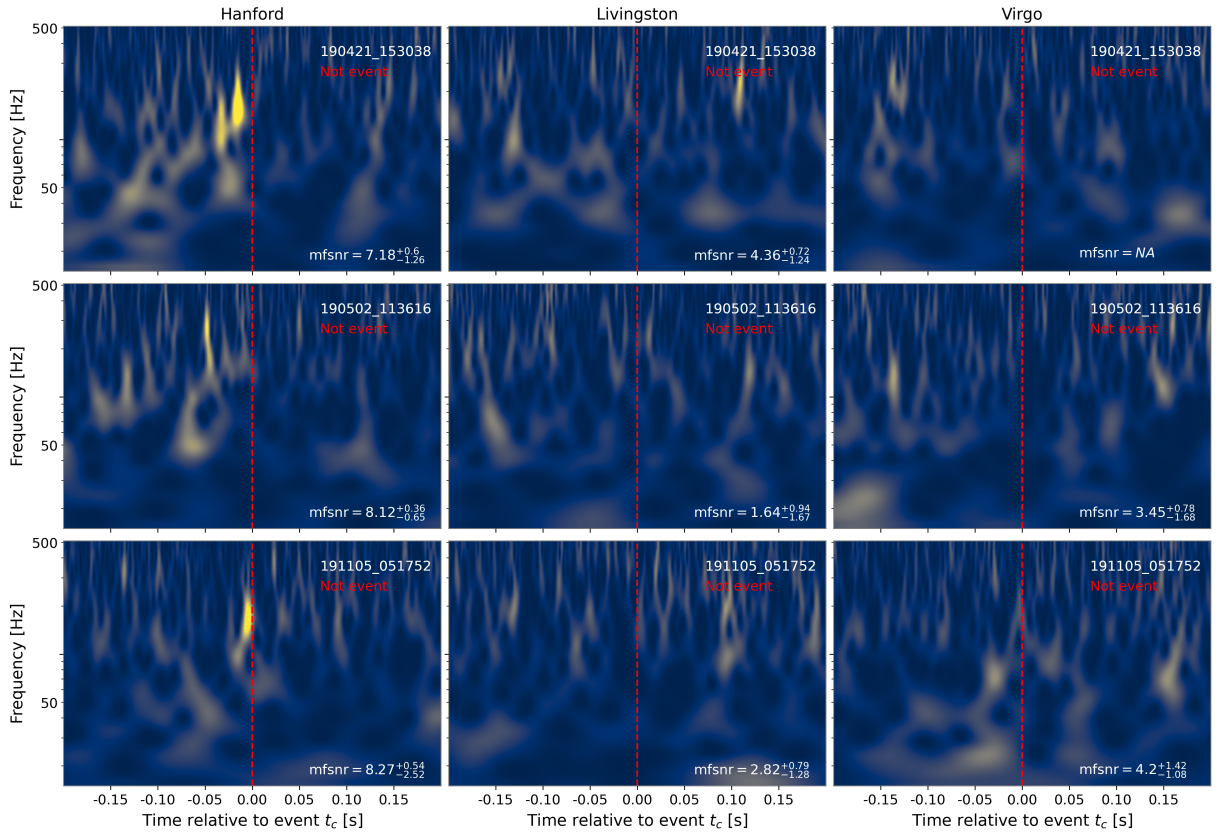


Figure 5.24: Q-transforms of all other candidates shown in Table 5.5.2 for each available detector. We bandpass the data between 15 Hz and 512 Hz followed by whitening before creating the Q-transform. Please refer to the supplementary code for more details. The subplots provide the candidate ID and the detector matched filtering SNR for all available detectors. Detector SNR for V1 is not provided for candidate `SC190421_153038` since parameter estimation, unfortunately, did not include it.

particularly excels in the confusion region, where Type I and Type II errors are most likely, making complex decisions that establish robust decision boundaries. Many of these capabilities are absent in other machine learning pipelines, which explains why SAGE is able to identify signals that other ML pipelines miss.

Noise Robustness: SAGE has been trained on highly varied data and features an architecture specifically designed to learn complex signal characteristics. This includes spatial and channel-wise attention, bespoke large kernel feature extraction, and explicit bias-reducing mechanisms. As a result, the network demonstrates robustness to noise, as evidenced by the results in the previous chapter (see chapter 4), and maintains performance even under challenging noise conditions. Glitches may even be present alongside or on signals, which might be severely downranked by matched filtering vetoes, but SAGE can intelligently make a decision in for or against the input sample in these situations.

Smaller Search Parameter Space: Compared to traditional matched filtering, SAGE operates on a smaller, more constrained search parameter space. This effectively lowers the false alarm threshold for a given FAR, particularly under the assumption of Gaussian noise. In combination with residual biases

that may further constrain the parameter space in complex ways, this can lead to higher sensitivity in specific regions. Notably, the newly found events cluster around similar chirp masses, suggesting that Sage may be more sensitive in this region of parameter space. Further investigation is required to confirm whether this concentration is due to inherent biases or a true astrophysical signal preference.

5.6 Conclusion

In this chapter, we have presented the full end-to-end pipeline for the detection and analysis of GW events using the Sage search. We have detailed the step-by-step methodology implemented to perform a production-level archival search on the O3 data, including the preprocessing of strain data, zero-lag and background estimation, and candidate identification. The search was conducted for the three detector combinations (HL, HV, LV), and the zero-lag triggers were rigorously compared against the estimated background distributions to quantify significance. From this procedure, we identified a total of 52 unique candidate events with a FAR below 2 per year. By performing a detailed catalogue coincidence analysis, we quantified which known GW events were successfully recovered and which were missed. This study revealed that the majority of ID events were robustly detected; however, certain events—particularly those with low chirp mass or extended duration—were missed, consistent with the residual biases identified in the model.

We find 43 events that were reported within GWTC at a FAR < 2/year at a Sage FAR of < 2/year. And notably, we find one event that was reported in GWTC with a FAR of 3.3/year at a Sage FAR of < 1/year, thereby transitioning it beyond the confident detection threshold of 2/year. In addition to recovering known events, Sage identified previously unreported candidate GW events. Nine candidates were flagged with a FAR below 2 per year. We performed a thorough analysis of these candidates. Bayesian parameter estimation was subsequently performed for all candidates, enabling us to characterise their properties. From this analysis, we pruned the candidate list to five high-confidence GW events with a FAR of less than 2 per year, all of which exhibit posteriors consistent with astrophysical BBH mergers. Notably, the newly identified events cluster in a narrow range of chirp masses, reflecting the sensitivity of Sage in this parameter space and highlighting its capability to detect signals in regions where traditional matched filtering pipelines or other ML-based searches may be less effective.

We have also studied the reasons behind both missed detections and newly discovered events. Missed events are primarily associated with low chirp masses, residual biases in the model, or anomalous noise features, including glitches or poorly-behaved PSD segments. Conversely, the discovery of new events is enabled by Sage's architecture, which incorporates physics-informed representation learning, bias mitigation strategies, spatial and channel-wise attention, and large-kernel feature extraction. These features allow the network to learn robust decision boundaries in the presence of complex noise, handle label noise in training, and maintain high sensitivity even in non-ideal or non-Gaussian conditions.

Future work for publication: Future work will focus on further investigation of the factors influencing both newly identified candidate events and missed detections observed in this study. In particular, we will examine signal-processing and statistical effects that may contribute to the observed performance of Sage across different regions of parameter space, as well as the conditions under which sensitivity may be reduced. This includes a systematic assessment of potential residual biases associated with model architecture, training data, and noise characteristics, together with comparative studies of selection effects present in matched filtering pipelines. In addition, we will explore approaches to improving the statistical interpretation of ML-based ranking statistics, including the development of more robust estimates of the astrophysical probability of triggers that account for uncertainty in ML-derived metrics. Further follow-up analyses will also investigate candidate triggers that may be influenced by non-Gaussian noise transients, such as glitch–signal overlap scenarios. These studies aim to strengthen the reliability, interpretability, and statistical robustness of ML-assisted searches in future archival and production-level GW analyses.

Overall, this chapter demonstrates that Sage is a capable and versatile search method, able to recover known GW events, identify previously unreported candidates, and provide insights into the limitations and residual biases inherent in the model. The results presented here highlight the potential of ML-based searches to complement traditional matched filtering, particularly in challenging noise environments, and establish a foundation for future improvements in GW detection pipelines.

6

Contributions & Future Extension

This thesis has investigated the application of machine learning techniques to the detection and characterisation of gravitational waves, with a particular focus on incorporating domain knowledge and mitigating biases inherent in data-driven models. Across the preceding chapters, we have explored both the theoretical foundations of gravitational wave detection and the practical implementation of neural networks for low signal-to-noise ratio detection scenarios. The work has aimed to balance methodological innovation with empirical validation, highlighting the potential of physics-informed representation learning to improve sensitivity, robustness, and interpretability in astrophysical signal analysis.

We first provide an overview of the work presented in the preceding chapters, emphasising the methodological developments and empirical results that underpin our approach. We then detail the key contributions, both in terms of novel algorithms and insights into the behaviour of learning-based detectors. Building on these foundations, we discuss a range of potential extensions and future directions. Finally, we offer a concluding reflection on the broader implications of this work and its place within the ongoing development of gravitational wave astronomy.

6.1 Thesis Summary

Chapter 1: Introduces statistical detection theory, covering the Neyman–Pearson and Bayesian frameworks, and discusses these concepts in the context of gravitational wave detection. It then delves into the fundamentals of waveform morphology and detector properties, and finally presents key open questions motivating the premise of subsequent chapters.

Chapter 2: Introduces matched filtering for gravitational-wave detection, covering data preprocessing, template banks, signal detection, false alarm estimation, and noise handling. Provides a fully working basic implementation of matched filtering with a simple grid template bank on the GW parameter space metric. Concludes with limitations and open questions motivating machine learning approaches.

Chapter 3: Examines machine learning for signal detection, covering network architectures, training,

representational capacity, inductive biases, and applications to gravitational-wave detection. Provides a step-by-step pedagogical review of the detection problem, starting from a single neuron to training a complex neural network on detecting a Bessel function in low-SNR settings. I hope the reader gets at least one *aha!* moment when reading this chapter. I sure had many during the course of my PhD.

