Advancing the Search for Gravitational Waves using Machine Learning

Hunter Gabbard

Submitted in fulfilment of the requirements for the

Degree of Doctor of Philosophy

School of Physics and Astronomy

College of Science and Engineering

University of Glasgow

September 2021

Abstract

Over 100 years ago Einstein formulated his now famous theory of General Relativity. In his theory he lays out a set of equations which lead to the beginning of a brand-new astronomical field, Gravitational wave (GW) astronomy. The LIGO-Virgo-KAGRA Collaboration (LVK)'s aim is the detection of GW events from some of the most violent and cataclysmic events in the known universe. The LVK detectors are composed of large-scale Michelson Morley interferometers which are able to detect GWs from a range of sources including: binary black holes (BBHs), binary neutron stars (BNSs), neutron star black holes (NSBHs), supernovae and stochastic GWs. Although these GW events release an incredible amount of energy, the amplitudes of the GWs from such events are also incredibly small.

The LVK uses sophisticated techniques such as matched filtering and Bayesian inference in order to both detect and infer source parameters from GW events. Although optimal under many circumstances, these standard methods are computationally expensive to use. Given that the expected number of GW detections by the LVK will be of order 100s in the coming years, there is an urgent need for less computationally expensive detection and parameter inference techniques. A possible solution to reducing the computational expense of such techniques is the exciting field of machine learning (ML).

In the first chapter of this thesis, GWs are introduced and it is explained how GWs are detected by the LVK. The sources of GWs are given, as well as methodologies for detecting various source types, such as matched filtering. In addition to GW signal detection techniques, the methods for estimating the parameters of detected GW signals is described (i.e. Bayesian inference). In the second chapter several machine learning algorithms are introduced including: perceptrons, convolutional neural networks (CNNs), autoencoders (AEs), variational autoen-

coders (VAEs) and conditional variational autoencoders (CVAEs). Practical advice on training/data augmentation techniques is also provided to the reader. In the third chapter, a survey on several ML techniques applied a variety of GW problems are shown.

In this thesis, various ML and statistical techniques were deployed such as CVAEs and CNNs in two first-of-their-kind proof-of-principle studies. In the fourth chapter it is described how a CNN may be used to match the sensitivity of matched filtering, the standard technique used by the LVK for detecting GWs. It was shown how a CNN may be trained using simulated BBH waveforms buried in Gaussian noise and signals with Gaussian noise alone. Results of the CNN classification predictions were compared to results from matched filtering given the same testing data as the CNN. In the results it was demonstrated through receiver operating characteristics and efficiency curves that the ML approach is able to achieve the same levels of sensitivity as that of matched filtering. It is also shown that the CNN approach is able to generate predictions in low-latency. Given approximately 25000 GW time series, the CNN is able to produce classification predictions for all 25000 in 1s.

In the fifth and sixth chapters, it is shown how CVAEs may be used in order to perform Bayesian inference. A CVAE was trained using simulated BBH waveforms in Gaussian noise, as well as the source parameter values of those waveforms. When testing, the CVAE is only supplied the BBH waveform and is able to produce samples from the Bayesian posterior. Results were compared to that of several standard Bayesian samplers used by the LVK including: `Dynesty`, `ptemcee`, `emcee`, and `CPnest`. It is shown that when properly trained the CVAE method is able to produce Bayesian posteriors which are consistent with other Bayesian samplers. Results are quantified using a variety of figures of merit such as probability-probability (p-p) plots in order to check the 1-dimensional marginalised posteriors from all approaches are self-consistent with the frequentist perspective. The Jensen—Shannon (JS)-divergence was also employed in order to compute the similarity of different posterior distributions from one another, as well as other figures of merit. It was also demonstrated that the CVAE model was able to produce posteriors with 8000 samples in under a second, representing a 6 order of magnitude increase in performance over traditional sampling methods.

Contents

List of Tables

List of Figures

List of Acronyms

GW Gravitational wave

VC Vapnik-Chervonenkis

GRB gamma-ray burst

BBH binary black hole

CW continuous gravitational waves

EM electromagnetic

TAP true alarm probability

CBC compact binary coalescence

BNS binary neutron star

NSBH neutron star black hole

PSD power spectral density

ELBO evidence lower bound

PN post-Newtonian

NR numerical relativity

IMR inspiral-merger-ringdown

Phenom Phenomenological

EOB effective-one-body

LIGO advanced Laser Interferometer Gravitational wave Observatory

LISA Laser Interferometer Space Antenna

CVAE conditional variational autoencoder

KL Kullback–Leibler

GPU graphics processing unit

LVC LIGO-Virgo Collaboration

p-p probability-probability

SNR signal-to-noise ratio

AE autoencoder

VAE variational autoencoder

LSTM Long Short Term Memory

GR general relativity

CWB Coherent WaveBurst

FAR false alarm rate

FAP false alarm probability

MH Metropolis-Hastings

ASD amplitude spectral density

FFT fast Fourier transform

CNN convolutional neural network

ROC receiver operator characteristic

MCMC Markov Chain Monte Carlo

ML machine learning

ANN artificial neural network

MAF masked autoregressive flow

GAN generative adversarial network

RF random forest

GP genetic programming

JS Jensen—Shannon

NS neutron star

BH black hole

PDF probability density function

LVK LIGO-Virgo-KAGRA Collaboration

AI artificial intelligence

GMST Greenwich mean sidereal time

CMB cosmic microwave background

`NESSAI` Nested Sampling with Artificial Intelligence

MNIST Modified National Institute of Standards and Technology dataset

Acknowledgements

I would first off like to thank my parents (Lisa and Kurt) for all that they've done for me. Without their constant support and encouragement over the course of my life, I would not be where I am today. Additionally, I would like to thank the rest of my family including my siblings (Mallory and Schuyler), as well as my grandfather (Evan Lewis).

To my friends here in Glasgow, I can confidently say that the last 4 years have been some of the best of my life. I've had so many wonderful experiences with you all; from many nights out to the pub, camping/hiking trips, climbing and random hangouts. I will dearly miss having you all in one place, but I'm sure that our paths will cross again at some point in the near future. A special thank you to Jennie Wright. I could not ask for a better best friend.

To the many advisors that I've had over my short research career including: Prof. Marco Cavaglia, Dr. Florent Robinet, Dr. Soma Mukherjee, Dr. Andrew Lundgren, Prof. Ik Siong Heng and Dr. Chris Messenger. You have all in your own way inspired me to continue in physics through to my PhD. A big thank you to my primary PhD advisor Chris, who has taught me more than I ever could have imagined I would learn. You really are a truly gifted and amazing advisor and I feel incredibly lucky to have had the opportunity to work with you.

Declaration

With the exception of chapters 1, 2 and 3, which contain introductory material, all work in this thesis was carried out by the author under the supervision of Dr. Chris Messenger and Prof. Ik Siong Heng unless otherwise explicitly stated.

The work carried out in Ch. 4, was primarily done by myself under the supervision of Dr. Messenger and Prof. Heng, along with support from co-authors Mr. Fergus Hayes and Mr. Michael Williams. Both Mr. Williams and Mr. Hayes had carried out preliminary studies and produced some initial code to work from. Mr. Williams, along with myself, produced the machine learning code used in the analysis. I produced the matched filtering code used in the analysis. All co-authors on the published paper version of this analysis [1] have also contributed to the text of the published manuscript. Both Dr. Messenger and Prof. Heng provided advice on the thesis chapter version of the published manuscript.

The work carried out in Ch. 5 and Ch. 6 was lead by myself under the supervision of Dr. Messenger and Prof. Heng, along with support from co-authors Mr. Francesco Tonolini and Prof. Roderick Murray-Smith. Both Mr. Tonolini and Prof. Murray-Smith provided guidance on the machine learning methods used, some initial machine learning code to work from, as well as input on the text released in the public version of the manuscript found here [2]. Dr. Messenger and Prof. Heng also provided input to the text in the public form of the manuscript, as well as advice on the thesis chapter version.

The final conclusions chapter (Ch. 7) was written by myself, with advice from Dr. Messenger and Prof. Heng.

Chapter 1

An Introduction to Gravitational Waves, Search Methods and Parameter Estimation Techniques

1.1 Gravitational Wave Detections

Since it's inception, the LVK has carried out several observation runs. Initial operations ran from 2002 to 2010, but no GWs were detected during this time period. During the first observing run in the advanced detector era (September 2015 - January 2016), the LVC detected a total of 3 BBH mergers including: GW150914 [3], GW151226 [4] and GW151012 [5]. GW150914 was the very first detected BBH with a SNR value of ~ 25.1 [6]. GW151012 was originally labeled as a less significant potential GW detection (LVT151012) due to its high false alarm rate, but was subsequently upgraded to a confirmed GW event in the GWTC-1 catalogue [7] because its false alarm rate was less than 1 per 30 days (a threshold determined by the LVC). The change in false alarm rate for GW151012 can largely be attributed to various improvements made to all search algorithms used in the first observation run between the initial detection and up to publication of the GWTC-1 catalogue paper (for further details, see [7, 8]).

During the second observing run (November 2016 - August 2017) the LVC detected an additional 7 BBHs with total masses between $\sim 18.6 M_{\odot}$ and $\sim 85.1 M_{\odot}$. The second observation

run excitingly also saw the very first detection of a BNS event. The BNS event had the highest network SNR of any event over all of O1 and O2 (~ 32.4). Interestingly, there was also a large non-astrophysical noise transient which overlapped with a portion of the BNS event in the LIGO Livingston detector. This noise transient was successfully mitigated through an excising technique known as time-domain gating [9]. Approximately 1.7s following GW170817, a gamma-ray burst (GRB) (GRB170817) was observed across multiple wavelengths of the electromagnetic (EM) spectrum over the course of several weeks [10]. The delay between GRB170817 and GW170817 has been used to place strong constraints on various physical phenomena including: the speed of gravity, Lorentz invariance and tests of the equivalence principal [10]. Additionally, given that GW observations provide direct estimates on the redshift and luminosity distance of the system, it was shown in [11] that these observations may be used to provide the first gravitational wave based independent measurement on the Hubble constant.

Most recently, during the first half of the third observing run (April 2019 - March 2020) the LVK collaboration made an additional 39 confirmed GW event detections [5, 7] . The increase in number of detections can largely be attributed to higher sensitivities of the detectors during this observation run over previous runs, with a BNS range[1] of 108Mpc, 135Mpc and 45Mpc for Hanford, Livingston and Virgo respectively (approximately 80Mpc, 80Mpc and 25Mpc during O2). The GWTC-2 catalogue contains detected signals with component masses lower and higher than the lowest and highest component masses contained in all of GWTC-1. The most up-to-date merger rate constraints according to GWTC-2 were also updated to be $\sim 23.9\text{Gpc}^{-3}\text{yr}^{-1}$ for BBHs and $\sim 320\text{Gpc}^{-3}\text{yr}^{-1}$ for BNSs (originally $9.7-101\text{Gpc}^{-3}\text{yr}^{-1}$ for BBHs and $110-3840\text{Gpc}^{-3}\text{yr}^{-1}$ for BNSs for O2).

In January of 2020, the LVC collaboration reported the first detection of two NSBH events (GW200105,GW200115) [15] . The primary component masses of both events are $\sim 8.9\text{M}_\odot$ and $\sim 5.7\text{M}_\odot$ respectively, whose mass values are both above the maximum allowed mass of a neutron star (NS) defined in [16], so may therefore be likely classified as black hole (BH)s. The

[1]The BNS range is a scalar value which is often used to represent the performance of the LVC detectors. It is quantified by determining the luminosity distance at which a single detector could detect a $1.4\text{M}_\odot$ BNS pair with an SNR ≥ 8 averaged over the sky location and orientation of the source with respect to the detector. The range is dependent on a number of factors including: source mass/spin and the noise curve [12] of the detector. See [13, 14] for more details.

secondary masses of each event were given as $\sim 1.5M_{\odot}$ and $\sim 1.9M_{\odot}$ respectively and were reported to be within the range of known NSs [17].

As improvements are made to the LVC detectors over the coming years, it is expected that the rate of detections will increase dramatically [18]. It is predicted that at design sensitivity, the LVC will observe $\mathcal{O}(100s)$ of events per year [18]. Current methods for both GW detection and parameter estimation, while optimal in many cases, are often computationally expensive [6, 19, 20]. Algorithms which produce estimates on source parameter values of GW signals can take upwards of weeks to run (see Tab. 5.1 and Tab. 5.2 in Ch. 5). Given that follow-up observations of GW EM components heavily depend upon GW sky location alerts from the LVC and the rapid decay of GW EM signatures [21], there is an urgent need for faster techniques which can not only identify the presence of GW signals in detector data, but also identify source parameter values like the sky location of a GW event.

1.2 Multi-Messenger Astronomy

After a GW signal has been identified, alerts are sent out to EM partners around the globe in order to perform follow-up observations. Astronomical partners include instruments which look across the whole range of the EM spectrum: Radio, Microwave, infrared, visible light, ultra-violet, X-ray and gamma ray. A full Bayesian posterior (Sec. 1.7) is generally produced on all viable GW candidates. In addition to the full Bayesian analysis, the collaboration is also able to produce low-latency parameter estimation products (e.g. sky maps) using tools such as `Bayestar` [22], where an analysis using `Bayestar` is shown in Fig. 1.1. `Bayestar` operates under the assumption that a large degree of the information contained in a GW signal is encapsulated within a small number of data products produced by the search matched filtering process, namely: the time, amplitude and phase of the signal at each detector. Using a simplified likelihood function and the Fisher information matrix [23], `Bayestar` is able to produce estimates on a limited number of source parameters (sky location, distance and orientation) in under a few minutes which provides a good approximate of the full Bayesian posterior [24].

Prompt sky location, distance and orientation data products using tools like `Bayestar` and

`Bilby` along with observations from EM partners can provide new insights into fundamental astrophysical processes. For example, as BNS signals reach the final stages of the inspiral phase of the merger, the internal structure of the sources has more of a pronounced effect on the resulting GW signal. Information on tidal disruption processes may be gleamed from this part of the GW signal in combination with prompt observations from the EM spectrum [25]. EM follow-up analysis can also be used in tandem with Bayesian analysis in order to produce inferences on the Hubble constant, though may also be performed without an EM counterpart using galaxy catalogues [26]. A Hubble constant measurement can be done by obtaining accurate estimates on the luminosity distance directly from the GW signal through Bayesian inference and using EM partners to identify a likely host galaxy, whereby correct identification of the host galaxy may benefit from low-latency alerts. For GW170817, LVC and EM partners were able to infer a Hubble constant value of $\sim 69^{+17}_{-8}\text{kms}^{-1}\text{Mpc}^{-1}$ [9, 11, 27].