Chapter 4: Investigates machine-learning biases in gravitational-wave detection and presents strategies to identify and mitigate them. Using the bespoke pipeline Sage, the chapter analyses sources of bias in training data, network architecture, optimisation, and data processing, and demonstrates mitigation primarily through physics-informed representation learning and the careful use of inductive biases. Empirical results show that Sage outperforms traditional matched filtering and previous machine-learning pipelines in detection sensitivity, handles out-of-distribution noise, and rejects non-Gaussian transients. The chapter highlights the importance of interpretability and bias-aware design in building reliable learning-based detectors.

Chapter 5: Applies the Sage pipeline to O3 data, covering search design, event cataloguing, comparison with existing catalogues, new detections, and parameter estimation, demonstrating the practical impact of physics-informed machine learning on gravitational-wave discovery.

6.2 Key Contributions

We identify and characterise sources of ML learning bias in BBH GW detection and demonstrate mitigation through physics-informed representation learning and carefully chosen inductive biases. Our pipeline, Sage, achieves improved detection performance, outperforming PyCBC (a production-level matched filtering pipeline) and previous ML pipelines in the benchmark MLGWSC-1 mock data challenge at a FAR of 1 per month and discovering 5 new gravitational-wave events in O3 data at a FAR of less than 1 per year. Unlike previous pipelines, Sage does not perform on-the-fly PSD estimation for each samples but rather amortises the process over a large distribution, simplifying preprocessing and improving efficiency (less moving parts). Bias mitigation contributes to effective glitch rejection, while training on real and out-of-distribution PSDs enhances classifier confidence and robustness. Ablation studies further show that inductive biases enable high detection sensitivity with fewer parameters, highlighting the importance of architecture-informed design.

6.3 (near-) Future Extensions

Building on the contributions of this thesis, there exist multiple avenues for extending the methods and applications presented. These future directions aim to enhance detection sensitivity, generalisation across detectors and source types, and the interpretability of machine-learning pipelines. Collectively, these extensions outline a roadmap for advancing physics-informed machine learning for gravitational-wave detection.

6.3.1 Predicting the Odds Ratio

One of the primary concerns posed by sceptics of ML, including myself, is that the ranking statistic from a detection pipeline feels arbitrary, and it does not seem to make physical sense. But this assumption is very far from the truth. The ML ranking statistic that we obtained in chapters 4 and 5 in fact is highly meaningful and appropriate for GW science. Let us try proving this theoretically.

Consider a binary classification problem with labels $y \in \{0, 1\}$ and input $\mathbf{x}$. Let the neural network output a scalar $z(\mathbf{x}) \in \mathbb{R}$, and let the predicted probability of class 1 be

$$\hat{p}(y = 1|\mathbf{x}) = \sigma(z(\mathbf{x})) = \frac{1}{1 + e^{-z(\mathbf{x})}}, \tag{6.1}$$

where $\sigma(\cdot)$ is the sigmoid function. The binary cross-entropy (BCE) loss is

$$\mathcal{L} = -\Big[y \log \hat{p} + (1 - y) \log(1 - \hat{p}) \Big]. \tag{6.2}$$

For a dataset with conditional distribution $p(y|\mathbf{x})$, the expected loss at a given $\mathbf{x}$ is

$$\mathbb{E}_{y|\mathbf{x}}[\mathcal{L}] = -\Big[p \log \hat{p} + (1 - p) \log(1 - \hat{p}) \Big], \tag{6.3}$$

where $p \equiv p(y = 1|\mathbf{x})$ is the true conditional probability. To minimise $\mathcal{L}(\hat{p})$, differentiate with respect to $\hat{p}$ and set to zero:

$$\frac{d\mathcal{L}}{d\hat{p}} = -\frac{p}{\hat{p}} + \frac{1 - p}{1 - \hat{p}} = 0. \tag{6.4}$$

Multiplying through by $\hat{p}(1 - \hat{p})$ gives

$$-p(1 - \hat{p}) + (1 - p)\hat{p} = 0, \tag{6.5}$$

which simplifies to

$$\hat{p} - p = 0 \quad \Longrightarrow \quad \hat{p} = p. \tag{6.6}$$

Thus, the BCE is minimised when the predicted probability equals the true conditional probability:

$$\hat{p} = p(y = 1|\mathbf{x}) \,. \tag{6.7}$$

Since $\hat{p} = \sigma(z)$, we have

$$p = \sigma(z) = \frac{1}{1 + e^{-z}} \,. \tag{6.8}$$

Solving for z gives

$$1 + e^{-z} = \frac{1}{p} \implies e^{-z} = \frac{1 - p}{p} \,, \tag{6.9}$$

$$z = \log \frac{p}{1 - p} \,. \tag{6.10}$$

Minimising the BCE loss forces the network's pre-sigmoid output $z(\mathbf{x})$ to converge to

$$z(\mathbf{x}) = \log \frac{p(y = 1|\mathbf{x})}{p(y = 0|\mathbf{x})} \,, \tag{6.11}$$

which is the ***log odds ratio*** of the true conditional probability. This demonstrates why, in binary classification, the raw neural network output naturally represents the logit of the class probability. We can practically prove this for the gravitational wave detection problem by computing the actual log odds using nested sampling and comparing it with the network ranking statistic. We hope this gives a bit more insight into the ranking statistics seen so far and their scales.

6.3.2 Equivariant Machine Learning Detection

A central principle in machine learning for physical sciences is the incorporation of symmetries into model design. Current neural architectures routinely exploit translation equivariance, for instance through convolutional layers as seen in chapter 3 section 3.11.1. However, the detection of GWs suggests a natural extension: equivariance not only to translations, but also to scalings of the input.

Scale-translation equivariant models extend the standard convolutional framework by considering the composition of translation and scaling groups, in a manner analogous to the group-equivariant networks developed for rotational and reflectional symmetries. By enforcing equivariance to both transformations, we can provide theoretical guarantees for the extrapolation capability of the network. Such guarantees reduce the reliance on extensive training data, lower the number of trainable parameters while increasing representational capacity, and diminish the need for artificial data augmentation.

In the context of GW detection, this extension is particularly compelling. Any observed GW waveform can, in general, be related to another via functional scaling and translation laws that emerge from the underlying physics. While the full transformation space is unbounded in theory, it can be rendered tractable by applying data-informed bounds and physical constraints on astrophysical parame-

ters. This makes scale-translation equivariance not only mathematically elegant but also physically well motivated. We argue that this is the next logical evolution of the SAGE architecture, further aligning its inductive biases with the known structure of the GW detection problem.

6.3.3 Uncertainty Quantification for the Ranking Statistic

Uncertainty quantification is essential for understanding both how well a network is learning and how its predictions may be biased toward certain regions of the GW parameter space. Incorporating uncertainty naturally provides a more Bayesian outlook on the detection problem, moving beyond point estimates toward probabilistic statements about detection confidence.

There exist multiple approaches to obtaining predictive uncertainties, each with distinct trade-offs. A fully Bayesian neural network, where stochasticity is introduced into the weights and biases, represents the most direct solution. However, such models are computationally expensive to train and are often unstable in the low-SNR regime with significant class overlap, as is relevant to GW detection. A more practical alternative is to retain a deterministic backbone while replacing the final layer with a Bayesian last layer. This approach is significantly cheaper, while assuming that the deterministic feature extraction performed by earlier layers is sufficiently expressive and that the last layer can adequately encode predictive uncertainties.

Another strategy is to adopt a post hoc approach. Here, the network is first trained conventionally to produce point estimates. We could then train a conditional normalising flow $q_\phi(R \mid \theta)$ to approximate the distribution of ranking statistics given GW parameters θ. At inference, sampling $R \sim q_\phi(\cdot \mid \theta)$ yields uncertainty estimates on the ranking statistic, reflecting both noise variability and residual model error. This offers a flexible way to capture predictive distributions without requiring a fully Bayesian training procedure.

Despite these advances, a fundamental difficulty remains: disentangling the different sources of uncertainty. Uncertainty may stem from the intrinsic stochasticity of the data (aleatoric uncertainty) or from model-related limitations and biases (epistemic uncertainty). Without a clear separation of these contributions, it is challenging to fully interpret uncertainty estimates. If such disentanglement can be achieved, however, it could simultaneously enable robust Bayesian error bars on network outputs and provide a pathway to quantifying—and ultimately mitigating—model biases.

This line of work would complete the Bayesian perspective initiated in section 6.3.1. Together with the identification of the raw network output as a log-odds ratio, uncertainty quantification would allow the output of the detection pipeline to be interpreted entirely within a Bayesian framework.

6.3.4 On-the-fly Background Estimation

In a coherent analysis, time-slides cannot be performed in the same way as in coincidence-based searches. Rather than mix-matching single-detector ranking statistics, one must shift $h(t)$ directly and rerun the entire search for each slide. This procedure is computationally expensive and infeasible to perform on-the-fly in low-latency search settings, where rapid background estimation is required.

Solution 1: Time-slides are impractical for coherent analyses. An alternative is to precompute backgrounds offline by: (i) estimating representative PSDs from the run, (ii) training a normalising flow to sample new PSDs, (iii) generating coloured Gaussian noise and injecting glitches, and (iv) running the search pipeline to precompute background. This yields a large synthetic background distribution that can be stored compactly and queried in real time, reducing false-alarm uncertainty without expensive time-slides.

Solution 2: If the network directly outputs the Bayesian log odds ratio, time-slide bootstraps become unnecessary. The log odds inherently encode both signal and noise uncertainties, providing a principled, self-contained ranking statistic for detection.

Solution 3: You can do a small number of timeslides, train a generative model (e.g., a normalising flow) to learn the background distribution of ranking statistics, and then sample from it to estimate the tail cheaply. This can be updated online by retraining or fine-tuning the model with new PSDs as they come in, providing a scalable, adaptive background estimator without performing full timeslides.

These solutions are much more scalable than changing the network architecture to store intermediate latent space encodings per detector and mix-matching them on the fly and get a ranking statistic with a relatively simple network.

6.3.5 Learning Long Term Dependencies

In both second- and third-generation (2G and 3G) detectors, long-duration signals—especially from binary neutron stars—pose significant challenges due to the extended time they spend in-band at low frequencies. Processing these signals is computationally expensive, and even with the bias-mitigation strategies developed in this thesis, existing ML architectures struggle to learn effectively over such long sequences. Addressing this requires models capable of capturing long-term dependencies efficiently. Potential solutions include state-space models, recurrent architectures, or other sequence-aware networks that can scale to long-duration signals without prohibitive memory or computational costs. These approaches are essential for applying ML to long signals in both 2G and 3G observatories.

6.3.6 OOD Detection

An important limitation of any supervised ML detector is that it may encounter data that lies outside the distribution of its training set. For gravitational-wave searches, this could include unusual noise

artefacts, novel glitch types, or signals with parameters not well represented in the training data. Without awareness of such OOD inputs, the network's outputs may be unreliable. Implementing OOD detection allows the pipeline to flag inputs that are likely unfamiliar, enabling the results to be treated with caution, discarded, or sent for additional analysis. This ensures more trustworthy detection and reduces the risk of false confidence in spurious outputs.

6.3.7 Sub-Nyquist Waveform Compression

Compressed sensing exploits the fact that gravitational-wave signals are often sparse or compressible in a suitable basis, such as wavelets or chirplets. Instead of sampling at the full Nyquist rate, a small number of linear measurements can be taken, and the full waveform can be reconstructed by solving an optimisation problem that promotes sparsity. This allows one to capture the essential information of the signal with fewer samples, reducing storage and computational costs. By leveraging prior knowledge about the signal structure, compressed sensing provides a principled way to represent and process waveforms efficiently, potentially enabling analysis below the traditional Nyquist-limited sampling.

6.3.8 Big Picture and Research Impact

The schematic in Figure 6.1 summarises the full scope of the research presented in the future extensions, highlighting how the individual contributions integrate into a coherent, high-performance gravitational-wave detection framework. At the core is Sage, a scale- and translation-equivariant architecture capable of efficiently processing long-duration signals while preserving high representational capacity. Its output consists of the Bayesian log odds ratio, complemented by an uncertainty estimate, providing a fully Bayesian probabilistic ranking statistic. Prior to entering Sage, the input strain $h(t)$ is screened by an out-of-distribution detector, ensuring that only familiar, reliable data is analysed; any OOD inputs are flagged and reported. Trusted data are then passed through a physics-informed compression scheme that reduces dimensionality while retaining essential signal structure, enabling efficient processing and mitigating storage and computational costs. Amortised or precomputed background estimation further ensures that the pipeline can operate in low-latency settings without the need for expensive real-time bootstraps. Each numbered element in the schematic corresponds to a specific idea presented in this section, demonstrating how they collectively address key challenges in gravitational-wave detection—from bias mitigation and uncertainty quantification to efficient long-sequence processing—culminating in a unified, robust, and interpretable detection system.

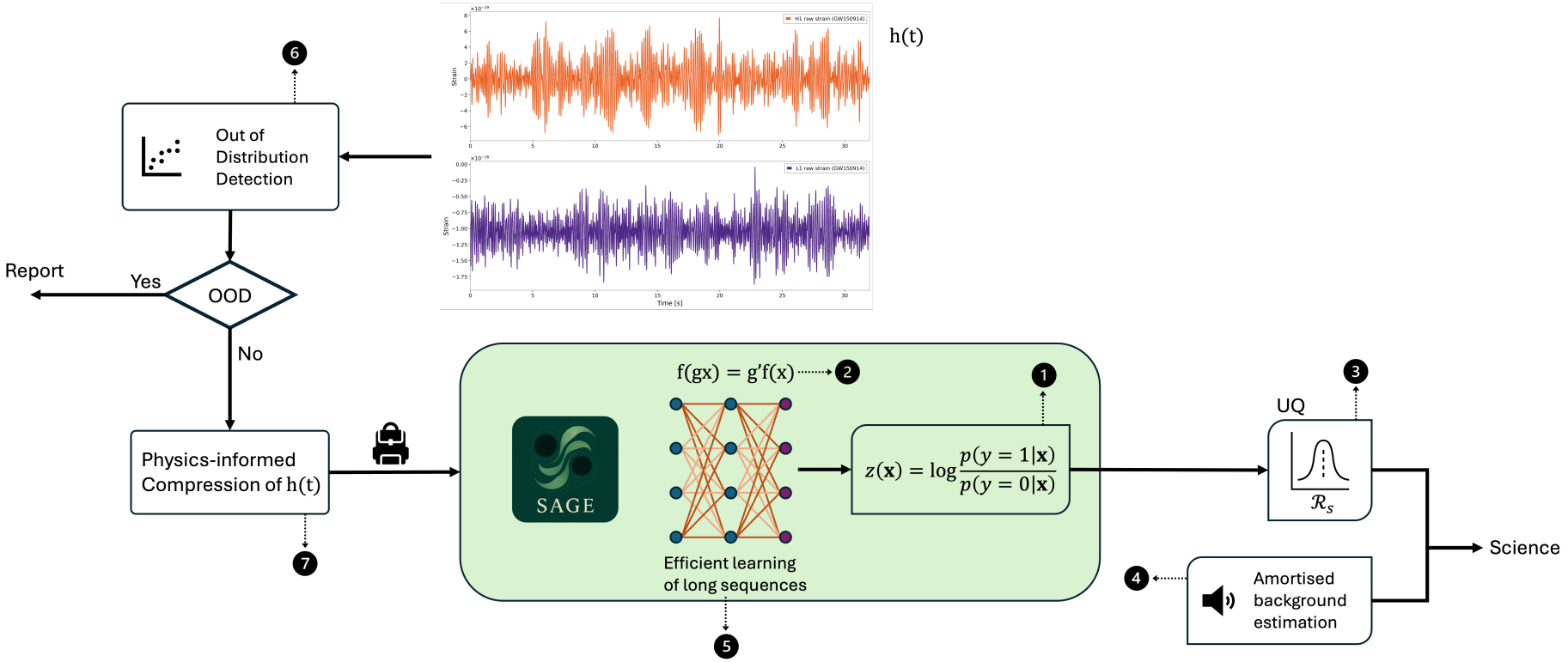


Figure 6.1: Schematic overview of a future potential SAGE pipeline operating in low-latency. The diagram summarises the key components and their interactions, illustrating the end-to-end processing from input strain to final probabilistic output. Numbered labels correspond to individual ideas presented in this thesis.

Research Impact: A future ML-based search pipeline, like the one described, has the potential to a strong trustworthy candidate to deploy alongside matched-filtering for GW science. The output of the network being the Bayes factor with uncertainty quantification (see sections 6.3.1 and 6.3.3) contributes significantly to moving away from a black-box mentality about ML and allows for more informed decision making. It also completely negates the need for bootstrap methods like time-sliding for background estimation. OOD detection (as mentioned in section 6.3.6) furthers trust by actively being aware of data-quality and acknowledging what the search model might be able to handle.

Improving the architecture to be scale-translation equivariant (see section 6.3.2) would reduce the need for large training datasets and reliance on more trainable parameters for better detection performance. This is especially important for the future given that the third-generation ground-based GW detectors [50, 49] alongside the space-based detectors [51, 52] will provide much higher sensitivity, lead to the use of more accurate waveform models and effectively enlarge the search parameter space significantly [120, 302, 88]. They create a search regime where traditional matched-filtering becomes infeasible given the large data volumes. Given that it took us $\approx$ 80 million training samples to get `Sage` to a reasonable sensitivity, training a model like the current version of `Sage` would become infeasible for the future.

Long transient GW that are detectable in the current era and prospective discoveries expected from future detectors, all have prohibitive length scales for non ML and even standard ML techniques to handle efficiently. ML has had long-standing issues when learning long-term dependencies [303]. Significant strides have been made in addressing this issue over the last decades, the most prominent of which include recurrent neural networks [304] and the attention mechanism in transformers [162]. Although transformers are popular for their lower time complexity compared to the previous architectures, they still require a lot of compute when scaled-up to extremely long sequences as the attention mechanism scales quadratically in sequence length.

Previous search pipelines that have dealt with long transient GWs with ML [305, 306, 307] have fallen short due to being computational expensive and exhibiting bias against learning certain regions of the GW parameter space. The third generation of GW detectors will have significantly improved sensitivities, especially in the low frequency regime. A BNS merger will be $\approx$1 day in length given a signal low frequency cut-off of 2 Hz and more than 5 days in length for 1 Hz. For the Einstein telescope, even BBH mergers will be $\approx$5 hours long given a low frequency cut-off of 1 Hz [87].

The improvements mentioned in this chapter mark the next steps forward for **`Sage`**.

6.4 Coda

Ultimately, the pursuit of understanding is what motivates the work presented in this thesis. By confronting complex challenges, integrating diverse methodologies, and pushing the boundaries of both theory and practice, we gain new perspectives on the phenomena under study. While open questions remain and opportunities for further discovery abound, the results presented here highlight the value of rigorous inquiry, careful reasoning, and creative synthesis. To you, the reader, I hope this thesis offers a small glimpse of the enjoyment I found during my PhD, and that you may choose to carry this work forward as one useful puzzle piece in your broader armamentarium of ideas.

Bibliography

[1] The LIGO Scientific Collaboration, the Virgo Collaboration, the KAGRA Collaboration, *et al.*, arXiv e-prints , arXiv:2508.18082 (2025), arXiv:2508.18082 [gr-qc] .

[2] B. P. Abbott *et al.* (LIGO Scientific, Virgo), Phys. Rev. X **9**, 031040 (2019), arXiv:1811.12907 [astro-ph.HE] .

[3] R. Abbott *et al.* (LIGO Scientific, Virgo), Physical Review X **11**, 021053 (2021), arXiv:2010.14527 [gr-qc] .

[4] R. Abbott *et al.*, Phys. Rev. D **109**, 022001 (2024), arXiv:2108.01045 [gr-qc] .