The arrival time (along with an accurate estimation of the luminosity distance) of a GW event can be compared to the observation time of the GRB from a BNS merger counterpart. Comparing arrival times of both components allows us to test the effect gravitational potentials have on BNS EM radiation and BNS GWs (equivalence principle), as well as the speed of gravity [10]. In addition, we can perform tests on the accuracy of general relativity itself through residual noise waveform subtraction tests, inspiral-merger-ringdown consistency tests and parameterised tests of GW generation under a Bayesian framework [28]. Depending on the duration and SNR of the signal, accurate constraints may be placed on the graviton Compton wavelength and non-general relativity (GR) polarization states. Other tests of GR performed over both event catalogues GWTC-1 and GWTC-2 are listed in great detail in [28, 29, 30].

1.3 The Weber Bar Detector and the Hulse-Taylor Pulsar

By the early 1950s technology had progressed enough such that serious attempts at experimentally verifying the existence of GWs by Einstein were possible. One of the earliest and most well-known attempts at doing so was by Joseph Weber at the University of Maryland where he used an instrument known as a resonant-mass detector (Weber Bar Detector) [31]. The We-

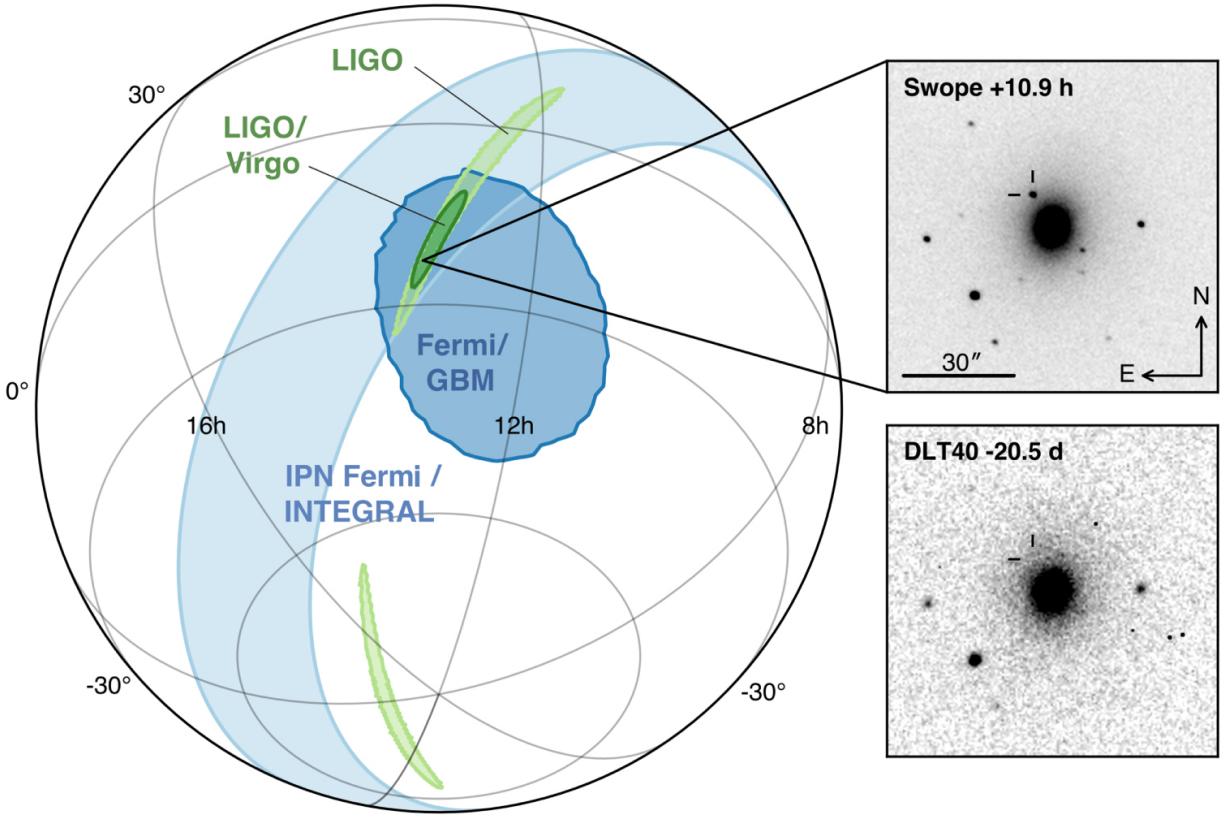


Figure 1.1: Sky localization for the first confirmed detection of a BNS merger by the LVC. Areas shaded in green are the data products from `Bayestar` using both LIGO alone and LIGO/Virgo combined, dark blue are predictions from Fermi/GBM and light blue are predictions from IPN Fermi/INTEGRAL. Black and white images on right-hand side are visible light measurements of a bright event around the galaxy NGC 4993 thought to contain the afterglow of the BNS event. This figure was produced by the authors of [21].

ber Bar operates on the principle that when a GW impinges on the bar (usually made of some type of metal alloy, in this case high Q aluminium[2]) at a specific frequency, since the bar is a harmonic oscillator which is driven by the Riemann curvature tensor, it will cause the bar to resonate. If the frequency of the GW is equivalent to the natural resonant frequency of the bar, the bar will oscillate at a higher amplitude than if the signal were at any other frequency, thus a GW would theoretically be detectable [32] Weber made the claim in 1968 that there was "good evidence" for several detections made by his experiment [32], but unfortunately no others were able to reproduce his results. Although follow-up results from other independent studies were disappointing, Weber's work encouraged many others to build their own improved experiments with breakthrough technological developments at the time and kick-started the subsequent field of GW detection [33]. Fortunately, in the subsequent years after Weber's first published results, there would come the first indirect observational evidence for the existence of GWs.

In 1975, Russell Hulse and Joseph Taylor made the first direct observation of a binary pulsar, which subsequently won them the 1993 Nobel Prize in Physics [34]. Both Hulse and Taylor observed that the orbital period of the binary pulsar appeared to experience orbital decay as a function of time. The orbital decay was thought to likely be attributed to a loss of energy in the system due to GW radiation predicted by GR. The observed period decay as a function of time (in years) is represented in Fig. 1.2. As can be seen in the figure, there is a striking level of agreement between the theoretical decay curve predicted by Einstein's GR and the observations made by Hulse and Taylor. Importantly, this work was also one of the first pieces of indirect observational evidence for the existence of GWs. Pioneering work by both Hulse-Taylor and Weber, spurred the development of advanced GW detectors aimed at directly observing GW events. In the following section we will discuss how the current generation of GW detectors works and detects GWs.

[2]Q is approximately equivalent to the ratio of the initial energy stored in the oscillator to the energy lost over one radian of the oscillation cycle.

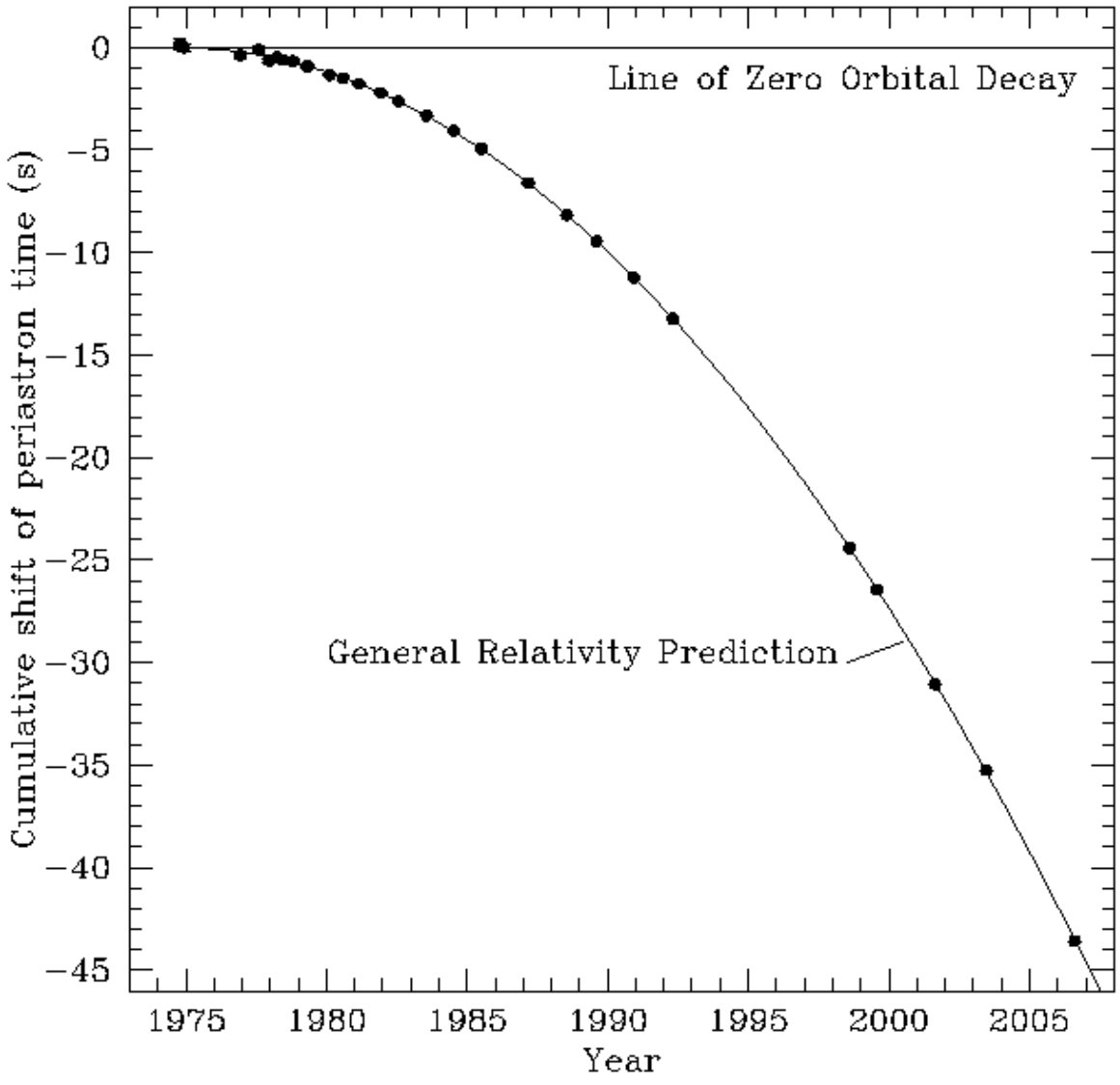


Figure 1.2: Illustrated is Hulse and Taylor's observations of binary pulsar PSR B1913+16 orbital decay as a function of time in years. The orbital decay is quantified by the total cumulative amount the binary system has been offset from it's first observation in 1975 with respect to the binary's perihelion point (black dots). The solid black curve is representative of the theoretical decay curve predicted by GR. This figure was produced by the authors of [35].

1.4 Ground Based Interferometric Detectors

The current generation of GW ground-based detectors in the LVK are composed of three observatories, two in North America (Hanford, Washington State and Livingston, Louisiana) [36] and one in Pisa, Italy (Virgo) [37]. There are also other ground-based detectors in Hannover, Germany (GEO) [38] and Kamioka, Japan (KAGRA) [39]. In addition to ground-based detectors there are eventual plans to build space-based observatories such as the Laser Interferometer Space Antenna (LISA) [40] and TianQin [41] which will search for super massive BBHs (among other sources). Each detector (with LISA and TianQin being the exception) can be thought of as a large-scale Michelson-Morley Interferometer [42] composed of two arms orthogonal to each other. Each arm of the LIGO detectors is 4km in length, with the Virgo arms being slightly shorter in length at 3km in length.

A simplified schematic of an interferometric GW detector is shown in Fig. 1.3. In the schematic, it is shown that a 1064nm laser beam is emitted from a laser on the left-hand side, it then passes through a phase modulator (PMOD) and enters the power recycling cavity (PRM). This effectively boosts the power of the signal. The laser then passes through a beam splitter (BS), which splits the laser beam path into two separate parts. Each part travels through an input test mass and hits end test masses at the end of both interferometer arms. The beams are then caught in a Fabry-Perot cavity [43] which acts to extend the distance traveled of the laser light photons, as well as the power [44]. In other words, the cavity "stores" the photons for a long period ($\sim$ 1 ms) which allows a potential GW signal more time to interact with the photons, thus increasing the sensitivity of the interferometer at low frequencies. Some laser light escapes back down both arms and recombines at the BS where the recombined beam passes through a signal recycling mirror (SRM). Finally, the beam hits a set of photodiodes (PD) which produce the interferometer readout, where the readout is also a measure of the phase difference between photons from both arms. The phase difference information is encoded in the interference pattern on the readout of the the detector photodiodes, which is the final output of the detectors.

Starting from the well-known postulate from relativity that the distance between two points in spacetime, known as the interval, along the path of a light ray travelling along the coordinate

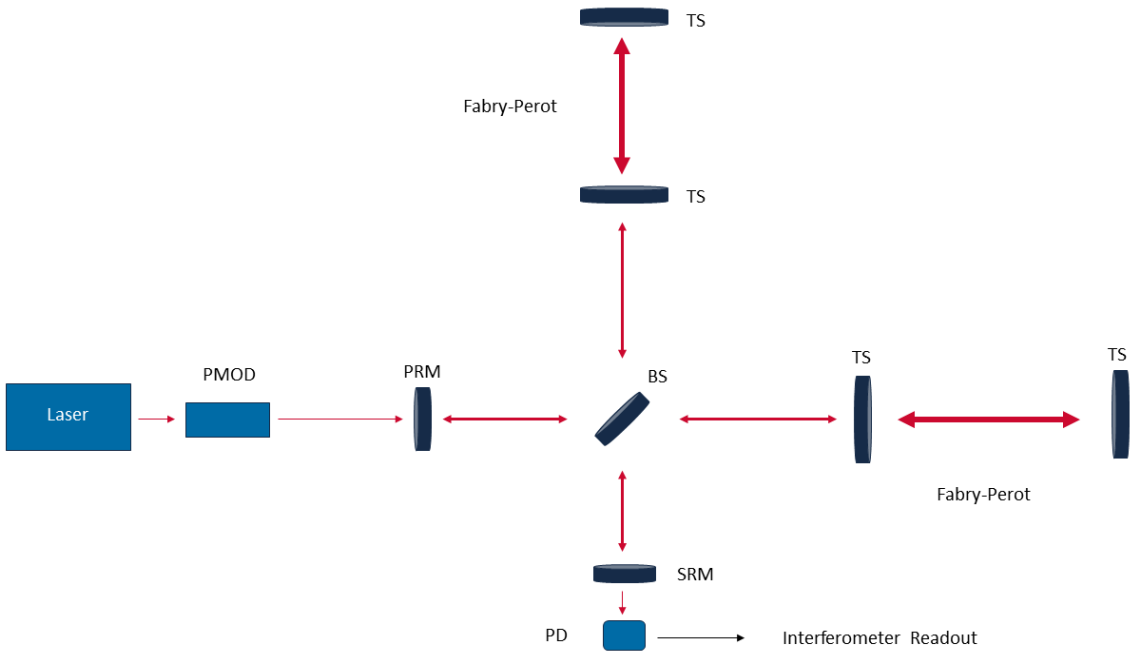


Figure 1.3: An illustration of the LIGO detectors. An illustration of the LIGO detectors. The detectors operate by first emitting photons from an initial laser port. Photons emitted from the laser pass through a beam splitter and down two orthogonal arms of the detectors in the form of vacuum sealed beam tubes guided by mirror optics. The photons then hit test mass mirrors at both ends and are caught in what is known as a Fabry-Perot signal recycling cavity [43]. A Fabry-Perot cavity acts to effectively increase the sensitivity of the arms by positively modulating the amount of time spent by the light in the arm, which consequently also increases the laser power in the arm [44]. After reflecting back and forth in the Fabry-Perot cavities, photons are released from the Fabry-Perot cavities and return to the beam splitter, subsequently recombining and are recorded on a set of photodiodes which measure the phase difference between photons from both encoded in the interference pattern on the readout of the detector photodiodes, which is the final output of the detectors.

vector defined by $dx^0, dx^1, dx^2, dx^3 = t, x, y, z$ may be expressed as

$$ds^2 = 0 = g_{\mu\nu} dx^\mu dx^\nu. \tag{1.1}$$

It can be shown through some algebraic manipulation (explained in [45]) that we can quantify the light travel time difference of photons originating from the initial laser port traveling up and down the two interferometer arms as

$$\Delta\tau(t) = h(t)\frac{2L}{c} = h(t)\tau_{\text{rt0}}. \tag{1.2}$$

where τ_{rt0} is the return trip time down one arm and the phase difference being

$$\Delta\phi(t) = h(t)\tau_{rt0}\frac{2\pi c}{\lambda}. \tag{1.3}$$

Here we can clearly see that the phase difference between the two light signals is scaled by the length of the interferometer arms L. The detectors are tuned through control systems such that the photons arriving back at the final readout port of the detector act to destructively interfere with each other (i.e. create an interference pattern on a dark fringe). If a GW impinges on the detector it will compress one arm while stretching the other arm. There will then be a detectable difference in phase between the light traveling down both arms. Due to this phase difference, the light recombining at the beam splitter will no longer destructively interfere and a signal will appear on the photodetectors in the form of a interference pattern [46]. In the next section, we will discuss how the sensitivity of the detectors may be influenced by the orientation of the detectors with respect to the GW source.

1.4.1 Detector Response

The LVC detectors are not equally sensitive to all parts of the sky. Mathematically, the sky dependent sensitivity of the detectors may be expressed through their antenna patterns. The antenna pattern of a detector is dependent upon the location and wave polarisation of the GW source with respect to the detector. The antenna patterns themselves have a direct impact on the

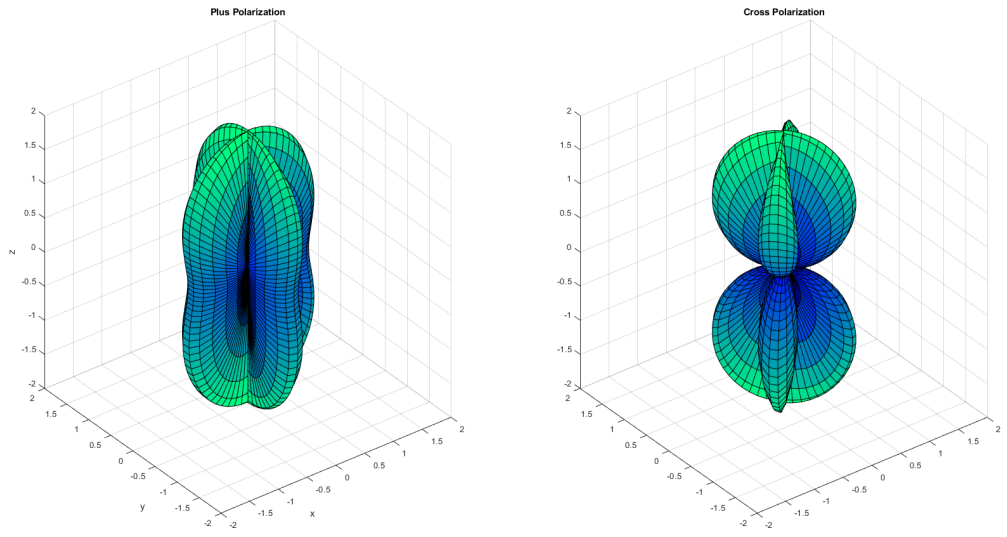


Figure 1.4: An illustration of the LVC detector antenna patterns for both the $h_\times$ and h_+ GW polarisations. The detector itself would lie in the x-y plane with one arm along the x-axis and the other along the y-axis.

amount of strain measured by the LVC detectors.

As shown in [47], the antenna pattern may be expressed as

$$\mathrm{F}_+ = -\frac{1}{2}(1+\cos^2\theta)\cos 2\varphi\cos 2\psi - \cos\theta\sin 2\varphi\sin 2\psi \tag{1.4}$$

$$\mathrm{F}_\times = +\frac{1}{2}(1+\cos^2\theta)\cos 2\varphi\sin 2\psi - \cos\theta\sin 2\varphi\cos 2\psi \tag{1.5}$$

where θ is the azimuthal angle and φ is the polar angle, both with respect to a reference frame fixed at the center of the Earth. Parameter ψ is the polarisation angle which is defined as being counter-clockwise around the direction of GW propagation and is the angle from the line of nodes to the x-axis of the GW source reference frame. We note that both θ and φ may also be used to convert to the standard right ascension $\alpha = (\varphi +$ Greenwich mean sidereal time of GW signal arrival time) and declination $\delta = (\pi/2) - \theta$ coordinates. Parameters θ, φ and ψ are all Euler angles which describe the frame of the binary system with respect to the detector. The shape of the antenna response is illustrated pictorially in Fig. 1.6 in a Cartesian coordinate system, where the detector arms lie along the x-y plane perpendicular to each other. Areas where

the detector is least sensitive to GWs are given by the null areas of Fig. 1.6 and occur when both $\psi = 0$ and $\iota = 0$. Areas where the detectors are most sensitive correspond to systems where $\iota = 0$ (i.e. GW sources directly above and below the detectors).

The GW strain of both the "plus" and "cross" polarized portions (Eq. 1.19, Eq. 1.20) may be generalised for a given polarization angle with respect to the direction of GW propagation as [48]

$$h'_+ = h_+ \cos 2\psi - h_\times \sin 2\psi, \tag{1.6}$$

$$h'_\times = h_+ \sin 2\psi + h_\times \cos 2\psi. \tag{1.7}$$

Measured GW strain is then shown to be a summation of $h'_\times$ and h'_+ attenuated by antenna patterns $F_\times$ and F_+ and is given by

$$h(t) = F_\times(\theta, \psi, \phi) h'_\times + F_+(\theta, \psi, \phi) h'_+, \tag{1.8}$$

Since the antenna patterns are sky dependent, they are therefore also time dependent. They are time dependent because the Earth rotates as a function of time and thus the orientation of the antenna pattern for a given detector therefore also changes. For example, very long signals (e.g., continuous gravitational waves (CW) signals, Sec. 1.6.3)) experience time varying antenna responses as the Earth rotates and search methods are designed to take this into account [49]. In the next section, we will discuss how the detectors are also influenced by non-astrophysical sources of strain.

1.4.2 Detector Noise

The LVC detectors, in addition to being incredibly sensitive to strain from GW signals, are also sensitive to an abundant number of non-astrophysical noise sources. These noise sources can produce periods of excess power in measured detector data which may limit the sensitivity of the detectors. Some common noise sources include gravity-gradient noise, seismic noise, thermal noise and quantum shot noise.

In practice, the performance of the detector may largely be characterized by a quantity known as the power spectral density (PSD). Assuming the absence of a GW signal, the output of the detector may be assumed to be equivalent to the detector noise as a function of time $n(t)$ sampled at regular intervals of Δt. We define the auto-correlation function as

$$K(t_1,t_2) \equiv \mathrm{E}[n(t_1)n^*(t_2)], \tag{1.9}$$

where E is the expectation value over an ensemble of realisations of the noise and $*$ is the complex conjugate. Assuming stationary noise, we can write K as being merely dependent on $\tau \equiv |t_1 - t_2|$. As shown in [50, 51], the PSD of the detector may thus be written as the Fourier transform of the auto-correlation function $K(\tau)$

$$S_n(f) \equiv \frac{1}{2}\int_{-\infty}^{\infty} K(\tau)e^{2\pi i f\tau}d\tau = \lim_{\Delta t\to 0} 2\sigma^2\Delta t,\ f \geq 0, \tag{1.10}$$

where frequencies are given to be greater than zero and σ^2 is the variance. We will now describe in detail the types of noise sources which may affect the performance of the detector PSD.

Seismic Noise

Seismic noise largely affects the sensitivity of the detectors in the low frequency regime ($\sim 10^{-2} - 10^2$Hz [53]) due to a variety of sources including: earthquakes, anthropogenic motion and wind. Earthquakes produce sets of waves (p,s,r-waves) which travel both through the Earth's core/mantel and also along the surface of the earth [54]. When one of these seismic waves hits the detectors they can induce horizontal ground motion on the optical components of the detector. In order to isolate the detector optics from horizontal ground motion ($\sim 0.03 - 0.1$Hz), optics are suspended on multi-layered seismic isolation stacks [55]. Between $\sim 1 - 3$Hz anthropogenic noise can cause short duration noise transients in the detector output. Sources of anthropogenic noise may result from individuals walking around in the LVC control rooms or large trucks passing on a nearby highway [56]. Wind greater than $10 - 20$Mph can also adversely influence the detector sensitivity at frequencies of $0.15 - 15$Hz [57].

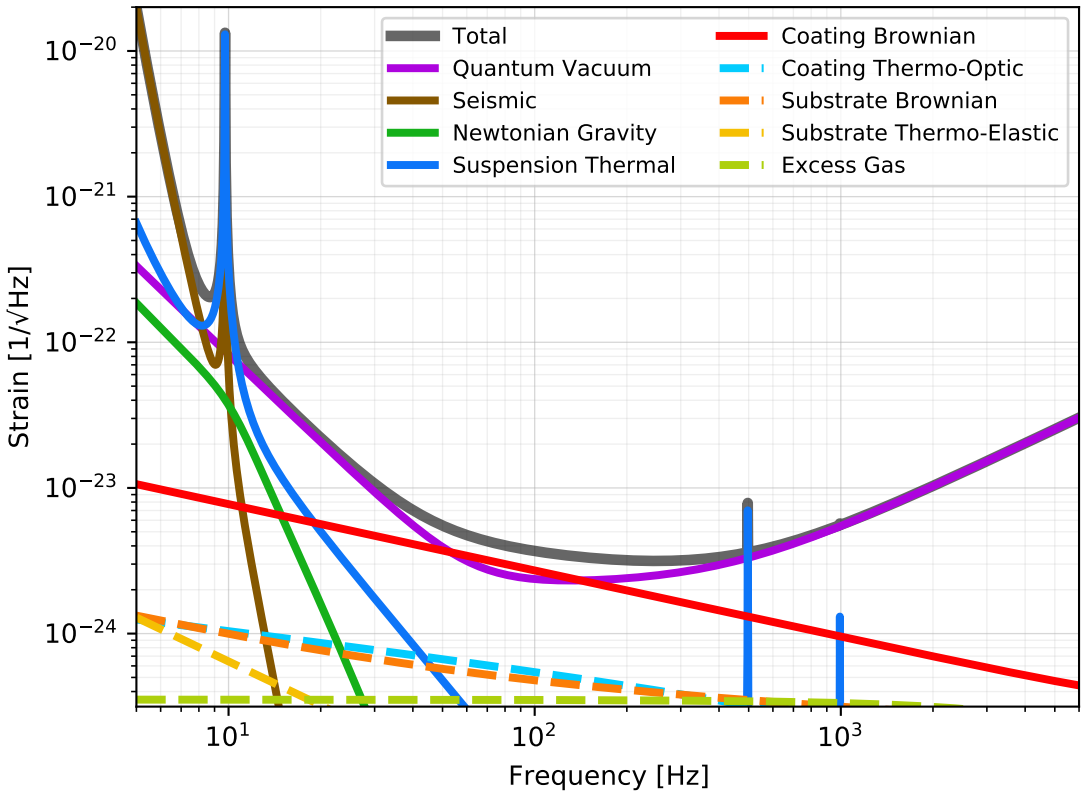


Figure 1.5: The theoretical design sensitivity noise budget curves for Advanced LIGO. As can be seen in the illustration, lower frequencies are largely dominated by seismic motion, mid-range frequencies are dominated by thermal coating Brownian motion on the mirrors as well as quantum vacuum noise between 10 and 100 Hz and higher frequencies are dominated by quantum vacuum shot noise. The plot was generated using the `pygwinc` computing package [52].

Thermal Noise

Thermal noise may be classified into two distinct types: suspension and Brownian coating thermal noise and primarily limits detector sensitivity in the frequency band of $10-500$Hz. Suspension noise results from thermal motion in the suspension fibers which can induce motion into the detector mirrors [58]. Brownian coating noise results from thermal fluctuations in detector mirror coatings [59]. Both of these sources may be quantified through the application of the fluctuation-dissipation theorem as shown in [58]. Materials for the mirror coatings are chosen such that they have minimal light absorption at the wavelength of the laser [60]. Possible mitigation strategies for reducing thermal noise also involve careful choice of coating thickness, as well as the use of Cryogenic systems for cooling the suspensions/optics of the detector [61].

Quantum Noise Sources

There are two sources of quantum noise in the detectors: quantum shot noise and quantum thermal radiation pressure noise. Both are produced by internal measurement/readout processes. Quantum shot noise is related to from the wave packet-like behavior of light as it travels through a medium. Since the distribution of photons arriving within a time interval is governed by Poisson statistics, we know that the uncertainty on the number of photons arriving at the detector photodiodes after is proportional to the square root of the expected number of photons arriving within that same time interval (optical power). We can express mathematically the amount of strain induced by shot noise on the detectors h_{S}, where subscript s stands for shot noise. given by

$$h_{\mathrm{S}} = \frac{1}{L}\sqrt{\frac{\hbar c\lambda}{2\pi P}}, \tag{1.11}$$

where L is the arm length of the interferometer, $\hbar$ is Planck's constant, λ is the laser light wavelength, P is the power of the laser and c is the speed of light [62].

The second type of noise source is quantum radiation pressure noise. Radiation pressure noise arises from the effect of photon momentum transfer onto the test mass mirrors of the detector. When a photon from the detector laser hits a mirror, it transfers some momentum to that mirror. Since all photons do not hit the mirror at the exact same time, there is some

variability of pressure exerted on the mirror as a function of time. This moves the mirror in a variable manner which leads to a change in the detector arms length and thus the noise on the output. This can be mathematically expressed as the amount of strain induced on the detector due to radiation pressure noise h_{R} by

$$h_{\mathrm{R}} = \frac{1}{Lmf^2}\sqrt{\frac{\hbar P}{2\pi^3 c\lambda}}, \tag{1.12}$$

where m is the mass of the mirror, R stands for radiation pressure and f is the frequency measured in the GW detector [62].