[5] R. Abbott *et al.* (LIGO Scientific Collaboration, Virgo Collaboration, and KAGRA Collaboration), Phys. Rev. X **13**, 041039 (2023).

[6] S. M. Kay, *Fundamentals of statistical processing, Volume 2: Detection theory* (Pearson Education India, 2009).

[7] C. Scott and R. Nowak, IEEE Transactions on Information Theory **51**, 3806 (2005).

[8] C. W. Misner, K. S. Thorne, and J. A. Wheeler, *Gravitation* (W. H. Freeman, San Francisco, 1973).

[9] B. F. Schutz, *A FIRST COURSE IN GENERAL RELATIVITY* (Cambridge Univ. Pr., Cambridge, UK, 1985).

[10] B. P. Abbott *et al.* (LIGO Scientific, Virgo), Phys. Rev. Lett. **116**, 061102 (2016), arXiv:1602.03837 [gr-qc] .

[11] M. Bailes *et al.*, Nature Rev. Phys. **3**, 344 (2021).

[12] A. Abramovici, W. E. Althouse, R. W. P. Drever, Y. Gursel, S. Kawamura, F. J. Raab, D. Shoemaker, L. Sievers, R. E. Spero, K. S. Thorne, R. E. Vogt, R. Weiss, S. E. Whitcomb, and M. E. Zucker, Science **256**, 325 (1992).

[13] T. L. S. Collaboration *et al.*, Classical and Quantum Gravity **32**, 074001 (2015).

[14] A. Buikema *et al.*, Phys. Rev. D **102**, 062003 (2020), arXiv:2008.01301 [astro-ph.IM] .

[15] M. Tse *et al.*, Phys. Rev. Lett. **123**, 231107 (2019).

[16] T. Accadia *et al.* (VIRGO), JINST **7**, P03012.

[17] T. Accadia *et al.*, in *16th Lomonosov Conference on Elementary Particle Physics* (2015) pp. 261–270.

[18] F. Acernese *et al.*, Classical and Quantum Gravity **32**, 024001 (2015), arXiv:1408.3978 [gr-qc] .

[19] F. Acernese *et al.*, Phys. Rev. Lett. **123**, 231108 (2019).

[20] F. Acernese *et al.*, Classical and Quantum Gravity **40**, 185006 (2023), arXiv:2210.15633 [gr-qc] .

[21] Kagra Collaboration *et al.*, Nature Astronomy **3**, 35 (2019), arXiv:1811.08079 [gr-qc] .

[22] T. Akutsu *et al.*, Progress of Theoretical and Experimental Physics **2021**, 05A101 (2020), https://academic.oup.com/ptep/article-pdf/2021/5/05A101/37974994/ptaa125.pdf .

[23] Y. Aso, Y. Michimura, K. Somiya, M. Ando, O. Miyakawa, T. Sekiguchi, D. Tatsumi, and H. Yamamoto, Phys. Rev. D **88**, 043007 (2013), arXiv:1306.6747 [gr-qc] .

[24] K. Somiya, Classical and Quantum Gravity **29**, 124007 (2012), arXiv:1111.7185 [gr-qc] .

[25] D. V. Martynov *et al.*, Phys. Rev. D **93**, 112004 (2016), arXiv:1604.00439 [astro-ph.IM] .

[26] F. Acernese *et al.*, Classical and Quantum Gravity **32**, 024001 (2015), arXiv:1408.3978 [gr-qc] .

[27] T. Akutsu *et al.*, Progress of Theoretical and Experimental Physics **2021**, 05A101 (2021), arXiv:2005.05574 [physics.ins-det] .

[28] R. Abbott *et al.*, Physical Review X **13**, 011048 (2023), arXiv:2111.03634 [astro-ph.HE] .

[29] B. P. Abbott *et al.* (LIGO Scientific, Virgo), Phys. Rev. Lett. **119**, 161101 (2017), arXiv:1710.05832 [gr-qc] .

[30] B. P. Abbott *et al.* (LIGO Scientific, Virgo), Phys. Rev. Lett. **116**, 061102 (2016), arXiv:1602.03837 [gr-qc] .

[31] I. Dvorkin, J.-P. Uzan, E. Vangioni, and J. Silk, Mon. Not. Roy. Astron. Soc. **479**, 121 (2018), arXiv:1709.09197 [astro-ph.HE] .

[32] S. Vitale, W. M. Farr, K. Ng, and C. L. Rodriguez, Astrophys. J. Lett. **886**, L1 (2019), arXiv:1808.00901 [astro-ph.HE] .

[33] S. E. Perkins, *Tests of fundamental physics with gravitational waves: Theory and application*, Ph.D. thesis, Illinois U., Urbana (main), Illinois U., Urbana (2022).

[34] J. Sakstein and B. Jain, Phys. Rev. Lett. **119**, 251303 (2017).

[35] A. Finke, S. Foffa, F. Iacovelli, M. Maggiore, and M. Mancarella, Phys. Rev. D **104**, 084057 (2021), arXiv:2107.05046 [gr-qc] .

[36] B. S. Sathyaprakash and B. F. Schutz, Living Reviews in Relativity **12**, 10.12942/lrr-2009-2 (2009).

[37] B. P. Abbott *et al.*, Nature **551**, 85 (2017), arXiv:1710.05835 [astro-ph.CO] .

[38] S. Mastrogiovanni, C. Karathanasis, J. Gair, G. Ashton, S. Rinaldi, H.-Y. Huang, and G. Dálya, Annalen Phys. **536**, 2200180 (2024).

[39] S. Endlich, V. Gorbenko, J. Huang, and L. Senatore, JHEP **09**, 122, arXiv:1704.01590 [gr-qc] .

[40] R. Takahashi and T. Nakamura, in *International Cosmic Ray Conference*, International Cosmic Ray Conference, Vol. 5 (2003) p. 3153.

[41] J. M. Diego, O. A. Hannuksela, P. L. Kelly, G. Pagano, T. Broadhurst, K. Kim, T. G. F. Li, and G. F. Smoot, A&A **627**, A130 (2019), arXiv:1903.04513 [astro-ph.CO] .

[42] S.-S. Li, S. Mao, Y. Zhao, and Y. Lu, MNRAS **476**, 2220 (2018), arXiv:1802.05089 [astro-ph.CO] .

[43] A. Perego, D. Radice, and S. Bernuzzi, ApJ **850**, L37 (2017), arXiv:1711.03982 [astro-ph.HE] .

[44] S. Valenti, D. J. Sand, S. Yang, E. Cappellaro, L. Tartaglia, A. Corsi, S. W. Jha, D. E. Reichart, J. Haislip, and V. Kouprianov, ApJ **848**, L24 (2017), arXiv:1710.05854 [astro-ph.HE] .

[45] K. D. Alexander, E. Berger, W. Fong, P. K. G. Williams, C. Guidorzi, R. Margutti, B. D. Metzger, J. Annis, P. K. Blanchard, D. Brout, D. A. Brown, H. Y. Chen, R. Chornock, P. S. Cowperthwaite, M. Drout, T. Eftekhari, J. Frieman, D. E. Holz, M. Nicholl, A. Rest, M. Sako, M. Soares-Santos, and V. A. Villar, ApJ **848**, L21 (2017), arXiv:1710.05457 [astro-ph.HE] .

[46] A. C. Searle, P. J. Sutton, and M. Tinto, Classical and Quantum Gravity **26**, 155017 (2009), arXiv:0809.2809 [gr-qc] .

[47] J. D. Romano and N. J. Cornish, Living Reviews in Relativity **20**, 2 (2017), arXiv:1608.06889 [gr-qc] .

[48] T. L. S. Collaboration, the Virgo Collaboration, and the KAGRA Collaboration, Gwtc-4.0: Population properties of merging compact binaries (2025), arXiv:2508.18083 [astro-ph.HE] .

[49] M. Maggiore, C. Van Den Broeck, N. Bartolo, E. Belgacem, D. Bertacca, M. A. Bizouard, M. Branchesi, S. Clesse, S. Foffa, J. García-Bellido, S. Grimm, J. Harms, T. Hinderer, S. Matarrese, C. Palomba, M. Peloso, A. Ricciardone, and M. Sakellariadou, J. Cosmology Astropart. Phys. **2020**, 050 (2020), arXiv:1912.02622 [astro-ph.CO] .

[50] E. D. Hall, Galaxies **10**, 90 (2022).

[51] K. Danzmann and the LISA study team, Classical and Quantum Gravity **13**, A247 (1996).

[52] Z.-C. Liang, Y.-M. Hu, Y. Jiang, J. Cheng, J.-d. Zhang, and J. Mei, Phys. Rev. D **105**, 022001 (2022).

[53] P. Ajith, M. Hannam, S. Husa, Y. Chen, B. Brügmann, N. Dorband, D. Müller, F. Ohme, D. Pollney, C. Reisswig, L. Santamaría, and J. Seiler, Phys. Rev. Lett. **106**, 241101 (2011), arXiv:0909.2867 [gr-qc] .

[54] Y. Pan, A. Buonanno, A. Taracchini, L. E. Kidder, A. H. Mroué, H. P. Pfeiffer, M. A. Scheel, and B. Szilágyi, Phys. Rev. D **89**, 084006 (2014), arXiv:1307.6232 [gr-qc] .

[55] G. Pratten, C. García-Quirós, M. Colleoni, A. Ramos-Buades, H. Estellés, M. Mateu-Lucena, R. Jaume, M. Haney, D. Keitel, J. E. Thompson, and S. Husa, Phys. Rev. D **103**, 104056 (2021), arXiv:2004.06503 [gr-qc] .

[56] A. Ramos-Buades, A. Buonanno, H. Estellés, M. Khalil, D. P. Mihaylov, S. Ossokine, L. Pompili, and M. Shiferaw, Phys. Rev. D **108**, 124037 (2023), arXiv:2303.18046 [gr-qc] .

[57] R. Schnabel (LIGO Scientific), EPJ Web Conf. **57**, 02002 (2013).