Shot noise may be partially mitigated through increasing the circulating light power of the laser, since it is known that sensitivity of the detector to GWs is proportional to the laser power, whereas shot noise is proportional to the square root of the optical power [63]. Unfortunately, as the optical power of the laser is increased, so to does thermal radiation pressure noise, which sets an effective upper limit on optical laser power in the detector. Radiation pressure noise can be reduced by either increasing the mass of the mirrors m, or the length of the detectors L, though it should be noted that any increase in either of these values comes with added technological and sheer monetary cost constraints. For a more detailed description of shot noise, see [62].

Gravity-Gradient Noise

Gravity-gradient noise (or Newtonian noise) is noise which results from small stochastic perturbations to the gravitational field background in and around the LVC detector test masses. Some sources of gravity gradient noise include: seismic noise and atmospheric fluctuations (specifically, changes in air pressure which carry with it changes in air density) [64]. Gravity-gradient noise decreases steeply with increasing frequency and is a primary limiting factor in detector sensitivity below frequencies of 1Hz [50].

1.5 General Relativity and Gravitational Waves

GWs were predicted by Einstein in his theory of GR well over 100 years ago [65]. In his theory, Einstein shows that the more massive an object is, the more curvature in spacetime that object creates. This curvature then has an effect on the motion of objects which encounter it. This effect is summarized succinctly by John Archibald Wheeler where he states that "Spacetime tells matter how to move; matter tells spacetime how to curve". Curvature may be quantified by first defining a term known as the Riemann curvature tensor $R^{\rho}_{\sigma\mu\nu}$. The Riemann tensor describes the change experienced by a vector which has been parallel transported over a curved manifold[66]. Einstein formalises the relationship between matter/energy and the curvature of space-time in his field equations as

$$G_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}, \tag{1.13}$$

where Λ is the cosmological constant (scalar measurement describing the energy density of space), $g_{\mu\nu}$ is the metric which describes the geometric structure of space-time, G is Newton's gravitational constant, c is the speed of light, and $T_{\mu\nu}$ is the stress energy tensor which describes the density, direction, and flow of energy in space-time. The Einstein tensor, $G_{\mu\nu}$, is defined as

$$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu} \tag{1.14}$$

where $R_{\mu\nu}$ is the Ricci curvature tensor (a contraction of the Riemann curvature tensor) and R is the Ricci scalar defined as the trace of the Ricci curvature tensor with respect to the metric.

The Einstein field equations described by Eq. 1.13 can unfortunately only be solved exactly analytically in a limited number of situations. Some solutions include the Schwarzschild solution for a non-spinning singularity (where a singularity is defined as a point in spacetime where there is predicted to be infinite curvature [66]) and the Kerr solution for a spinning singularity.

In the regime of small perturbations to spacetime, considering we would like to explore the behavior of non-linear, time-dependent systems in terms of Einstein's field equations, we need to put his equations into a more linear form. This can be done by describing the spacetime

metric $g_{\mu\nu}$ in terms of an easily computable known solution in flat spacetime with the addition of some small perturbation, where GWs may be defined as small perturbations over the curved background spacetime metric $g_{\mu\nu}$. In Euclidean space, the metric is generally described by the identity matrix in Cartesian coordinates. However, in GR we have to add the time dimension and in flat spacetime this can be described by the Minkowski metric tensor $\eta_{\mu\nu}$ given by

$$\eta_{\mu\nu} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{1.15}$$

where each column and row represent a dimension of spacetime (from left to right and top to bottom: ct, x, y, z) where t denotes time, c is the speed of light, and x, y, z denote the 3 spacial dimensions. Although the Minkowski metric can be written in non-diagonal forms depending on choice of coordinates, this diagonal form is chosen because it is computationally easy to invert and simple to compute the determinant of the matrix.

Since a GW is a perturbation, we can write the metric tensor of a small perturbation in flat spacetime as

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu}, \tag{1.16}$$

where $h_{\mu\nu}$ is the GW perturbation tensor ($|h_{\mu\nu} \ll 1|$).

Using $g_{\mu\nu}$, we find that one of the solutions Einstein's field equations may be written as a plane-wave solution given by

$$h_{\mu\nu} = \mathrm{Re}[A_{\mu\nu} e^{ik_\alpha x^\alpha}], \tag{1.17}$$

where $A_{\mu\nu}$ is a amplitude tensor made up of multiple independent components and $h_{\mu\nu}$ is a sinusoidal wave traveling along the null wavevector k [48].

Exploiting guage freedoms [66], we can shift to the Transverse Traceless gauge (since this simplifies the plane-wave solution of Einstein's field equations to a simplified metric which is

only made up of two unique components) and $h_{\mu\nu}$ can be rewritten as

$$h_{\mu\nu} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_+ & h_\times & 0 \\ 0 & h_\times & -h_+ & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{1.18}$$

where h_+ and $h_\times$ are representative of the two polarization states of a GW which are orthogonal to one-another [51, 66].

As derived in [67], assuming that the source of the GW is emitted from a binary astrophysical system (Sec. 1.6.1) with component masses m_1, m_2, the leading order terms of the plus and cross polarization states can be expressed as being equivalent to

$$h_+ \equiv \frac{1}{d}(1+\cos^2\iota)2\mu(M\Omega^{2/3})\cos(2(\Omega t - \phi_0)) \tag{1.19}$$

$$h_\times \equiv \frac{1}{d}\cos\iota 2\mu(M\Omega^{2/3})\sin(2(\Omega t - \phi_0)) \tag{1.20}$$

where d is the distance to the source, ι is the inclination angle of the binary with respect to an observer, M is the total mass of the system, ϕ_0 is the initial phase of the system, t is time, μ is the reduced mass given as $(m_1 m_2)/(M)$ and Ω is an approximation in a non-relativistic regime of the orbital angular frequency given by

$$\Omega = \sqrt{\frac{M}{a^3}} \tag{1.21}$$

where a is the separation distance between the two component masses of the system. The separation distance between the two bodies will decrease and the orbital angular frequency will increase as a function of time via GW emission.

The effect that a passing GW waveform has on a set of freely floating test particles as a function of the waveform's phase ϕ for a given polarization state is illustrated in Fig. 1.6. This effect on freely floating test masses is what is measured by LVK GW detectors in the form of strain h. As shown derived in [48], the strain a GW induces on only two free point masses can

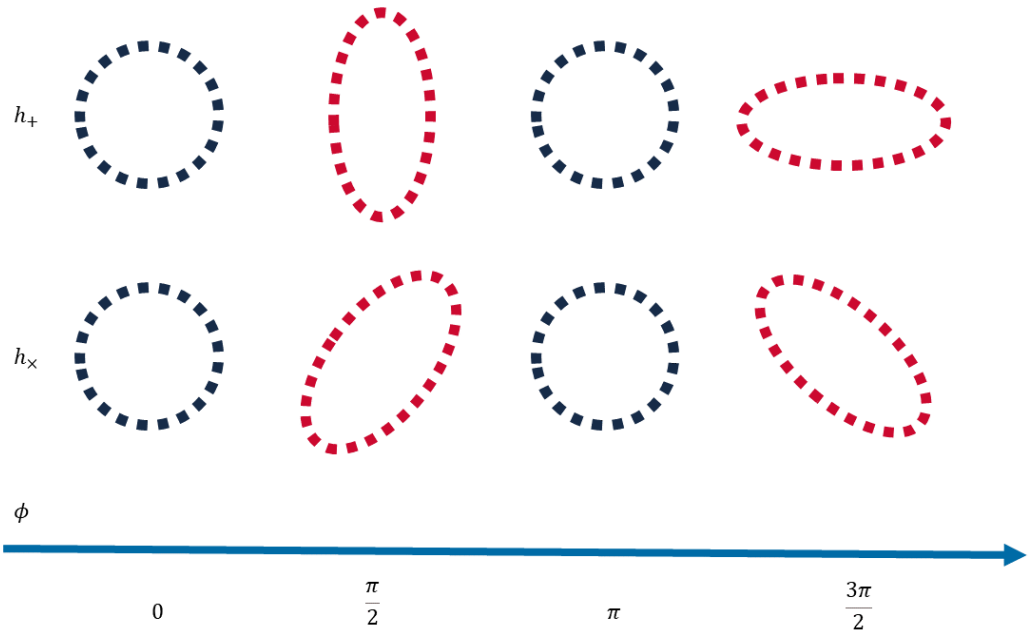


Figure 1.6: An illustration of the h_+ and $h_\times$ polarizations of a GW signal impinging on a set of freely floating test masses as a function of the phase of the GW ϕ from 0 to $3\pi/2$.

be expressed as

$$h(t) = \frac{2\Delta L}{L}, \tag{1.22}$$

where $h(t)$ is the strain amplitude of the GW as a function of time, ΔL is the absolute change in distance between two point masses induced by the GW and L is the distance between two point masses in the non-presence of a GW.

For a full derivation concerning the generation and propagation of GW signals, I refer the interested reader to [68]. Now that we have defined how GWs propagate and induce strain on freely floating test masses, we will introduce in the following section the various sources and methods for detecting GWs.

1.6 Astrophysical Sources and Search Methods

There are a variety of sources which produce GW signals. Most signals are not large enough to be seen by the LVC detectors, but some are indeed sufficiently strong enough to be detected. GW signals may be categorized into 4 distinct types including: compact binary coalescence (CBC),

burst, continuous and stochastic GWs. In this section I will explain the unique characteristics which describe each of these signal types. I will also describe several methods used by the LVC to search for signals from CBC, GW, burst and stochastic GWs.

1.6.1 Compact Binary Coalescences

CBC signals arise from the collision of massive (of order $\sim M_{\odot}$) compact binary objects (such as BHs and NSs) moving at relativistic speeds. NSs are defined as being the leftover cores of dead stars which have exploded in a supernovae and then collapsed down into an object roughly the mass of our sun and with a radius of $O(10)$ km. CBC systems can include BBHs, NSBH pairs and BNSs. A CBC signal waveform is described by three components: the inspiral, merger and ringdown phase and can be approximated using a combination of post-Newtonian theory [69, 70, 71, 72], the effective-one-body formalism [73], phenom waveform approach [74, 75**?**], and numerical relativity simulations [76].

BBH signals are parameterised by 15 different parameters (discounting eccentricity [77]). Two of these parameters describe the two component masses of each compact object in the binary system (m_1,m_2) and are sometimes commonly combined in an expression known as the chirp mass given by

$$M_{\rm c} = \frac{(m_1 m_2)^{3/5}}{(m_1 + m_2)^{1/5}}. \tag{1.23}$$

Other parameters include: the time at which the binary coalesced, luminosity distance, phase of the waveform at coalescence, sky location (right ascension, declination), inclination angle, polarization angle, spin magnitudes, tilt angles, spin azimuthal angle and azimuthal position. BNS and NSBH events are also parameterized by the same BBH parameters mentioned above, but their waveform is additionally impacted by the internal structure of the NS. The affect the NS internal structure has on the GW waveform may be parameterized in the form of additional tidal parameters given in [78].

As the two objects rotate about each other, energy is radiated away in the form of GWs primarily due to the mass quadrupole moment of the binary system [48]. Over the course of millions or even billions of years [79] , the two objects will inspiral towards each other [80]. As

the orbital separation decreases, the objects will move faster and increasingly radiate away more and more energy in the form GWs. This is known as the "inspiral" phase where the frequency and amplitude of the GW waveform increases as a "chirp-like" signal. During the "merger" phase, the objects plunge towards each other releasing a tremendous amount of energy, producing peak luminosities equivalent to $\mathcal{O}(10^{56}\text{erg s}^{-1})$, making CBC signals some of the most luminous events in the universe [5, 7]. If the objects are BBHs, they will merge and coalesce into a single perturbed BH which emits GWs at a set of frequencies which is parameterised by the remnant single black hole total mass and spin angular momentum [81, 82, 83] (also known as the "ringdown" phase of the GW signal). When BNSs collide, they are thought to generate short GRBs followed by a kilonovae [10]. Importantly, both the short GRB and kilonovae components may be measured by other EM telescopes across the spectrum. Theoretically, NSBH events should also be able to produce EM radiation, but this is largely dependent on several factors including the mass ratio of the binary system, BH spin and NS radius [84].

The frequency near which the two objects will merge (and the point at which the frequency of the GW waveform stops increasing) is related to a quantity known as the innermost stable circular orbit R_{ISCO}. This radius is the distance between two compact objects in orbit at which their motion becomes unstable and the two objects rapidly decrease their radial distance between each other. The radius at which this occurs is defined as

$$R_{\text{ISCO}} = \frac{6GM}{c^2}, \tag{1.24}$$

where G is the gravitational constant, M is the total mass of the system, and c is the speed of light. The corresponding frequency at which the orbit becomes unstable may be estimated using Kepler's third law and can be expressed as

$$f_{\text{ISCO}} = \frac{1}{T} \lesssim \sqrt{\frac{GM}{4\pi^2 R_{\text{ISCO}}^3}} \simeq 2.2\text{kHz}\frac{M_\odot}{M}, \tag{1.25}$$

where T is the period of the orbit and M is the total mass of the system [48]. Now that we've characterised CBC signals, we will describe data analysis methods used for detecting CBC signals.

1.6.2 Compact Binary Coalescence Search Method

The output of the LVK GW detector (measured strain) is given as a time series $s(t)$

$$s(t) = h(t) + n(t). \tag{1.26}$$

where $h(t)$ is the astrophysical GW strain as a function of time and $n(t)$ is a combination of all other forms of the noise in the detector also as a function of time. If no signal is present, the measured strain of the detector is simply equivalent to the noise $s(t) = n(t)$. This is generally known as the null hypothesis H_0. We define Eq. 1.26 as the alternative hypothesis H_1. We assume that the noise is governed by a stochastic process and the joint-probability distribution of n. The noise is also assumed to good approximation to be both stationary and Gaussian, where stationary Gaussian noise has the consequence that noise samples are uncorrelated across frequency bins and that the noise in each frequency individual bin is characterised by a normal distribution [12]. In reality there are some exceptions where the detector noise may contain non-Gaussian noise transients and non-stationary noise. For those cases where there are non-Gaussian noise transients, various glitch identification tools and techniques are used to identify periods of excess noise in the detector data output $s(t)$ [12, 56, 85, 86, 87]. We note that in the context of short duration GW signals, we may assume that the detectors contain stationary noise if the performance of the detectors does not vary significantly over large time periods.

For simplicity-sake, let us first consider the case with purely Gaussian and stationary noise (we will later explain how non-Gaussian noise transients are dealt with). How do we then distinguish noise from actual GW signal? Fortunately, the problem of extracting low SNR signals from the background is not uncommon in physics and the field of statistics and may be accomplished through a technique known as matched filtering.

Matched Filtering

The primary method for detecting CBC signals is through matched filtering [88]. Matched filtering has been applied in a variety of contexts outside of GW astronomy including: radar/sonar [89] and digital communications [90]. In this subsection, I will describe how matched filtering is used

within the context of GW detection of CBC signals including deriving the optimal matched filter, explanation of additional statistical tests such as the χ^2 statistic, as well as a discussion on template bank generation and coincidence testing.

To start, we can take advantage of the fact that we generally have a good understanding of the form of $h(t)$ through a combination of analytic and numerical waveform modeling [91]. At large distances between the two compact objects of the binary system and velocities which are smaller than the speed of light $v \ll c$ (slow motion, weak-field), a post-Newtonian (PN) approximation may be used to model the GW waveform, as outlined in [71, 92]. However, at smaller separation distances, this approximation is not valid and computationally expensive numerical solutions to Einstein's field equations are then required through numerical relativity (NR) [93]. Modeling the entire waveform requires stitching together both PN and NR estimates in a semi-analytic approximate of the entire waveform. The two main approaches currently used to model the inspiral-merger-ringdown (IMR) phases of the whole signal use either the effective-one-body (EOB) formalism [94] (SEOBNR waveform family) or the Phenomenological (Phenom) framework [74, 75] (IMRPhenom waveform family). For further details and an overview of GW waveform modeling see [95].

Given that we have an accurate understanding of GW waveforms through waveform approximation techniques, we will derive an algorithm for testing the alternative hypothesis H_1 of whether or not a signal is present in the detector noise assuming that h is exactly known. We will be following the derivation found in [51]. In order to do this we must first define the likelihood ratio given by

$$\Lambda(B|A) \coloneqq \frac{P(A|B)}{P(A|\neg B)}, \tag{1.27}$$

where $\Lambda(B|A)$ is the likelihood of B given A, $P(A|B)$ is the conditional probability of A given B and $P(A|\neg B)$ is the conditional probability of A given B is not true where $P(\neg B) = 1 - P(B)$. If we want to find the likelihood of the alternative hypothesis H_1, given the measured strain output of the detector s, we can substitute this into Eq. 1.27 and rewrite as

$$\Lambda(H_1|s) = \frac{p(s|H_1)}{p(s|H_0)}, \tag{1.28}$$

where p is representative of a probability density We also define the joint probability distribution of a Gaussian $p_G(x)$ as a function of a frequency series $x = (x_1, x_2, \ldots, x_N)$ indexed by N frequency bins where each bin is composed of independent Gaussian random variables as

$$p_G(x) = \prod_{j=1}^{N} \left(\frac{1}{\sigma_j \sqrt{2\pi}} \right) e^{-\frac{1}{2\sigma_j^2} x_j^2} \tag{1.29}$$

where the mean of the jth bin of the frequency frequency series with M samples in the bin ($\mu_j = (1/M)\sum_{i=0}^{M} x_i$) is assumed to be zero since this is a Gaussian series, and σ_j is the standard deviation of the jth frequency bin distribution
($\sigma_j = \sqrt{(1/M)\sum_{i=1}^{M}(x_i - \mu_j)^2}$). Acknowledging that Eq. 1.26 may be rewritten as $n(t) = s(t) - h(t)$ under H_1 and as $n(t) = s(t)$ under the null hypothesis H_0, we can substitute Eq. 1.29 into Eq. 1.28 under the two different hypotheses H_0, H_1 and write as

$$\Lambda(H_1|s) = \frac{\prod_{j=1}^{N} \left(\frac{1}{\sigma_j \sqrt{2\pi}} \right) e^{-\frac{1}{2\sigma_j^2} |\tilde{s}_j - \tilde{h}_j|^2}}{\prod_{j=1}^{N} \left(\frac{1}{\sigma_j \sqrt{2\pi}} \right) e^{-\frac{1}{2\sigma_j^2} |\tilde{s}_j|^2}}, \tag{1.30}$$

where $|\tilde{s}_j|$ represents the modulus of the Fourier transform of timeseries s_j. Rearranging we see that the two pre-factor terms in both the numerator and denominator cancel out and we get

$$\Lambda(H_1|s) = \frac{\{\prod_{j=1}^{N} \left(\frac{1}{\sigma_j \sqrt{2\pi}} \right)\} e^{\sum_{j=1}^{N} -\frac{1}{2\sigma_j^2} |\tilde{s}_j - \tilde{h}_j|^2}}{\{\prod_{j=1}^{N} \left(\frac{1}{\sigma_j \sqrt{2\pi}} \right)\} e^{\sum_{j=1}^{N} -\frac{1}{2\sigma_j^2} |\tilde{s}_j|^2}},$$

$$= \frac{e^{\sum_{j=1}^{N} -\frac{1}{2\sigma_j^2} |\tilde{s}_j - \tilde{h}_j|^2}}{e^{\sum_{j=1}^{N} -\frac{1}{2\sigma_j^2} |\tilde{s}_j|^2}}. \tag{1.31}$$

Performing some simple algebraic manipulations we then arrive at

$$\Lambda(H_1|s) = e^{-\frac{1}{2}\sum_{j=1}^{N} \frac{|\tilde{s}_j - \tilde{h}_j|^2}{\sigma_j^2} + \frac{1}{2}\sum_{j=1}^{N} \frac{|\tilde{s}_j|^2}{\sigma_j^2}},$$

$$\ln \Lambda(H_1|s) = -\frac{1}{2} \sum_{j=1}^{N} \frac{(\tilde{s}_j^{\mathrm{re}} - \tilde{h}_j^{\mathrm{re}})^2 + (\tilde{s}_j^{\mathrm{im}} - \tilde{h}_j^{\mathrm{im}})^2 - (\tilde{s}_j^{\mathrm{re}})^2 - (\tilde{s}_j^{\mathrm{im}})^2}{\sigma_j^2}. \tag{1.32}$$

where "re" and "im" denote the real and imaginary parts of the data respectively. Further simplifying we get

$$\begin{aligned}\ln\Lambda(H_1|s) &= -\frac{1}{2}\sum_{j=1}^{N}\frac{(\tilde{h}_j^{\mathrm{re}})^2+(\tilde{h}_j^{\mathrm{im}})^2-2(\tilde{s}_j^{\mathrm{re}}\tilde{h}_j^{\mathrm{re}}+\tilde{s}_j^{\mathrm{im}}\tilde{h}_j^{\mathrm{im}})}{\sigma_j^2}\\ &= -\frac{1}{2}\sum_{j=1}^{N}\frac{|\tilde{h}_j|^2}{\sigma_j^2}-\frac{\tilde{s}_j\tilde{h}_j^*+\tilde{s}_j^*\tilde{h}_j}{\sigma_j^2}.\end{aligned} \tag{1.33}$$

As shown in [96], the variance σ_j^2 can be written as being equivalent to $S_jT/4$ where T is the total length of timeseries and S_j is the PSD for the jth frequency bin. Converting to be in terms of frequency we get that $\sigma_j^2 = S_j/4\Delta f$. Plugging this in for the variance in Eq. 1.33 we see

$$\begin{aligned}\ln\Lambda(H_1|s) &= -\frac{1}{2}\sum_{j=1}^{N}\left(\frac{4|\tilde{h}_j|^2}{S_j}-\frac{4\tilde{s}_j\tilde{h}_j^*+\tilde{s}_j^*\tilde{h}_j}{S_j}\right)\Delta f\\ &= 2\sum_{j=1}^{N}\left(\frac{\tilde{s}_j\tilde{h}_j^*+\tilde{s}_j^*\tilde{h}_j}{S_j}-\frac{|\tilde{h}_j|^2}{S_j}\right)\Delta f.\end{aligned} \tag{1.34}$$

It then follows that the matched filter SNR, ρ, may be expressed as [97]

$$\rho = \sum_{j=0}^{N}\frac{\tilde{s}_j\tilde{h}_j^*+\tilde{s}_j^*\tilde{h}_j}{S_j}\Delta f = \langle\tilde{s},\tilde{h}\rangle, \tag{1.35}$$

where the greater the value of ρ, the more likely a given GW waveform $h(t)$ is in the detector output $s(t)$

The phase of the template waveform is also maximised over, whereby each template waveform has orthogonal phase components h'_+ and $h'_\times$ (Eq. 1.6,Eq. 1.7). As shown in [6], after maximising over phase we arrive at

$$\rho^2 = \frac{\langle s,h'_+\rangle^2}{\langle h'_+,h'_+\rangle}+\frac{\langle s,h'_\times\rangle^2}{\langle h'_\times,h'_\times\rangle} = \frac{\langle s,h'_+\rangle^2+\langle s,h'_\times\rangle^2}{\langle h'_+,h'_+\rangle}, \tag{1.36}$$

where ρ^2 is the matched filter SNR and $\langle a,b\rangle$ denotes the noise weighted inner product of timeseries a and b. The matched filter SNR is the primary statistic used to determine if a specific template is a good match for a given detector output $s(t)$. If ρ^2 is above a pre-determined threshold using a given template waveform $h(t)$, at a specific time t within a given period, we

say that there is a *trigger* in the observed data at this time. There are also additional statistical tests performed, such as the the χ^2 test, which will be discussed later in this section.

Whitening

We also mention briefly here that LVK timeseries data can have information content which is broadband. Often, low frequency noise power in a given timeseries can contain so much power that it effectively "drowns out" the high frequency portion of the signal. One method for dealing with this issue is using a technique known as whitening. In whitening, we normalise the content in a given timeseries, such that the power is equal across all frequency bins in the signal. Whitening is usually applied prior to performing any kind of signal analysis, such as matched filtering, and only requires assuming a PSD and knowledge of the sampling frequency of the timeseries. For a timeseries, s, this whitening procedure is given by the following mathematical expression as

$$s_{\mathrm{w}} = \mathscr{F}^{-1}\left(\mathscr{F}(s)\sqrt{\frac{2}{S_n(f)f_{\mathrm{s}}}}\right), \tag{1.37}$$

where $\mathscr{F}$ and $\mathscr{F}^{-1}$ are the fast Fourier transform (FFT) [98] and inverse FFT respectively, f_s is the sampling frequency of the timeseries, $S_n(f)$ is the PSD and s_{w} is the whitened timeseries. After whitening, the noise content of the whitened data will have unit variance (assuming that the PSD used in the Eq. 1.37 is the correct PSD).

Template Bank Placement

Choosing how to sample GW templates from the vast parameter space which best match $h(t)$ can be challenging. This is typically done by first constructing what is known as a *template bank* (a bank of GW template waveforms). We can quantify the coverage of the template bank through an expression known as the *minimal match* MM given as [99]

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\theta}_i} \left\langle h(\boldsymbol{\theta}), h(\boldsymbol{\theta}_i) \right\rangle \geq \mathrm{MM} \tag{1.38}$$

where $h(\boldsymbol{\theta})$ denotes a given template waveform characterised by parameters $\boldsymbol{\theta}$ and $h(\boldsymbol{\theta}_i)$ is representative of a set of templates characterised by set a of parameters over a range $\theta_i, i = 1, 2, \ldots$.

A high minimal match percentage value (where maximum coverage would be 100%) for a given template bank essentially means for any given GW, the distance of that GW waveform from any existing template in the template bank should be no more than a pre-determined distance away. This distance is quantified by determining the amount of optimal SNR (Eq. 4.1) loss if that GW waveform were to lie exactly in between its nearest templates [100]. The spacing between discrete templates in the template bank is parameterised by the square of the proper distance between intrinsic template waveform parameters given by Eq. 2.14 of [100].

In practice, the template bank is constructed such that at least one template in the bank has a minimum match greater than or equal to $\sim 97\%$ for any GW. Deciding on an *MM* value can be tricky, if one chooses an *MM* value which is too low, then template bank will be coarse and GW signals may be missed. On the flip side, if we generate a bank of templates which is too finely spaced, then we run the risk of having a higher false alarm rate (where a high false alarm rate means a higher probability of falsely identifying periods of excess noise as GW signal) There is also the added trade-off of larger template banks being exceedingly computationally expensive to build. There are a multitude of techniques used to generate template banks given in [88, 100, 101, 102, 103, 104, 105, 106, 107, 108] and we refer the reader to those manuscripts for a more detailed discussion on template bank placement. Following template bank generation, we compute the matched filter SNR of templates in the template bank with observed pieces of data $s(t)$, in order to determine the best matching template waveform for $h(t)$.

The χ^2 Test

Up to now, we have assumed only Gaussian noise, but unfortunately the global network of interferometric detectors are also often affected by non-Gaussian noise artefacts. These "glitches", or noise transients, can mimic high SNR events. High SNR glitch events typically contain lots of power across a broad frequency band, where distinguishing factors which separate glitches from GW events include: glitches only appearing in one detector at a time rather than coincidentally across multiple detectors, as well as signal morphology in time, frequency or the time-frequency plane. To take this into account, an additional test for quantifying the likelihood of a candidate event originating from a real GW signal is used known as the χ^2 test. The test operates under

the principle that the time-frequency distribution of the power of the observed data $s(t)$ should be consistent with the expected power in the matched template waveform $h(t)$, as explained in greater detail here [6, 109].

Given a trigger with a corresponding best matching template waveform, the χ^2 test is constructed by first dividing up the best matching template waveform into q frequency bins, whereby each bin is defined such that they contribute an equal amount of power to the total matched filter SNR. Next, a matched filter SNR, ρ_i, is then computed for given observed data $s(t)$ and template $h(t)$ summed over all q frequency bins. The resulting statistic may be described by

$$\chi^2 = q\sum_{i=1}^{q}\left[\left(\frac{\rho_{\times}^2}{q}-\rho_{\times,i}^2\right)^2+\left(\frac{\rho_{+}^2}{q}-\rho_{+,i}^2\right)^2\right], \tag{1.39}$$

where ρ_+ and $\rho_\times$ are the SNR values of orthogonal templates $h'_+, h'_\times$. Large values of χ^2 indicate a greater likelihood of a trigger resulting from a noise transient and as such are typically downweighted if their reduced chi-squared value $\chi_r^2 = \chi^2/2q-2$ is greater than 1 in the form of a re-weighted SNR $\hat{\rho}$ given as

$$\hat{\rho} = \frac{\rho}{\left[\frac{(1+(\chi_r^2)^3)}{2}\right]^{\frac{1}{6}}}. \tag{1.40}$$

If after SNR re-weighting the match filter SNR lies below a user predefined value, the candidate trigger is discarded and not considered for further analyses [6]. The final detection statistic is the quadrature sum of the chi-squared weighted matched filter SNR across all detectors where the event was seen.

It is also important to ensure that triggers which we observe in one detector are generally consistent with what we would expect to find in other active GW detectors around the globe. If a trigger has been identified in one detector, it must also be coincident in time with other triggers in other active detectors. Coincidence is defined as triggers from multiple detectors being within the expected maximum window of time it would require for the GW to travel from one detector to another (at the speed of light). Travel time varies depending on sky location of the event (with added noise due to measurement uncertainty), where for example at a maximum it is expected that it should take a GW to travel from the LIGO Livingston detector to the LIGO Hanford

detector in $\sim$ 10ms [6]. In order to account for added noise from measurement uncertainty, the maximum allowed GW travel time between detectors is usually expanded to $\sim$ 15ms [6], also known as the coincidence window. Triggers which are within the coincidence window are then ranked by the expression

$$\hat{\rho}_c = \sqrt{\sum_{i=1}^{N_\mathrm{d}} \hat{\rho}_i{}^2}, \tag{1.41}$$

where $\hat{\rho}_i$ are the re-weighted SNRs of the coincident triggers in each detector [6], N_d is the total number of detectors, and $\hat{\rho}_c$ is our final detection statistic for a coincident event.