[58] L. Santamaría, F. Ohme, P. Ajith, B. Brügmann, N. Dorband, M. Hannam, S. Husa, P. Mösta, D. Pollney, C. Reisswig, E. L. Robinson, J. Seiler, and B. Krishnan, Phys. Rev. D **82**, 064016 (2010).

[59] S. Husa, S. Khan, M. Hannam, M. Pürrer, F. Ohme, X. J. Forteza, and A. Bohé, Phys. Rev. D **93**, 044006 (2016).

[60] S. Khan, S. Husa, M. Hannam, F. Ohme, M. Pürrer, X. J. Forteza, and A. Bohé, Phys. Rev. D **93**, 044007 (2016).

[61] A. D. Whalen, IEEE Communications Magazine **22**, 10.1109/MCOM.1984.1091969 (1984).

[62] J. G. Proakis, *Digital signal processing: principles, algorithms, and applications, 4/E* (Pearson Education India, 2007).

[63] A. V. Oppenheim, *Discrete-time signal processing* (Pearson Education India, 1999).

[64] A. C. Searle, S. M. Scott, and D. E. McClelland, Classical and Quantum Gravity **20**, S721 (2003), arXiv:gr-qc/0308024 [gr-qc] .

[65] P. B. Covas *et al.* (LSC Instrument Authors), Phys. Rev. D **97**, 082002 (2018).

[66] A. H. Nitz, Class. Quant. Grav. **35**, 035016 (2018), arXiv:1709.08974 [gr-qc] .

[67] C. Pankow *et al.*, Phys. Rev. D **98**, 084016 (2018), arXiv:1808.03619 [gr-qc] .

[68] B. P. Abbott *et al.* (LIGO Scientific, Virgo), Class. Quant. Grav. **37**, 055002 (2020), arXiv:1908.11170 [gr-qc] .

[69] R. Abbott *et al.* (LIGO Scientific Collaboration, Virgo Collaboration, and KAGRA Collaboration), Phys. Rev. D **106**, 102008 (2022).

[70] A. G. Abac *et al.* (LIGO Scientific, VIRGO, KAGRA), (2025), arXiv:2511.16863 [gr-qc] .

[71] E. Seo, X. Shan, J. Janquart, O. A. Hannuksela, M. A. Hendry, and B. Hu, Astrophys. J. **988**, 159 (2025), arXiv:2503.02186 [gr-qc] .

[72] D. Sivia and J. Skilling, *Data Analysis: A Bayesian Tutorial*, Oxford science publications (OUP Oxford, 2006).

[73] A. C. Searle, in *12th Gravitational Wave Data Analysis Workshop* (2008) arXiv:0804.1161 [gr-qc] .

[74] L. S. Finn, Phys. Rev. D **46**, 5236 (1992).

[75] C. Capano, *Searching for Gravitational Waves from Compact Binary Coalescence Using LIGO and Virgo Data*, Ph.d. dissertation, Syracuse University (2011), physics - Dissertations.

[76] F. Harris, Proceedings of the IEEE **66**, 51 (1978).

[77] D. Brillinger, *Time Series: Data Analysis and Theory*, Classics in Applied Mathematics (Society for Industrial and Applied Mathematics, 2001).

[78] B. P. Abbott *et al.*, Classical and Quantum Gravity **37**, 055002 (2020), arXiv:1908.11170 [gr-qc] .

[79] C. Talbot, E. Thrane, S. Biscoveanu, and R. Smith, Phys. Rev. Res. **3**, 043049 (2021).

[80] T. Cokelaer, Phys. Rev. D **76**, 102004 (2007), arXiv:0706.4437 [gr-qc] .

[81] B. J. Owen, Phys. Rev. D **53**, 6749 (1996).

[82] B. J. Owen, Phys. Rev. D **53**, 6749 (1996).

[83] S. Babak, R. Balasubramanian, D. Churches, T. Cokelaer, and B. S. Sathyaprakash, Classical and Quantum Gravity **23**, 5477 (2006), arXiv:gr-qc/0604037 [gr-qc] .

[84] B. S. Sathyaprakash and S. V. Dhurandhar, Phys. Rev. D **44**, 3819 (1991).

[85] C. Messenger, R. Prix, and M. A. Papa, Phys. Rev. D **79**, 104017 (2009).

[86] I. W. Harry, B. Allen, and B. S. Sathyaprakash, Phys. Rev. D **80**, 104014 (2009).

[87] M. L. Chan *et al.*, Phys. Rev. D **97**, 123014 (2018).

[88] D. Wadekar, T. Venumadhav, J. Roulet, A. K. Mehta, B. Zackay, J. Mushkin, and M. Zaldarriaga, Phys. Rev. D **110**, 044063 (2024), arXiv:2405.17400 [gr-qc] .

[89] M. Wąs, M.-A. Bizouard, V. Brisson, F. Cavalier, M. Davier, P. Hello, N. Leroy, F. Robinet, and M. Vavoulidis, Classical and Quantum Gravity **27**, 015005 (2010), arXiv:0906.2120 [gr-qc] .

[90] C. Capano, T. Dent, C. Hanna, M. Hendry, C. Messenger, Y.-M. Hu, and J. Veitch, Phys. Rev. D **96**, 082002 (2017).

[91] B. Allen, W. G. Anderson, P. R. Brady, D. A. Brown, and J. D. E. Creighton, Phys. Rev. D **85**, 122006 (2012), arXiv:gr-qc/0509116 .

[92] B. Allen, Phys. Rev. D **71**, 062001 (2005).

[93] S. A. Usman *et al.*, Class. Quant. Grav. **33**, 215004 (2016), arXiv:1508.02357 [gr-qc] .

[94] G. S. Davies, T. Dent, M. Tápai, I. Harry, C. McIsaac, and A. H. Nitz, Phys. Rev. D **102**, 022004 (2020).

[95] M. Cabero *et al.*, Class. Quant. Grav. **36**, 15 (2019), arXiv:1901.05093 [physics.ins-det] .

[96] C. McIsaac and I. Harry, Phys. Rev. D **105**, 104056 (2022).

[97] S. Bini, G. Vedovato, M. Drago, F. Salemi, and G. A. Prodi, Class. Quant. Grav. **40**, 135008 (2023), arXiv:2303.05986 [gr-qc] .

[98] M. Zevin, S. Coughlin, S. Bahaadini, E. Besler, N. Rohani, S. Allen, M. Cabero, K. Crowston, A. K. Katsaggelos, S. L. Larson, T. K. Lee, C. Lintott, T. B. Littenberg, A. Lundgren, C. Østerlund, J. R. Smith, L. Trouille, and V. Kalogera, Classical and Quantum Gravity **34**, 064003 (2017), arXiv:1611.04596 [gr-qc] .

[99] M. Zevin *et al.*, Eur. Phys. J. Plus **139**, 100 (2024), arXiv:2308.15530 [gr-qc] .

[100] K. Cannon, S. Caudill, C. Chan, B. Cousins, J. D. E. Creighton, B. Ewing, H. Fong, P. Godwin, C. Hanna, S. Hooper, R. Huxford, R. Magee, D. Meacher, C. Messick, S. Morisaki, D. Mukherjee, H. Ohta, A. Pace, S. Privitera, I. de Ruiter, S. Sachdev, L. Singer, D. Singh, R. Tapia, L. Tsukada, D. Tsuna, T. Tsutsui, K. Ueno, A. Viets, L. Wade, and M. Wade, SoftwareX **14**, 100680 (2021), arXiv:2010.05082 [astro-ph.IM] .

[101] T. Adams, D. Buskulic, V. Germain, G. M. Guidi, F. Marion, M. Montani, B. Mours, F. Piergiovanni, and G. Wang, Class. Quant. Grav. **33**, 175012 (2016), arXiv:1512.02864 [gr-qc] .

[102] C. Rover, J. Phys. Conf. Ser. **228**, 012008 (2010), arXiv:0911.5051 [gr-qc] .

[103] A. C. Searle, in *12th Gravitational Wave Data Analysis Workshop* (2008) arXiv:0804.1161 [gr-qc] .

[104] P. Kumar and T. Dent, Phys. Rev. D **110**, 043036 (2024), arXiv:2403.10439 [gr-qc] .

[105] T. Dent and J. Veitch, Phys. Rev. D **89**, 062002 (2014), arXiv:1311.7174 [gr-qc] .

[106] J. Yan, M. Avagyan, R. E. Colgan, D. Veske, I. Bartos, J. Wright, Z. Márka, and S. Márka, Phys. Rev. D **105**, 043006 (2022), arXiv:2104.03961 [astro-ph.IM] .

[107] S. Babak, R. Balasubramanian, D. Churches, T. Cokelaer, and B. S. Sathyaprakash, Class. Quant. Grav. **23**, 5477 (2006), arXiv:gr-qc/0604037 .

[108] S. V. Dhurandhar and B. S. Sathyaprakash, Phys. Rev. D **49**, 1707 (1994).

[109] B. J. Owen and B. S. Sathyaprakash, Phys. Rev. D **60**, 022002 (1999), arXiv:gr-qc/9808076 [gr-qc] .

[110] J. Aasi *et al.* (LIGO Scientific), Class. Quant. Grav. **32**, 074001 (2015), arXiv:1411.4547 [gr-qc] .

[111] B. P. Abbott *et al.* (LIGO Scientific, Virgo), Class. Quant. Grav. **33**, 134001 (2016), arXiv:1602.03844 [gr-qc] .

[112] M. Cabero *et al.*, Class. Quant. Grav. **36**, 15 (2019), arXiv:1901.05093 [physics.ins-det] .

[113] S. Biscoveanu, C.-J. Haster, S. Vitale, and J. Davies, Phys. Rev. D **102**, 023008 (2020), arXiv:2004.05149 [astro-ph.HE] .

[114] B. P. Abbott *et al.* (LIGO Scientific, Virgo), Class. Quant. Grav. **35**, 065010 (2018), arXiv:1710.02185 [gr-qc] .

[115] C. Messick *et al.*, Phys. Rev. D **95**, 042001 (2017), arXiv:1604.04324 [astro-ph.IM] .

[116] B. Allen, Phys. Rev. D **71**, 062001 (2005), arXiv:gr-qc/0405045 .