Timeslides

Now that we have a detection statistic in the form of $\hat{\rho}_c$, we need a method for determining the statistical significance of this statistic. This can be done by first determining the false alarm rate (FAR), where a false alarm is defined as how often the search would identify a non-astrophysical noise event with a re-weighted SNR as high (or higher) than a given candidate GW event's re-weighted SNR. An accurate FAR requires that we have an accurate estimate on the statistical properties of the background noise distribution. In order to get an accurate approximate of the noise background of the LVC detectors, we perform an exercise known as time-slides [6, 110]. Time slides involve artificially randomly shifting the time stamps of triggers from one detector by an offset which is greater than the coincidence window ($\sim$ 10ms if considering only LIGO Hanford and Livingston). We then compute $\hat{\rho}_c$ for all coincident triggers (above a pre-determined low SNR threshold) between the time-slide triggers and those from other detectors which are within the coincidence window. These post time-slide coincident triggers are unlikely to contain real GW events. It should be noted that a real GW from LIGO Livingston would no longer be coincidence with the same signal in another detector such as LIGO Hanford, but it may get matched up with some noise above threshold, so thus it is still possible a coincident trigger could contain a real GW event. Time-slide triggers are considered to be a representative approximation of the distribution of noise background events. The FAR may thus be calculated

for a given event with a measured matched filter SNR as

$$\mathrm{FAR} = \frac{N_\mathrm{b}}{T_\mathrm{b}}, \tag{1.42}$$

where N_b are the total number of noise background time-slide events found which have a coincident SNR greater than or equal to the given event measured matched-filter SNR and T_b is the total duration of the background event data. The total duration of the background data T_b may be defined by considering that we have T seconds of data from say 2 detectors which have been time-slide in steps of dt where dt is the maximum allowed coincidence window . We then get T/dt possible slides, which means we get a total simulated background time of T/dt sets equivalent to $T_\mathrm{b} = T^2/dt$.

Given $\hat{\rho}_c$ for these noise background coincident events, we can additionally calculate the false alarm probability (FAP) of a particular event resulting from non-astrophysical noise by comparing $\hat{\rho}_c$ from background events, to $\hat{\rho}_c$ from the foreground (candidate GW trigger) events. The FAP can be expressed in the form

$$\mathrm{FAP} = 1 - e^{-N_\mathrm{b}(T_0/T_\mathrm{b})}, \tag{1.43}$$

where T_0 is the search time. Eq. 1.43 is essentially the Poisson probability of getting one or more events given an expected number of events equal to $N_\mathrm{b}(T_0/T_\mathrm{b})$. Since N_b/T_b is the expected number of background events per unit time then by multiplying by the foreground time (the actual time over which the search is done) then the expected number of events in that time is obtained. For more details on determining GW candidate trigger significance, see [6, 110, 111]. We will now move on to discuss another source GW radiation, CWs.

1.6.3 Continuous Waves

CW signals are canonically associated with spinning non-axisymmetric NSs. Other more exotic sources include boson clouds, which may produce CW GW signals through boson annihilation or level transition around fast-spinning BHs [49]. Continuous GW signals from rotating NSs are

produced through the mechanism of mass quadrupole radiation. Radiation is most commonly emitted through cracking and cooling of the NS crust, internal non-axisymmetric magnetic field flows, and mountains of mass accrued on the surface from a larger companion star [49]. CW signals are incredibly long in duration and will span (or even outlast) an entire observing run. A CW waveform is fairly simple in shape (resembling that of a standard sinusoid) and the period to complete one full NS spin revolution can be on the order of $\sim 10^{-3}s$ up to $\sim 10s$ [112] [3]. The frequency of CW signals on short time scales is generally constant, however on longer time scales the frequency will shift due to a loss in angular momentum resulting from GW radiation, magnetic braking of the NS, and the shifting position of the LVC detectors with respect to the source due to Earth's rotation and orbit around the Sun [49]. Many searches have also been carried out by the LVC over the past several years for CW signals and while upper bounds have been placed on the intrinsic CW GW strain, NS elipticity and other CW parameters, there have yet to be any direct detections [113, 114]. We will now focus our attention on methods for detecting CW signals.

1.6.4 Continuous Wave Search Methods

There are estimated to be roughly $\sim 10^8 - 10^9$ NSs in our own Milky Way Galaxy [115], of which only ~ 2500 have already been observed by the EM community [49, 116] and many of these NSs may emit detectable GWs in the form of CWs (of the 2500, only the few hundred known millisecond (rapidly rotating) pulsars fall into the ground based detector sensitivity band [49, 117]). Methods for detecting CWs from NSs may be classified into 3 types: targeted, directed and all-sky. We will now briefly summarise each of these approaches.

Targeted Search

One method for detecting CW signals is to go after these already known $O(100)$s NSs and perform targeted searches. The GW strain given by a non-axisymmetric spinning NS is typically defined by [49]

$$h_0 = \frac{16\pi^2 G}{c^4}\frac{If^2}{d}\varepsilon, \tag{1.44}$$

[3] For the quadrupole GW mode, the CW waveform frequency is twice the NS spin frequency.

where d is the distance to the NS, I is the moment of inertia with respect to the rotation axis of the NS, f is the GW frequency (equivalent to twice the spin frequency for quadrupolar emission, $\approx 4/3$ of the spin frequency for r-mode emission, and has a component equal to the spin frequency if it is wobbling (precessing)) and ε is the elipticity defined as $(I_1 - I_2)/I$ where I_1 and I_2 are the moments of inertia of the star with respect to the principal axis orthogonal to the rotation axis [118]. In a targeted search, observations from EM observers, which provide estimates on the sky position, frequency, spin, are used as input to CW GW searches in order to search for unknown parameters $(h_0, \phi_0, \psi, \cos \iota)$. Where $\cos \iota$ is the cosine of the angle between the NS source's rotation axis and the line-of-sight of the detector to the source, ϕ_0 is the signal phase offset and ψ is the polarization angle. There are many methods for performing a targeted search, such as the use of data reduction techniques (time-domain heterodyne) in combination with Bayesian inference to produce posteriors on unknown parameters [119]. There is also matched filtering, specifically the $\mathscr{F}$ statistic, which analytically maximises the log likelihood ratio of a signal+noise (combined noise-free GW signal and detector noise) model over the 4 unknown parameters previous listed [120]. Then finally there is Fourier domain analysis using the "5-vector" method, as outlined in detail here [121]. For further discussions on each technique listed above, see [119, 120, 121].

All-Sky and Directed Searches

In contrast to both a targeted and a directed search, all-sky searches impose the least amount of constraints on the observable parameter space, with a lower computational cost. Specifically, it's a search for a well modelled signal with unknown frequency, frequency derivative(s), unknown sky location and potentially unknown orbital parameters[4]. Generally speaking, an all-sky search is performed by first breaking up observational time series data into many smaller time segments, or different frequency bands using the sideband method [122]. These time segments are then analysed coherently, after which the results for each time segment may be recombined in an incoherent manner. This is otherwise known as a semi-coherent search.

In order to combine results from coherent segments incoherently, there are many methods

[4] if in a binary system

which have been developed over the past several years. Such methods include: time-domain $\mathscr{F}$-statistic, frequency-Hough, Viterbi, Powerflux and cross-corr. For a full description of these and other methods for combining coherent segments, I refer the reader to [123].

We also briefly mention a second method, a directed search, where it is only assumed that the sky location is well known and that the rotational frequency and other parameters are not known. Examples of such searches include: GW searches in the core of our own galaxy [124] and Scorpius X-1 [125]. Although not too dissimilar from an all-sky search, directed searches are useful when we have a particular source in mind and would like to tune our search to that source [126]. It should also be noted that directed searches, since they do not have to search over the sky, may thus lengthen their coherent segments to gain more sensitivity, which is done at a greater computational cost. Directed searches are also less sensitive than targeted searches due to the fact that they are mostly semi-coherent which is intrinsically less sensitive and that by searching over many templates, they also incur a trials factor which increases the FAR [127] We will now discuss another GW source type, burst signals.

1.6.5 Burst Signals

A burst GW signal is produced by sources which are either unknown or known, but difficult to model due to complicated physics. Unknown sources produced by as-yet understood astrophysical mechanisms have the potential to produce transient bursts of excess power in the LVK detectors which may be detectable. Known, difficult to model burst signals are typically short in duration (less than a second) can result from core-collapse supernovae which may emit GWs via a mass-energy quadrupole moment at possible frequencies of $\sim 200-\gtrsim 1000$Hz [128] Other potential known sources include pulsar "glitches" from a NS rapidly increasing and then exponentially decreasing its spin due to surface mountain distortions [129] and soft gamma-ray flares from brief (~ 0.1s) bursts of soft gamma-rays with possible sources such as: Magnetars [130] or quake stars [131]. Due to the fact that GWs are likely emitted from deep inside the core of a star going supernovae and are not heavily influenced by extraneous material between the source and the detector, there is the potential for much insight to be gained on the physics which govern the dynamics inside collapse such as the equation of state of hot nuclear matter inside the star [50].

In the next subsection we will discuss methods used to detect both known and unknown burst signals.

1.6.6 Burst Search Method

Burst-like signals are typically not modeled due to the complicated nature of the event and the possibility of detecting as yet unknown signals. Since the search is unmodeled, template waveforms which are exactly described by a deep knowledge of numerical relativity, post-Newtonian dynamics or GR aren't necessarily employed. As such, burst searches use methods which can detect a wide range of possible waveform types and may be classified into two distinct categories: coincident and coherent searches. Coincident searches identify clusters of times of excess power (sometimes represented through wavelet transformations) in individual detectors. After times of excess power have been individually identified in each detector, coincidence between times across multiple detectors is checked [132]. Coherent searches such as [133, 134, 135], are fundamentally different from coincident searches in that detector responses are first summed together into a single combined piece of data. Burst events are then identified in the combined data through a coherent statistic derived from the likelihood ratio functional shown in [136]. Coherent methods have the added advantage of not being limited by the least sensitive detector in the analysis, the generation of other useful coherent statistics as a byproduct of the analysis and the ability to construct the source coordinates of the GW waveforms [134], but coherent analyses are slow because they have to essentially search over the entire sky. We end this section by explaining another source of GWs, stochastic GWs.

1.6.7 Stochastic Gravitational Waves

There exists a background noise of random (stochastic) GW events which is detectable and may be classified into two categories (cosmological and astrophysical) which are broadly defined by their different amplitudes and spectral properties. The cosmological stochastic GW background is produced by a number of mechanisms including: density perturbations resulting from the amplification of vacuum fluctuations during the inflationary period, and cosmic strings resulting

from phase transitions in the early universe [137, 138]. If detected, GWs from the cosmological stochastic background could provide key insights into fundamental physical mechanisms and processes of the early universe [139]. The astrophysical stochastic GW background may be described as the random supposition of many GWs from a wide range of signals (core-collapse supernovae, compact binary inspirals, isolated neutron stars) which are weak in SNR, independent and unresolvable [140]. If detected, the astrophysical GW background would provide us with a further understanding on early astrophysical source population properties, as well as surce population formation mechanisms [140]. In subsection. 1.6.8, we move on to briefly discuss methods for detecting stochastic GW backgrounds.

1.6.8 Stochastic Search Method

The stochastic search method largely involves attempting to separate GW stochastic noise from the detector environmental/non-astrophysical noise. The noise background associated with stochastic GWs may be characterised by its energy density per unit logarithmic frequency and is very similar to the the detector instrumental noise (i.e. may be approximately described by a non-white Gaussian-normal distribution [50]). Assuming an isotropic stochastic GW background, the noise induces a strain spectral noise density on the detector. This strain would be detectable using a single detector only if it were significantly stronger than the detector noise PSD [50]. The noise is typically cross-correlated between more than one detector in order to search for a correlated noise components and one can obtain improved sensitivities using multiple detectors [137, 141][5]. This may be computed using a multitude of techniques discussed in further detail here [50, 137, 140, 141].

We also briefly mention that there are methods, other than through laser interferometers, which stochastic GWs may be detected. Given that the arrival times of pulses from Millisecond pulsars are so incredibly stable over large time scales [142], one can also perform cross-correlation analysis between pulses from multiple pulsars. Through cross-correlation, one may then be able to to distinguish between intrinsic pulse variability and pulse variability associated with the stochastic GW background. This method is known as pulsar timing and is discussed

[5]This is performed under the assumption that instrumental noise is not correlated between multiple detectors

in more detail here [142, 143]. In the following chapter, we will move from discussing GW sources and their detectability, to inferring the parameters which characterise GW sources, using a technique known as Bayesian inference.

1.7 Bayesian Inference

It is not only important that we detect a GW event, but also vital that we infer the underlying properties of that event in the form of its source parameters (i.e. component mass, distance, sky location, etc.). In LVK, the tried-and-true method for inferring source parameters is done through Bayesian inference, which is derived from Bayes theorem [144]. Bayes theorem was first proposed by Reverend Thomas Bayes in the 18th century and in it he formulated a new paradigm for thinking about the laws of conditional probability.

Bayesian probability, is a fundamentally different way of interpreting statistics from the more traditional frequentist approach. For a frequentist, an unknown parameter of interest θ is often considered to be a fixed quantity. A frequentist would determine the value of θ through sampling of observational data to form a distribution. From this distribution, a frequentist would then be able to determine confidence intervals on their estimate of θ. For example a confidence interval of 95% is stating that the true value of θ (for say 100 observations) would lie within the confidence interval in $95/100$ repeat observations. The other 5 repeat observations are not guaranteed to be close to the confidence interval and may take on any value.

On the other hand, a Bayesian does not consider the unknown parameter to be a fixed value. Rather, the Bayesian considers the unknown parameter to be a random variable which is described by a probability distribution with credibility intervals. A 95% credibility interval, corresponds to a 95% probability that the true value of parameter θ lies within the interval, given a single piece of observational data $\boldsymbol{d} = d_1, d_2, \ldots d_n$. Parameters $\boldsymbol{\theta} = \theta_1, \theta_2, \ldots \theta_n$ may then be inferred through direct application of Bayes theorem. In the following section will describe Bayes theorem in detail and refer the reader to [145] for further discussions on frequentist inference.

To describe succinctly, Bayes theorem states that one can infer the distribution of an unknown parameter, the posterior, by computing the likelihood of a given observation, scaled by

our prior belief on the distribution of that unknown parameter. To put it in the context of GW astronomy, given observed detector data and some prior assumptions about the source parameters of a GW signal, the posterior can be described by the source parameter values of that signal while also taking into account the uncertainty added by the signal being buried in noise. The posterior may be expressed as $p(\boldsymbol{\theta}|\boldsymbol{d},I)$, where $p(\boldsymbol{\theta}|\boldsymbol{d},I)$ is the probability density of the source parameters of the signal ($\boldsymbol{\theta}$ being a continuous variable), given some observed data and all other assumed relevant information I. The integral over the total posterior is normalised such that

$$\int d\boldsymbol{\theta}\, p(\boldsymbol{\theta}|\boldsymbol{d},I) = 1. \tag{1.45}$$

According to Bayes theorem, we can write the posterior as

$$p(\boldsymbol{\theta}|\boldsymbol{d},I) = \frac{p(\boldsymbol{d}|\boldsymbol{\theta},I)p(\boldsymbol{\theta}|I)}{p(\boldsymbol{d}|I)}. \tag{1.46}$$

Where each term in Eq. 1.46 can be described as

- $p(\boldsymbol{d}|\boldsymbol{\theta},I)$: The probability of the data $\boldsymbol{d}$ given the source parameters $\boldsymbol{\theta}$ and information I, also known as the likelihood of the source parameters.
- $p(\boldsymbol{\theta}|I)$: Our prior belief on the distribution of source parameters $\boldsymbol{\theta}$ given information I.
- $p(\boldsymbol{d}|I)$: A normalisation factor called the Bayesian evidence which is obtained by integrating the likelihood times the prior over all possible parameters $\boldsymbol{\theta}$ given information I.

The prior $p(\boldsymbol{\theta}|I)$ is largely informed by our understanding on the formation channels of GW sources and our current knowledge on the general physics which govern events. For example, one would intuitively think that the distance of an object in 3-dimensions should always be positive and governed by a Euclidian distance, so will set the priors such that the distance of a GW source with respect to the detectors must always lie between two positive values and also be in the form of a Euclidian distance However, if we aren't as knowledgeable about a particular parameter $\boldsymbol{\theta}$, we might try choosing a relatively uninformative prior. Choice of prior can also be

incredibly influential on the posterior shape for some GW parameters [146], but in general both the prior and the likelihood have equal weight in terms of how they influence the posterior.

The way in which the likelihood $p(\boldsymbol{d}|\boldsymbol{\theta},I)$ is defined is characterised by the noise distribution model. For GW astronomy, we typically define a likelihood which assumes that the detectors operate under Gaussian noise-like conditions. The Gaussian-noise likelihood function may be written as

$$p(\boldsymbol{d}|\boldsymbol{\theta},I) = \prod_i \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{1}{2}\frac{(d_i-\mu(\theta)_i)^2}{\sigma_i^2}\right), \tag{1.47}$$

where i is the frequency bin index, d_i is the observed data and $\mu(\boldsymbol{\theta})_i$ is a the signal model GW waveform parameterised by source parameters $\boldsymbol{\theta}$. The parameter σ_i^2 is proportional to the noise PSD defined earlier in Eq. 1.10, since the amplitude spectral density (ASD) is given as being the square root of the PSD. We note that the likelihood function in Eq. 1.47 for GW analysis is calculated on the frequency domain representation of the data. It is only in the frequency domain whereby the probabilities from each frequency bin can be multiplied since the data is independent.

The evidence $p(\boldsymbol{d}|I)$ can be defined as

$$p(\boldsymbol{d}|I) = \int p(\boldsymbol{d}|\boldsymbol{\theta},I)p(\boldsymbol{\theta}|I)d\boldsymbol{\theta}. \tag{1.48}$$

The evidence is usually also referred to as the marginal likelihood. It is often used in order to perform model selection where the evidence is required in order to calculate a quantity known as the Bayes factor. The Bayes factor is a quantifiable method for which computing the likelihood of one model versus another. The Bayes factor is defined as the ratio of evidence for two different models/hypothesis. For example, as shown in [147] one could investigate the likelihood for a model which assumes a signal+noise model $p(\boldsymbol{d}|I)_s$ versus a noise-alone model $p(\boldsymbol{d}|I)_n$. The Bayes factor for such an investigation may be written as the ratio of one evidence assuming the signal+noise hypothesis over another evidence assuming the noise-alone hypothesis expressed as

$$B_n^s = \frac{p(\boldsymbol{d}|I)_s}{p(\boldsymbol{d}|I)_n}, \tag{1.49}$$

where the noise-alone evidence integral $p(\boldsymbol{d}|I)_n$ may be written as

$$p(\boldsymbol{d}|I)_n = \prod_i \frac{1}{2\pi\sigma_i^2} \exp\left(\frac{1}{2}\frac{|d_i^2|}{\sigma_i^2}\right), \tag{1.50}$$

where again the data is considered to be in the frequency domain. If one uses a more formal definition, determining the preference of one model A over another B given observed data $\boldsymbol{d}$ is actually determined through a term known as the odds ratio O_B^A. The odds ratio is a combination of the Bayes factor and the prior odds ratio given as

$$O_B^A = \frac{p(\boldsymbol{d}|I)_A}{p(\boldsymbol{d}|I)_B}\frac{\pi_A}{\pi_B}, \tag{1.51}$$

where π_A, π_B are the prior beliefs on hypothesis A and B respectively.

If we are purely interested in parameter estimation and since we are marginalising over all parameters $\boldsymbol{\theta}$ in Eq. 1.48, we can state that the evidence is independent of $\boldsymbol{\theta}$. Since the evidence is independent of $\boldsymbol{\theta}$ and it is prohibitively expensive to compute this factor[6] (because we are integrating over the whole parameter space of $\boldsymbol{\theta}$), most Bayesian practitioners purely interested in performing parameter estimation will ignore Eq. 1.48 (and rewrite Bayes theorem in the simpler form

$$p(\boldsymbol{\theta}|\boldsymbol{d},I) \propto p(\boldsymbol{d}|\boldsymbol{\theta},I)p(\boldsymbol{\theta}|I). \tag{1.52}$$

Given that sampling from the posterior distribution can become prohibitively computationally expensive as the number of dimensions $\boldsymbol{\theta}$ is increased [147], there is prime motivation for the use of efficient methods for sampling from the posterior. In the following subsections I will describe the basic principle of two popular methods for sampling from the posterior $p(\boldsymbol{\theta}|\boldsymbol{d},I)$: Markov Chain Monte Carlo (MCMC) and Nested Sampling.

[6] We note that in nested sampling, the main purpose of the algorithm is in fact to compute this quantity in a computationally feasible manner. See Sec. 1.7.2 for further details.

1.7.1 Markov Chain Monte Carlo

MCMC is used when we would like to try and sample from some distribution, for example the posterior $p(\boldsymbol{\theta}|\boldsymbol{d})$, or approximate the expectation value $E(f)$ of some function $f(\boldsymbol{\theta})$ which is of a high dimension/complexity Where the posterior is so complex that trying to sample from the posterior through traditional means would be prohibitively expensive. Other methods for sampling from complicated distributions such as, *importance sampling* and *random sampling* may also be used, but are outside of the scope of this thesis and I refer the interested reader to [148] for a more through discussion on both techniques.

The Monte Carlo portion of MCMC refers to a technique known as Monte Carlo sampling [149]. Monte Carlo sampling means to randomly sample from some distribution. For example, we could choose to randomly sample from a normal distribution $\mathcal{N}(0,1)$ or from uniform distribution $U(0,1)$ between 0 and 1. We would then define the normal or the uniform distribution that we're sampling from the proposal distribution. If we randomly sample from the proposal distribution enough times a histogram of the resulting samples should resemble that of the original proposal distribution.

A Markov Chain [150] is a sequence of numbers whereby each number in the sequence is only dependent on the previous number in the sequence. For example, if we again decided to randomly sample from proposal distribution $\mathcal{N}(0,1)$, but instead after each sample is drawn we change the mean of the proposal distribution to be equal to that of the previous sample $\mathcal{N}(\theta_{i-1},1)$, we would end up with something known as a random walk.

In order to construct an algorithm which is able to generate n samples from the posterior in steps indexed by i, we want to ensure that two criterion are met: stationarity (i.e. in the limit where $n \to \infty$, random numbers drawn from the sequence vary a significant amount) and a final set of samples which is equivalent to the posterior $p(\boldsymbol{\theta}|\boldsymbol{d},I)$ [148]. One method for ensuring both criteria is the Metropolis-Hastings (MH) algorithm [151]. We will now go on to explain how the MH algorithm works and have also provided a pseudo code of the algorithm (Alg. 1) for reference.

The MH algorithm begins by first drawing a random point, θ_c, (also sometimes known as a *walker*) (always from the prior space, $p(\boldsymbol{\theta}|I)$) and calculating the posterior probability density

Algorithm 1 A simple MH MCMC algorithm.

Generate random initial starting point θ_c;
for $i = 1$ to n **do**;
  $\theta_p = Q(\theta_p|\theta_c)$,
  $\alpha = \frac{p(\theta_p|d,I)}{p(\theta_c|d,I)} \frac{Q(\theta_c|\theta_p)}{Q(\theta_p|\theta_c)}$
  if $\alpha > U(0,1)$ **then**
    $\theta_c = \theta_p$
  end if
end for
return $\{\theta_c^1, \theta_c^2, \ldots \theta_c^n\}$.

function (PDF) at that point $p(\theta_c|d,I)$. We also define a simple proposal distribution Q which is dependent on the current state θ_c and from which we will sample from over n steps of the algorithm (Q is usually described by a Multi-variate Gaussian distribution whose mean is equivalent to the current state θ_c). Q is chosen to be a simple distribution since it is easy to sample from at each step, as opposed to the complex posterior distribution.

The MH algorithm then calculates the posterior PDF, $p(\theta_p|d,I)$ using the likelihood and the prior, given a new proposed sample θ_p drawn from the proposal distribution $Q(\theta_p|\theta_c)$ evaluated at current state θ_c. The posterior PDF value of the current position $p(\theta_c|d,I)$ is also evaluated with current state sample θ_c drawn from proposal distribution $Q(\theta_c|\theta_p)$ evaluated at θ_p. The ratio of the two posterior PDF values scaled by the ratio of $Q(\theta_c|\theta_p)$ and $Q(\theta_p|\theta_c)$ is then computed

$$\frac{p(\theta_p|d,I)}{p(\theta_c|d,I)} \frac{Q(\theta_c|\theta_p)}{Q(\theta_p|\theta_c)},$$

where both $p(\theta_p|d,I)$ and $p(\theta_c|d,I)$ are estimated through evaluating the prior distribution and the likelihood function at θ_p and θ_c respectively. The ratio is derived from the statistical concept of "detailed balance" which states that probability is conserved from one position to another and the full derivation is given in [148]. If this ratio is greater than 1, then we always accept the new proposed sample as the new current state. If however, the ratio is less than 1, we will not necessarily reject the new proposed sample. Instead, we determine an acceptance probability based on the ratio. The acceptance probability is determined by first drawing a uniform random number between 0 and 1. We accept the new proposed sample as the new current state if the ratio value is greater than the randomly drawn number. Otherwise, the current state remains at

position θ_c. As shown in [148], the probability of acceptance (α) of a new sample in the Markov Chain can be expressed as

$$\alpha = \begin{cases} \frac{p(\theta_p|d,I)}{p(\theta_c|d,I)} \frac{Q(\theta_c|\theta_p)}{Q(\theta_p|\theta_c)} & \text{if } p(\theta_p|d,I) < p(\theta_c|d,I) \\ 1 & \text{if } p(\theta_p|d,I) \geq p(\theta_c|d,I). \end{cases} \tag{1.53}$$

There are a few downsides to using the Metropolis Hastings algorithm. One of those downsides being that we have to choose a starting point for the random walk, which is initially liable to be far from the true posterior. This could be problematic if the posterior is composed of a small concentration(s) of probability with a large amount of likelihood concentrated in a small region of the parameter space. It may also take a large number of iterations for the algorithm to walk towards areas of high likelihood contained within a small region of the parameter space. Commonly, a number of samples at the beginning of the walk are discarded such that the remaining represent a point after which the algorithm has reached a stable equilibrium. The interval over which we choose to throw away samples is called the burn-in period. Another issue relates to something known as autocorrelation. Samples $\boldsymbol{\theta}$ generated through the Markov Chains in the Metropolis Hastings algorithm may be statistically correlated with each other and are thus not necessarily fully representative of the posterior [147], since correlation implies that the samples drawn are not statistically independent from each other (i.e. it violates the Markovian principle outlined earlier). We can mitigate such correlations through a process known as thinning. Thinning involves generating a large amount of samples from the proposal distribution, but only keeping every N^{th} sample from that large sample set, described in further detail here [152].

Expanding on from traditional MCMC methods, the MCMC algorithms used in this thesis (`emcee` [153], and `ptemcee` [154]) both apply their own additional methodologies and tweaks to the original algorithm. For example, in `emcee` there is an operation applied known as the "stretch move". The stretch move by be described by first considering an ensemble of walkers being evolved simultaneously over n steps. The stretch move involves proposing a new state for the kth walker, θ_p^k, which is dependent on the current state of another randomly selected walker, θ_c^j, from the ensemble. This is opposed to the traditional method where new proposed states are

entirely dependent on the previous state of that same chain. The new state of the kth walker is then given by

$$\theta_p^k = \theta_c^j + Z[\theta_c^k - \theta_c^j] \tag{1.54}$$

where Z is a randomly selected variable from a proposal distribution parameterised by $\theta_c^k - \theta_c^j$. This process is then repeated in a series over all walkers. The stretch move may also then be parallelised by splitting the walkers into two separate sets and updating walkers from one set with walkers from another. The methodology employed by `emcee` can have superior performance over other traditional methods in the form of shorter autocorrelation times (i.e. more statistically independent samples) when run on several complex distributions [153].

In `ptemcee`, multiple Markov chains are run in parallel at varying *temperatures*. Temperatures refer to different tempered versions of the posterior given by

$$p(\boldsymbol{\theta}|\boldsymbol{d})_{\mathrm{T}} \propto p(\boldsymbol{d}|\boldsymbol{\theta})^{1/T} p(\boldsymbol{\theta}), \tag{1.55}$$

where T is the temperature value. Temperatures are assigned to all Markov chains in a geometrically spaced ladder from 1 up to a T_{max}, where the maximum temperature, T_{max}, and the number of temperatures is pre-determined by the user. Temperature values for each chain are also periodically swapped with other adjacent chains according to an acceptance ratio defined in Eq. 2 of [154]. The advantage of using temperatures is that the likelihood $p(\boldsymbol{d}|\boldsymbol{\theta})^{1/T}$ gets flattened out at high T, thus making the distribution easier to sample from. `ptemcee` is especially useful when one wants to find the global maximum and multiple modes of a complex posterior distribution [155]. For further details which are outside the scope of this thesis, see [154, 155]. We will now discuss an alternative sampling method to MCMC, nested sampling.