[117] Q. Hu and J. Veitch, Astrophys. J. **945**, 103 (2023), arXiv:2210.04769 [gr-qc] .

[118] S. Wu and A. H. Nitz, Phys. Rev. D **107**, 063022 (2023), arXiv:2209.03135 [astro-ph.IM] .

[119] P. Relton, A. Virtuoso, S. Bini, V. Raymond, I. Harry, M. Drago, C. Lazzaro, A. Miani, and S. Tiwari, Phys. Rev. D **106**, 104045 (2022), arXiv:2208.00261 [gr-qc] .

[120] J. R. Gair, L. Barack, T. Creighton, C. Cutler, S. L. Larson, E. S. Phinney, and M. Vallisneri, Class. Quant. Grav. **21**, S1595 (2004), arXiv:gr-qc/0405137 .

[121] H. Ye, G. Y. Li, and B.-H. Juang, IEEE Wireless Communications Letters **7**, 114 (2018).

[122] J. Chen, Y. Wang, and D. Wang, IEEE/ACM Transactions on Audio, Speech, and Language Processing **22**, 1993 (2014).

[123] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, Trans. Mach. Learn. Res. **2022** (2022).

[124] K. Kawaguchi, L. Pack Kaelbling, and Y. Bengio, arXiv e-prints , arXiv:1710.05468 (2017), arXiv:1710.05468 [stat.ML] .

[125] G. V. Pérez, C. Q. Camargo, and A. A. Louis, ArXiv **abs/1805.08522** (2018).

[126] M.-P. Jarabo-Amores, M. Rosa-Zurera, R. Gil-Pita, and F. Lopez-Ferreras, IEEE Transactions on Signal Processing **57**, 4175 (2009).

[127] M.-P. Jarabo-Amores, D. d. la Mata-Moya, R. Gil-Pita, and M. Rosa-Zurera, EURASIP Journal on Applied Signal Processing **2013**, 10.1186/1687-6180-2013-44 (2013).

[128] Y. Bengio, A. Courville, and P. Vincent, IEEE Transactions on Pattern Analysis and Machine Intelligence **35**, 1798 (2013).

[129] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, Vol. 1 (MIT press Cambridge, 2016).

[130] G. Vardi, G. Yehudai, and O. Shamir, arXiv e-prints , arXiv:2106.01101 (2021), arXiv:2106.01101 [cs.LG] .

[131] I. B. Levitan and L. K. Kaczmarek, *The neuron: cell and molecular biology* (Oxford University Press, USA, 2002).

[132] M. T. Augustine, A survey on universal approximation theorems (2024), arXiv:2407.12895 [cs.LG] .

[133] D. George and E. A. Huerta, Physics Letters B **778**, 64 (2018), arXiv:1711.03121 [gr-qc] .

[134] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, Phys. Rev. Lett. **120**, 141103 (2018).

[135] D. Maulud and A. M. Abdulazeez, Journal of applied science and technology trends **1**, 140 (2020).

[136] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. d. F. Costa, and F. A. Rodrigues, PloS one **14**, e0210236 (2019).

[137] G. T. Reddy, M. P. K. Reddy, K. Lakshmanna, R. Kaluri, D. S. Rajput, G. Srivastava, and T. Baker, Ieee Access **8**, 54776 (2020).

[138] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, IEEE transactions on pattern analysis and machine intelligence **44**, 7327 (2021).

[139] L. Ciampiconi, A. Elwood, M. Leonardi, A. Mohamed, and A. Rozza, ArXiv **abs/2301.05579** (2023).

[140] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, IEEE Transactions on Pattern Analysis and Machine Intelligence **42**, 318 (2020).

[141] S. Ruder, (2016), arXiv:1609.04747 [cs.LG] .

[142] D. P. Kingma and J. Ba, CoRR **abs/1412.6980** (2014).

[143] I. Loshchilov and F. Hutter, arXiv preprint arXiv:1711.05101 (2017).

[144] S. Liu, D. Papailiopoulos, and D. Achlioptas, Advances in Neural Information Processing Systems **33** (2020).

[145] P. Zhou, J. Feng, C. Ma, C. Xiong, S. Hoi, and E. Weinan, in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20 (Curran Associates Inc., Red Hook, NY, USA, 2020).

[146] M. A. Nielsen, Neural networks and deep learning (2015).

[147] L. Ericsson, H. Gouk, C. C. Loy, and T. M. Hospedales, IEEE Signal Processing Magazine **39**, 42 (2022).

[148] S. S. Du and W. Hu, arXiv e-prints , arXiv:1901.08572 (2019), arXiv:1901.08572 [cs.LG] .

[149] R. Eldan and O. Shamir, arXiv e-prints , arXiv:1512.03965 (2015), arXiv:1512.03965 [cs.LG] .

[150] S. Sun, W. Chen, L. Wang, X. Liu, and T.-Y. Liu, arXiv e-prints , arXiv:1506.05232 (2015), arXiv:1506.05232 [cs.LG] .

[151] S. Elfwing, E. Uchibe, and K. Doya, arXiv e-prints , arXiv:1702.03118 (2017), arXiv:1702.03118 [cs.LG] .

[152] V. Feldman and C. Zhang, in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20 (Curran Associates Inc., Red Hook, NY, USA, 2020).

[153] D. Arpit, S. Jastrzundefinedbski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17 (JMLR.org, 2017) p. 233–242.

[154] S. Chatterjee, in *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 80, edited by J. Dy and A. Krause (PMLR, 2018) pp. 755–763.

[155] X. Glorot and Y. Bengio, in *International Conference on Artificial Intelligence and Statistics* (2010).

[156] L. Lu, Y. Shin, Y. Su, and G. Karniadakis, Communications in Computational Physics **28**, 1671 (2020).

[157] T. M. Mitchell, (1980).

[158] D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro, J. Mach. Learn. Res. **19**, 2822–2878 (2018).

[159] B. Elesedy and S. Zaidi, arXiv e-prints , arXiv:2102.10333 (2021), arXiv:2102.10333 [stat.ML] .

[160] T. S. Cohen *et al.*, Group equivariant convolutional networks (2016), arXiv:1602.07576 [cs.LG] .

[161] Z. Niu, G. Zhong, and H. Yu, Neurocomputing **452**, 48 (2021).

[162] A. Vaswani *et al.*, in *Neural Information Processing Systems* (2017).

[163] J.-B. Cordonnier, A. Loukas, and M. Jaggi, arXiv preprint arXiv:1911.03584 (2019).

[164] C. Yun, S. Bhojanapalli, A. S. Rawat, S. J. Reddi, and S. Kumar, arXiv preprint arXiv:1912.10077 (2019).

[165] C. Mingard, J. Skalse, G. V. Pérez, D. Martínez-Rubio, V. Mikulik, and A. A. Louis, ArXiv **abs/1909.11522** (2019).

[166] G. V. Pérez, C. Q. Camargo, and A. A. Louis, ArXiv **abs/1805.08522** (2018).

[167] D. George and E. A. Huerta, in *NiPS Summer School 2017* (2017) arXiv:1711.07966 [gr-qc] .

[168] D. George and E. A. Huerta, Phys. Rev. D **97**, 044039 (2018).

[169] C. Dreissigacker, R. Sharma, C. Messenger, R. Zhao, and R. Prix, Phys. Rev. D **100**, 044009 (2019).

[170] A. L. Miller, P. Astone, S. D'Antonio, S. Frasca, G. Intini, I. La Rosa, P. Leaci, S. Mastrogiovanni, F. Muciaccia, A. Mitidis, C. Palomba, O. J. Piccinni, A. Singhal, B. F. Whiting, and L. Rei, Phys. Rev. D **100**, 062005 (2019).

[171] T. D. Gebhard, N. Kilbertus, I. Harry, and B. Schölkopf, Phys. Rev. D **100**, 063015 (2019).

[172] M. L. Chan, I. S. Heng, and C. Messenger, Phys. Rev. D **102**, 043022 (2020).

[173] J. Bayley, C. Messenger, and G. Woan, Phys. Rev. D **102**, 083024 (2020).

[174] M. B. Schäfer, F. Ohme, and A. H. Nitz, Phys. Rev. D **102**, 063015 (2020).

[175] C. Dreissigacker and R. Prix, Phys. Rev. D **102**, 022005 (2020).

[176] C. Chatterjee, L. Wen, F. Diakogiannis, and K. Vinsen, Phys. Rev. D **104**, 064046 (2021).

[177] T. Marianer, D. Poznanski, and J. X. Prochaska, MNRAS **500**, 5408 (2021), arXiv:2010.11949 [astro-ph.IM] .

[178] S. Goyal, H. D., S. J. Kapadia, and P. Ajith, Phys. Rev. D **104**, 124057 (2021).

[179] F. Morawski, M. Bejger, E. Cuoco, and L. Petre, Mach. Learn. Sci. Tech. **2**, 045014 (2021), arXiv:2103.07688 [astro-ph.IM] .

[180] C. Messenger, C. Zerafa, E. Cuoco, M. J. Williams, and W. Reade, G2net gravitational wave detection (2021).

[181] M. B. Schäfer, O. c. v. Zelenka, A. H. Nitz, F. Ohme, and B. Brügmann, Phys. Rev. D **105**, 043002 (2022).

[182] M. B. Schäfer and A. H. Nitz, Phys. Rev. D **105**, 043003 (2022).

[183] X.-T. Zhang, C. Messenger, N. Korsakova, M. L. Chan, Y.-M. Hu, and J.-d. Zhang, Phys. Rev. D **105**, 123027 (2022).

[184] M. B. Schäfer, O. c. v. Zelenka, A. H. Nitz, H. Wang, S. Wu, Z.-K. Guo, Z. Cao, Z. Ren, P. Nousi, N. Stergioulas, P. Iosif, A. E. Koloniari, A. Tefas, N. Passalis, F. Salemi, G. Vedovato, S. Klimenko, T. Mishra, B. Brügmann, E. Cuoco, E. A. Huerta, C. Messenger, and F. Ohme, Phys. Rev. D **107**, 023021 (2023).

[185] P. Nousi, A. E. Koloniari, N. Passalis, P. Iosif, N. Stergioulas, and A. Tefas, Phys. Rev. D **108**, 024022 (2023).

[186] O. Zelenka, B. Brügmann, and F. Ohme, arXiv e-prints , arXiv:2402.07492 (2024), arXiv:2402.07492 [astro-ph.IM] .

[187] V. Skliris, M. R. K. Norman, and P. J. Sutton, Phys. Rev. D **110**, 104034 (2024).

[188] E. Marx, W. Benoit, A. Gunny, R. Omer, D. Chatterjee, R. C. Venterea, L. Wills, M. Saleem, E. Moreno, R. Raikman, E. Govorkova, M. Desai, J. Krupa, D. Rankin, M. W. Coughlin, P. Harris, and E. Katsavounidis, Phys. Rev. D **111**, 042010 (2025).

[189] N. Nagarajan and C. Messenger, Identifying and mitigating machine learning biases for the gravitational-wave detection problem (2025), arXiv:2501.13846 [gr-qc] .

[190] R. Abbott, others, V. C. Ligo Scientific Collaboration, and Kagra Collaboration, Physical Review X **13**, 041039 (2023), arXiv:2111.03606 [gr-qc] .

[191] F. Chollet, ArXiv **abs/1911.01547** (2019).

[192] Z. Fang, Y. Li, J. Lu, J. Dong, B. Han, and F. Liu, ArXiv **abs/2210.14707** (2022).

[193] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, ACM Comput. Surv. **54**, 10.1145/3457607 (2021).

[194] I. H. Sarker, Sn Computer Science **2** (2021).

[195] S. Babak, R. Biswas, P. R. Brady, D. A. Brown, K. Cannon, C. D. Capano, J. H. Clayton, T. Cokelaer, J. D. E. Creighton, T. Dent, A. Dietz, S. Fairhurst, N. Fotopoulos, G. González, C. Hanna, I. W. Harry, G. Jones, D. Keppel, D. J. A. McKechan, L. Pekowsky, S. Privitera, C. Robinson, A. C. Rodriguez, B. S. Sathyaprakash, A. S. Sengupta, M. Vallisneri, R. Vaulin, and A. J. Weinstein, Phys. Rev. D **87**, 024033 (2013), arXiv:1208.3491 [gr-qc] .

[196] B. P. Abbott *et al.*, Living Reviews in Relativity **21**, 3 (2018), arXiv:1304.0670 [gr-qc] .

[197] T. L. S. Collaboration *et al.*, Classical and Quantum Gravity **32**, 074001 (2015).

[198] A. Buikema *et al.*, Phys. Rev. D **102**, 062003 (2020).

[199] M. Tse *et al.*, Phys. Rev. Lett. **123**, 231107 (2019).

[200] D. V. Martynov *et al.*, Phys. Rev. D **93**, 112004 (2016).

[201] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Phys. Rev. Lett. **116**, 131103 (2016).

[202] G. M. Harry and (forthe LIGO Scientific Collaboration), Classical and Quantum Gravity **27**, 084006 (2010).

[203] F. Acernese *et al.* (Virgo Collaboration), Phys. Rev. Lett. **123**, 231108 (2019).

[204] M. B. Schäfer, O. Zelenka, A. H. Nitz, H. Wang, S. Wu, Z.-K. Guo, Z. Cao, Z. Ren, P. Nousi, N. Stergioulas, P. Iosif, A. E. Koloniari, A. Tefas, N. Passalis, F. Salemi, G. Vedovato, S. Klimenko, T. Mishra, B. Brügmann, E. Cuoco, E. A. Huerta, C. Messenger, and F. Ohme, Phys. Rev. D **107**, 023021 (2023), arXiv:2209.11146 [astro-ph.IM] .

[205] A. H. Nitz, T. Dent, T. D. Canton, S. Fairhurst, and D. A. Brown, The Astrophysical Journal **849**, 118 (2017).

[206] A. H. Nitz, T. Dal Canton, D. Davis, and S. Reyes, Phys. Rev. D **98**, 024050 (2018), arXiv:1805.11174 [gr-qc] .

[207] R. Balestriero, A. I. Humayun, and R. Baraniuk, On the geometry of deep learning (2024), arXiv:2408.04809 [cs.LG] .

[208] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, Phys. Rev. Lett. **120**, 141103 (2018), arXiv:1712.06041 [astro-ph.IM] .

[209] P. Nousi, A. E. Koloniari, N. Passalis, P. Iosif, N. Stergioulas, and A. Tefas, Phys. Rev. D **108**, 024022 (2023), arXiv:2211.01520 [gr-qc] .

[210] A. E. Koloniari, E. C. Koursoumpa, P. Nousi, P. Lampropoulos, N. Passalis, A. Tefas, and N. Stergioulas, Mach. Learn. Sci. Tech. **6**, 015054 (2025), arXiv:2407.07820 [gr-qc] .

[211] M. B. Schäfer and A. H. Nitz, Phys. Rev. D **105**, 043003 (2022), arXiv:2108.10715 [astro-ph.IM] .

[212] E. Marx, W. Benoit, A. Gunny, R. Omer, D. Chatterjee, R. C. Venterea, L. Wills, M. Saleem, E. Moreno, R. Raikman, E. Govorkova, D. Rankin, M. W. Coughlin, P. Harris, and E. Katsavounidis, A machine-learning pipeline for real-time detection of gravitational waves from compact binary coalescences (2024), arXiv:2403.18661 [gr-qc] .

[213] T. D. Gebhard, N. Kilbertus, I. Harry, and B. Schölkopf, Phys. Rev. D **100**, 063015 (2019), arXiv:1904.08693 [astro-ph.IM] .

[214] C. Cutler and É. E. Flanagan, Phys. Rev. D **49**, 2658 (1994), arXiv:gr-qc/9402014 [gr-qc] .

[215] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh, ArXiv **abs/1810.00825** (2018).

[216] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari, in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13 (Curran Associates Inc., Red Hook, NY, USA, 2013) p. 1196–1204.

[217] M. Denil and T. Trappenberg, in *Proceedings of the 23rd Canadian Conference on Advances in Artificial Intelligence*, AI'10 (Springer-Verlag, Berlin, Heidelberg, 2010) p. 220–231.

[218] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, A survey of large language models (2025), arXiv:2303.18223 [cs.CL] .

[219] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, Vol. 4 (Springer, 2006).

[220] M. S. Masiha, A. Gohari, M. H. Yassaee, and M. R. Aref, in *2021 IEEE International Symposium on Information Theory (ISIT)* (2021) pp. 2912–2917.

[221] R. J. N. Baldock, H. Maennel, and B. Neyshabur, arXiv e-prints , arXiv:2106.09647 (2021), arXiv:2106.09647 [cs.LG] .

[222] Y. Bengio, P. Simard, and P. Frasconi, IEEE Transactions on Neural Networks **5**, 157 (1994).

[223] S. E. Hihi and Y. Bengio, in *Neural Information Processing Systems* (1995).

[224] S. Hochreiter and J. Schmidhuber, Neural computation **9**, 1735 (1997).

[225] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, in *Neural Information Processing Systems* (2017).

[226] W. Luo, Y. Li, R. Urtasun, and R. S. Zemel, ArXiv **abs/1701.04128** (2016).

[227] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, in *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, 2019) pp. 5301–5310.

[228] R. Basri, M. Galun, A. Geifman, D. W. Jacobs, Y. Kasten, and S. Kritchman, in *International Conference on Machine Learning* (2020).

[229] G. Pratten, C. García-Quirós, M. Colleoni, A. Ramos-Buades, H. Estellés, M. Mateu-Lucena, R. Jaume, M. Haney, D. Keitel, J. E. Thompson, and S. Husa, Phys. Rev. D **103**, 104056 (2021).

[230] C. Mills and S. Fairhurst, Phys. Rev. D **103**, 024042 (2021), arXiv:2007.04313 [gr-qc] .

[231] R. Green, C. Hoy, S. Fairhurst, M. Hannam, F. Pannarale, and C. Thomas, Phys. Rev. D **103**, 124023 (2021), arXiv:2010.04131 [gr-qc] .

[232] J. Solomon, O. M., PSD computations using Welch's method (1991).

[233] J. Glanzer *et al.*, Gravity spy machine learning classifications of ligo glitches from observing runs 01, 02, 03a, and 03b (2021).

[234] B. S. Sathyaprakash and S. V. Dhurandhar, Phys. Rev. D **44**, 3819 (1991).

[235] J. P. Cohen, T. Cao, J. D. Viviano, C.-W. Huang, M. Fralick, M. Ghassemi, M. Mamdani, R. Greiner, and Y. Bengio, Cmaj **193**, E1391 (2021).

[236] E. Beede, E. Baylor, F. Hersch, A. Iurchenko, L. Wilcox, P. Ruamviboonsuk, and L. M. Vardoulakis, in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20 (Association for Computing Machinery, New York, NY, USA, 2020) p. 1–12.

[237] M. B. Schäfer, F. Ohme, and A. H. Nitz, Phys. Rev. D **102**, 063015 (2020), arXiv:2006.01509 [astro-ph.HE] .

[238] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, 2014 IEEE Conference on Computer Vision and Pattern Recognition , 580 (2013).

[239] K. He, X. Zhang, S. Ren, and J. Sun, IEEE Transactions on Pattern Analysis and Machine Intelligence **37**, 1904 (2015).

[240] K. He, G. Gkioxari, P. Dollár, and R. Girshick, in *2017 IEEE International Conference on Computer Vision (ICCV)* (2017) pp. 2980–2988.

[241] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[242] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, 2019 IEEE/CVF International Conference on Computer Vision (ICCV) , 3701 (2019).

[243] K. He, X. Zhang, S. Ren, and J. Sun, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016) pp. 770–778.

[244] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, in *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part VII* (Springer-Verlag, Berlin, Heidelberg, 2018) p. 3–19.

[245] D. Bank, N. Koenigstein, and R. Giryes, Deep Learning in Science (2021).

[246] J. Hu, L. Shen, and G. Sun, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018) pp. 7132–7141.

[247] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016) pp. 2921–2929.

[248] V. Tiwari and S. Fairhurst, The Astrophysical Journal Letters **913** (2020).

[249] Z. Cui, W. Chen, and Y. Chen, ArXiv **abs/1603.06995** (2016).

[250] X. Liu, B. Xu, S. Cao, and L. Zhang, Journal of Computational Physics **506**, 112944 (2024).

[251] X. Chai, W. Cao, J. Li, H. Long, and X. Sun, IEEE Transactions on Geoscience and Remote Sensing **62**, 1 (2024).

[252] R. Fang and Y. Xu, Addressing spectral bias of deep neural networks by multi-grade deep learning (2024), arXiv:2410.16105 [cs.LG] .

[253] B. L. Edelman, S. Goel, S. M. Kakade, and C. Zhang, ArXiv **abs/2110.10090** (2021).

[254] S. Hao, D.-H. Lee, and D. Zhao, Transportation Research Part C: Emerging Technologies **107**, 287 (2019).

[255] A. H. Nitz, S. Kumar, Y.-F. Wang, S. Kastha, S. Wu, M. Schäfer, R. Dhurkunde, and C. D. Capano, ApJ **946**, 59 (2023), arXiv:2112.06878 [astro-ph.HE] .

[256] D. Davis and M. Walker, Galaxies **10**, 12 (2022).

[257] M. Cabero *et al.*, Class. Quant. Grav. **36**, 15 (2019), arXiv:1901.05093 [physics.ins-det] .

[258] L. Nuttall *et al.*, Class. Quant. Grav. **32**, 245005 (2015), arXiv:1508.07316 [gr-qc] .

[259] A. H. Nitz, T. Dent, T. Dal Canton, S. Fairhurst, and D. A. Brown, Astrophys. J. **849**, 118 (2017), arXiv:1705.01513 [gr-qc] .

[260] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, Proceedings of the National Academy of Science **114**, 3521 (2017), arXiv:1612.00796 [cs.LG] .

[261] M. Toneva, A. Sordoni, R. T. des Combes, A. Trischler, Y. Bengio, and G. J. Gordon, ArXiv **abs/1812.05159** (2018).

[262] M. P. Kumar, B. Packer, and D. Koller, in *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10 (Curran Associates Inc., Red Hook, NY, USA, 2010) p. 1189–1197.

[263] F. Robinet, N. Arnaud, N. Leroy, A. Lundgren, D. Macleod, and J. McIver, SoftwareX **12**, 100620 (2020), arXiv:2007.11374 [astro-ph.IM] .

[264] S. Sahayam, J. Zakkam, and U. Jayaraman, Can we learn better with hard samples? (2023), arXiv:2304.03486 [cs.CV] .

[265] Y. Suh, B. Han, W. Kim, and K. M. Lee, in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019) pp. 7244–7252.

[266] R. Yu, Z. Dou, S. Bai, Z. Zhang, Y. Xu, and X. Bai, in *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI* (Springer-Verlag, Berlin, Heidelberg, 2018) p. 196–212.

[267] D. Arpit, S. Jastrzundefinedbski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17 (JMLR.org, 2017) p. 233–242.

[268] S. Chatterjee, in *International conference on machine learning* (PMLR, 2018) pp. 755–763.

[269] X. Wang, Y. Chen, and W. Zhu, IEEE Transactions on Pattern Analysis and Machine Intelligence **44**, 4555 (2022).

[270] R. Meyes, M. Lu, C. W. de Puiseau, and T. Meisen, ArXiv **abs/1901.08644** (2019).

[271] Y. Dauphin and Y. Bengio, CoRR **abs/1301.3583** (2013).

[272] S. Gao, M.-M. Cheng, K. Zhao, X. Zhang, M.-H. Yang, and P. H. S. Torr, IEEE Transactions on Pattern Analysis and Machine Intelligence **43**, 652 (2019).

[273] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, J. Mach. Learn. Res. **22**, 57:1 (2019).

[274] J. Wildberger, M. Dax, S. R. Green, J. Gair, M. Pürrer, J. H. Macke, A. Buonanno, and B. Schölkopf, Phys. Rev. D **107**, 084046 (2023).

[275] B. P. Abbott *et al.*, Living Reviews in Relativity **23**, 3 (2020).

[276] R. Abbott *et al.* (KAGRA, VIRGO, LIGO Scientific), Phys. Rev. X **13**, 011048 (2023), arXiv:2111.03634 [astro-ph.HE] .

[277] P. Mészáros, D. B. Fox, C. Hanna, and K. Murase, Nature Reviews Physics **1**, 585 (2019), arXiv:1906.10212 [astro-ph.HE] .

[278] M. Bailes, B. K. Berger, P. R. Brady, M. Branchesi, K. Danzmann, M. Evans, K. Holley-Bockelmann, B. R. Iyer, T. Kajita, S. Katsanevas, M. Kramer, A. Lazzarini, L. Lehner, G. Losurdo, H. Lück, D. E. McClelland, M. A. McLaughlin, M. Punturo, S. Ransom, S. Raychaudhury, D. H. Reitze, F. Ricci, S. Rowan, Y. Saito, G. H. Sanders, B. S. Sathyaprakash, B. F. Schutz, A. Sesana, H. Shinkai, X. Siemens, D. H. Shoemaker, J. Thorpe, J. F. J. van den Brand, and S. Vitale, Nature Reviews Physics **3**, 344 (2021).

[279] D. E. Holz and S. A. Hughes, ApJ **629**, 15 (2005), arXiv:astro-ph/0504616 [astro-ph] .

[280] A. K. Mehta, in *31st Texas Symposium on Relativistic Astrophysics* (2022) p. 82.

[281] B. P. Abbott *et al.*, Phys. Rev. Lett. **123**, 011102 (2019), arXiv:1811.00364 [gr-qc] .

[282] M. U. Kruckow, T. M. Tauris, N. Langer, M. Kramer, and R. G. Izzard, MNRAS **481**, 1908 (2018), arXiv:1801.05433 [astro-ph.SR] .

[283] A. Bauswein and H. T. Janka, Phys. Rev. Lett. **108**, 011101 (2012), arXiv:1106.1616 [astro-ph.SR] .

[284] L. Baiotti, Prog. Part. Nucl. Phys. **109**, 103714 (2019), arXiv:1907.08534 [astro-ph.HE] .

[285] B. P. Abbott *et al.* (LIGO Scientific, Virgo), Class. Quant. Grav. **37**, 055002 (2020), arXiv:1908.11170 [gr-qc] .

[286] D. George and E. A. Huerta, Physics Letters B **778**, 64 (2018), arXiv:1711.03121 [gr-qc] .

[287] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, Phys. Rev. Lett. **120**, 141103 (2018), arXiv:1712.06041 [astro-ph.IM] .

[288] J. Yan, M. Avagyan, R. E. Colgan, D. Veske, I. Bartos, J. Wright, Z. Márka, and S. Márka, Phys. Rev. D **105**, 043006 (2022), arXiv:2104.03961 [astro-ph.IM] .

[289] M. B. Schäfer and A. H. Nitz, Phys. Rev. D **105**, 043003 (2022), arXiv:2108.10715 [astro-ph.IM] .

[290] M. B. Schäfer, O. Zelenka, A. H. Nitz, H. Wang, S. Wu, Z.-K. Guo, Z. Cao, Z. Ren, P. Nousi, N. Stergioulas, P. Iosif, A. E. Koloniari, A. Tefas, N. Passalis, F. Salemi, G. Vedovato, S. Klimenko, T. Mishra, B. Brügmann, E. Cuoco, E. A. Huerta, C. Messenger, and F. Ohme, Phys. Rev. D **107**, 023021 (2023), arXiv:2209.11146 [astro-ph.IM] .

[291] P. Nousi, A. E. Koloniari, N. Passalis, P. Iosif, N. Stergioulas, and A. Tefas, Phys. Rev. D **108**, 024022 (2023), arXiv:2211.01520 [gr-qc] .

[292] O. Zelenka, B. Brügmann, and F. Ohme, arXiv e-prints , arXiv:2402.07492 (2024), arXiv:2402.07492 [astro-ph.IM] .

[293] E. Marx *et al.*, Phys. Rev. D **111**, 042010 (2025), arXiv:2403.18661 [gr-qc] .

[294] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, ACM Comput. Surv. **54**, 10.1145/3457607 (2021).

[295] C. Talbot and E. Thrane, Phys. Rev. Res. **2**, 043298 (2020).

[296] J. D. Romano and N. J. Cornish, Living Rev. Rel. **20**, 2 (2017), arXiv:1608.06889 [gr-qc] .

[297] W. M. Farr, J. R. Gair, I. Mandel, and C. Cutler, Phys. Rev. D **91**, 023005 (2015).

[298] S. Banagiri, C. P. L. Berry, G. S. Cabourn Davies, L. Tsukada, and Z. Doctor, Phys. Rev. D **108**, 083043 (2023).

[299] D. Williams, Class. Quant. Grav. **42**, 105012 (2025), arXiv:2401.08709 [astro-ph.HE] .

[300] P. Schmidt, F. Ohme, and M. Hannam, Phys. Rev. D **91**, 024043 (2015).

[301] D. Gerosa, M. Mould, D. Gangardt, P. Schmidt, G. Pratten, and L. M. Thomas, Phys. Rev. D **103**, 064067 (2021).

[302] D. C. Srivastava and S. K. Sahay, Mon. Not. Roy. Astron. Soc. **337**, 322 (2002), arXiv:gr-qc/0209009 .

[303] Y. Bengio *et al.*, IEEE transactions on neural networks **5 2**, 157 (1994).

[304] A. Sherstinsky, Physica D: Nonlinear Phenomena **404**, 132306 (2020).

[305] C. Dreissigacker *et al.*, Phys. Rev. D **100**, 044009 (2019).

[306] X.-T. Zhang *et al.*, Physical Review D (2022).

[307] M. B. Schäfer *et al.*, Phys. Rev. D **102**, 063015 (2020), arXiv:2006.01509 [astro-ph.HE] .