1.7.2 Nested Sampling

Nested sampling is another method which can be used in order to sample from the posterior. However, that wasn't the primary goal in mind when the method was first developed [156, 157] and the posterior is only really obtained as a byproduct (unlike MCMC which *directly*

samples from the posterior). An additional motivation for nested sampling relates to the fact that MCMC can have issues when trying to deal with widely spaced multi-modal and degenerate distributions, where a degenerate distribution in this case is defined as having posterior shapes which are very narrow. Nested sampling was first introduced by Skilling in 2004 [156] (later expanded upon in 2006 [157]) because he wanted a tractable method for computing the Bayesian evidence (sometimes known as the marginal likelihood) in order to compare different models effectively (See Bayes factor in Sec. 1.7). In nested sampling, samples from the posterior can be obtained as a byproduct following the evaluation of the evidence. One would be forgiven for naively assuming that it would be easy to compute this integral by simply evaluating the integrand for many values of $\boldsymbol{\theta}$ and then numerically integrating over the $\boldsymbol{\theta}$ space. Unfortunately, this procedure can get computationally expensive quickly as the number of inferred parameters $\boldsymbol{\theta}$ increases. Rather than integrating over the whole space of parameters $\boldsymbol{\theta}$ explicitly, it would be advantageous to redefine Eq. 1.48 such that it was only dependent on a single parameter and an approximate method for computing the integral could possibly be found. The fundamental idea that a complex high-dimensional problem with many inferred parameters may be represented in a simplified 1-dimensional form is one of the key ideas of nested sampling. This then begs the question, how do we convert to a simplified 1D form and compute the evidence from this 1D form? I will now describe the details of such an algorithm.

In order to convert the problem to a simpler 1D form, Skilling starts by defining the total prior mass X, stated as the total amount of prior contained within a given likelihood contour λ expressed as

$$X(\lambda) = \int_{p(\boldsymbol{d}|\boldsymbol{\theta},I)>\lambda} p(\boldsymbol{\theta}|I)d\boldsymbol{\theta}. \tag{1.56}$$

Rearranging Eq. 1.56 through an inversion of the equation we arrive at a new definition of the evidence integral

$$p(\boldsymbol{d}|I) = \int_0^1 p(X)dX, \tag{1.57}$$

where $dX = p(\boldsymbol{\theta}|I)d\theta$ and $p(X)$ is the likelihood evaluated at $p(\boldsymbol{d}|\boldsymbol{\theta}I) > \lambda$. It can be clearly seen from Eq. 1.57 that that the evidence integral is now a function of a single parameter X, the prior mass. The integral in Eq. 1.57 has limits from 0 to 1 since the prior mass decreases

from 1 to 0 as the likelihood contour λ increases. The reasoning for these limits is most clearly illustrated in Fig. 2 of [157] where at low likelihood values (i.e. broad likelihood contour and low λ), we see total contained prior mass as close to 1, and at increasing likelihood values the prior mass approaches values of 0.

In order to compute the integral of Eq. 1.57, Skilling applies the trapezoid rule to Eq. 1.57 and expresses Eq. 1.57 as a weighted summation defined as

$$p(\boldsymbol{d}|I) = \sum_{i}^{M} \omega_i p(X_i), \tag{1.58}$$

where $\omega_i = \frac{1}{2}(X_{i-1} - X_{i+1})$. Since $p(X)$ is not typically well-known, the nested sampling algorithm instead approximates $p(X)$ by drawing samples from the constrained prior mass X in M slices, where $0 < X_M < ... < X_1 < X_0 = 1$. As outlined in Alg. 2, the algorithm begins by first generating an initial set of N_{live} "live points" initially drawn from the prior $p(\boldsymbol{\theta})$ (usually on the order of ~ 1000). At each prior mass volume slice X_i, the likelihood values for each live point is calculated and the live point with the lowest likelihood value identified which gives us our approximate $p(X_i)$. The parameters associated with the minimum likelihood live point, θ_i, are then saved for later use in constructing the posterior (described shortly). The minimum likelihood live point is then discarded and replaced with a new live point from the prior $p(\boldsymbol{\theta}|I)$ such that the likelihood value of the new live point is greater than the likelihood value of the old minimum live point from the previous step i. There are several methods for generating a new live point from the prior under the given likelihood constraints, which will also be outlined shortly.

As shown in [157], the prior mass at each step X_i is equivalent to $X_i = t_i X_{i-1}$, where t_i is a probability distribution defined on the bounds $U(0,1)$. The probability distribution associated with taking the largest of N_{live} numbers from a uniform distribution may be expressed as

$$p(t_i) = N_{\text{live}} t_i^{N_{\text{live}}-1}. \tag{1.59}$$

Taking the expectation value of the distribution $E[\log p(t_i)]$, it is shown in [157] that the log prior

Algorithm 2 A simple nested sampling algorithm. A set of live points, $N_{\rm live}$, are first initialised. For each step, i, over M iterations, the minimum likelihood value for all current live points is determined, $p_i(\boldsymbol{d}|\boldsymbol{\theta},I)$. A weight, ω_i, is then calculated which is parameterised by the constrained prior volume, X_i. The likelihood of the minimum live point is then multiplied by the weight and added to the running evidence value, $p(\boldsymbol{d}|I)$. Both the minimum live point values $\boldsymbol{\theta}_i$ and weight ω_i are saved and the minimum likelihood live point is replaced with a new live point which is sampled from the constrained prior such that the new point likelihood is greater than the removed live point's. Sampling concludes once a user pre-defined stopping criterion is met.

set Generate $N_{\rm live}$ points $\theta_j,...,\theta_{N_{\rm live}}$ from the prior;
for $i = 1$ to M **do**;
  set $p_i(\boldsymbol{d}|\boldsymbol{\theta},I) = \min$ (likelihood values of active live points),
  set $X_i \approx -\frac{1}{N_{\rm live}} X_{i-1}$,
  set $\omega_i = \frac{1}{2}(X_{i-1} - X_{i+1})$,
  set $p(\boldsymbol{d}|I) = p(\boldsymbol{d}|I) + p_i(\boldsymbol{d}|\boldsymbol{\theta},I)\omega_i$,
  save live point $\boldsymbol{\theta}_i$ along with weight ω_i,
  remove $\boldsymbol{\theta}_i$,
  replace $\boldsymbol{\theta}_i$ with new live point sampled from constrained prior,
  ensure new live point likelihood $\geq$ old θ_i min likelihood,
end for
return $p(\boldsymbol{d}|I)$.

mass shrinks approximately by a factor of $\approx -1/N_{\rm live}$ for each step, thus giving us an accurate method for approximating X_i.

Unfortunately, there is no exact figure merit which guarantees that the nested sampler has converged. This is because there is always the marginal possibility that there will unexplored regions of the parameter space which may contain high likelihood in a small contour [157]. A rough approximate which most practitioners use to determine convergence is through a quantity known as the log evidence ratio. The log evidence ratio is defined as the estimated total evidence and the current accumulated evidence $p_i(\boldsymbol{d}|I)$ [158]. The estimated total evidence is a summation of the current accumulated evidence and the estimated remaining evidence. The estimated remaining evidence is approximated by identifying the current maximum likelihood value out of all current active live points $p_{\rm max}(\boldsymbol{d}|\boldsymbol{\theta},I)$ multiplied by the current enclosed prior mass X_i given by

$$p_{\rm est}(\boldsymbol{d}|I) = p_{\rm max}(\boldsymbol{d}|\boldsymbol{\theta},I)X_i. \tag{1.60}$$

A stopping criterion for the algorithm, dlog$Z$, is then defined as the ratio of the estimated total

evidence and the current accumulated evidence expressed as

$$\mathrm{dlog}Z = \log\left(\frac{p_{\mathrm{est}}(\boldsymbol{d}|I) + p_i(\boldsymbol{d}|I)}{p_i(\boldsymbol{d}|I)}\right) < \zeta, \tag{1.61}$$

where ζ is a user pre-defined stopping threshold, most nominally chosen to be ~ 0.1 [158]. The reasoning behind this definition of algorithm stopping criterion is that small changes in $p(\boldsymbol{d}|I)$ indicate that the accumulation of the evidence is tailing off, so thus the evidence is nearly fully integrated and sampling may thus be terminated.

Now that we have described how one may use nested sampling to accurately and efficiently approximate the evidence, it turns out that the posterior may also easily be sampled from as a byproduct of the nested sampling algorithm. Given that the posterior is simply the prior weighted by the likelihood and since we have already accumulated likelihood samples through Eq. 1.58 in the form of the minimum live point likelihood value over M steps, $p_i(\boldsymbol{d}|\boldsymbol{\theta},I)$ and samples from the prior through the saved parameter values of each that same minimum live point it is shown in [20, 159] that the posterior may thus be approximated as

$$p(\boldsymbol{\theta}|\boldsymbol{d},I) \approx \frac{\sum_i^M p(X_i)\omega_i\delta(\theta_i)}{p(\boldsymbol{d}|I)}, \tag{1.62}$$

where $\delta(\theta_i)$ is the Dirac delta function centered on the ith posterior sample θ_i.

The two nested sampling software packages used in this thesis (`Dynesty` [160], `CPNest` [161]) apply their own tweaks to the original nested sampling algorithm proposed in [157]. The primary difference between approaches lies in how each method decides to replace the discarded lowest likelihood live point at each step/contour i, given the constrained prior distribution at that step. In `CPNest`, this is accomplished by first randomly selecting one of the current active live points. An MCMC chain is then run from the starting point value equivalent to the randomly selected live point. The maximum length of the chain may be pre-defined by the user and varies from step to step in the `CPnest` algorithm according to the autocorrelation time scale (see Eq. 23 of [20] for the definition). Additional features are also included in `CPNest` which reduce the amount of manual tuning required for convergence and are explained in more detail here [20].

`Dynesty` by default uses a combination of MCMC chains and ellipsoids to produce independently and identically distributed posterior samples. Specifically, new live points are drawn to replace the lowest likelihood point at each step by approximating the bounds of the current prior mass X_i using ellipsoids (by default, the algorithm uses multiple ellipsoids). Ellipsoids are constructed and optimised using k-means clustering. Once a proper bound has been constructed, samples may be generated conditioned on those bounds using a multitude of methods outlined in Sec. 4.2 of [160]. By default, `Dynesty` draws a new sample from the constrained prior such that it is within the bounds defined by the ellipsoids and then evolves that sample through to an MCMC chain whose proposal distribution by default is dependent on one of the ellipsoids (selected randomly). Further details and additional tuning options available in `Dynesty` are outlined in [160]. We also briefly mention that many of the sampling methods included here are packaged into user-friendly analysis tools such as `bilby` [19] and `lalinference` [20] which are used extensively by the LVK.

1.8 Summary

The chapter opens by discussing the detections made by the LVC in the past 3 observation runs and how those detections have been used to further our understanding of cosmology, astrophysics and GW astronomy. A brief introduction to Einstein's Field Equations and how those field equations lead to the prediction that GWs exist was provided. It was shown that various sources are able to produce GWs and it was described how the LVC detectors physically operate to detect such signals. The search techniques for all source signals was described, along with a description on how predictions are generated on the underlying source parameters of GW signals. From the descriptions of the search and parameter estimation techniques given in Sec. 1.6.2 and Sec. 1.7 it was shown that while optimal under Gaussian-noise conditions, standard approaches used by the LVC are computationally expensive to run. This is especially problematic given the large number of expected signals the LVC will see in the coming years, with the additional need of alerting EM partners in low-latency due to short-lived GW EM counterparts. Given the urgent need for low-latency tools to perform both GW detection and

GW parameter estimation, we will show in the subsequent chapters (Ch. 4, Ch. 5) how recent advances in the field of ML may be applied in order to solve these problems. In the next chapter (Ch. 2), we will describe the basics of ML, as well as provide detailed descriptions of the ML algorithms used in this thesis.

Chapter 2

An Introduction to Machine Learning

In this chapter we will explain in detail the machine learning techniques used in this thesis including: fully-connected neural networks [162], CNNs [163] and CVAEs [164]. We will also describe some of the basic principals of how neural networks learn (i.e., backpropagation [165]), how they are initialized, best training practices, methods for evaluating performance, and methods for data augmentation/pre-processing. In order to dispel the notion of machine learning being uninterpretable, it should be stated that machine learning is perhaps most simply described as just function approximation. The overarching goal being to approximate a function which is obtained by finding a global minimum according to a cost function. How one defines this minimised function is the key and there are many methods for doing so.

A machine learning algorithm can perform a variety of objectives including: classification [166], regression [167], anomaly detection [168], denoising [169] and density estimation [170]. There are many other tasks a machine learning algorithm can tackle, but the two which are primarily used in this thesis are classification and density estimation. In classification, a machine learning algorithm attempts to learn the optimal function to classify a given set of inputs (e.g. a timeseries, image or any other data input) and returns as output the likelihood that the given input came from a particular class or set of classes (e.g. animal type, number). A machine learning algorithm can also alternatively perform non-linear regression by switching from outputting numeric values describing class probability to predicting a continuous variable (i.e. a changing temperature). Finally, we can even have a machine learning algorithm produce

probability distributions (e.g. GW source parameter posteriors) for a given input by enforcing that the algorithm predict parameters which fully describe a distribution (i.e. the mean and variance of a Gaussian).

2.1 Fully-Connected Deep Neural Networks

So, how do we build a machine learning algorithm? Let us first define a simple neural network architecture, the perceptron [171]. A perceptron is made up of a neuron which is composed of several tunable variables called weights and biases. Given an input sample (e.g. an image of a cat/dog), the neuron produces a prediction on a corresponding characteristic of that sample (i.e. likelihood of being an image of a cat or a dog). An illustrated diagram of a perceptron network is given in Fig. 2.1. Predictions from a perceptron could be in the form of a class or a parameter value which describes that given input sample. Multiple perceptrons can then be grouped into a layer of perceptrons (henceforth referred to as neurons) to form a network of neurons [162]. Multiple layers of neurons can then also be stacked together to form what is known as a fully-connected deep neural network (illustrated in Fig. 2.2). "Fully-connected" in this case meaning that the output from every neuron in the previous layer is given as input to every neuron in the next layer. This neural network, made up of many neurons, may then be trained to give better predictions by showing it multiple input samples (training samples). Better predictions are achieved by adjusting the tunable parameters of each neuron according to a term known as a cost function. The cost function describes how well the network is performing with respect to the true value of each input training sample [172].

For the sake of simplicity lets presume that we want to perform what is known as a binary classification task, defined as having the network predict a value which assigns the likelihood that a given input belongs to one of two classes. Lets presume that the true value for each class is in the form of a number from 0 to 1 where 0 represents one class and 1 represents the other. In order to get the neural network to make the correct predictions, weights and biases of each neuron are updated according to the performance of the output of the network with respect to the

cost function. The cost function of a neural network can take many forms, but the mean squared error is one of many well established choices for a binary classification problem and may be defined as

$$H = \frac{1}{n}\sum_{i}^{n}\sum_{j}\left(y_j^M(x^{(i)}) - y_j^{\mathrm{t}}(x^{(i)})\right)^2, \tag{2.1}$$

where y_j^{M} is the neural network prediction from the final layer M, superscript i indicates the sum over elements in a group (batch) of n training samples ($x^{(i)}$), subscript j indicates the sum over the output of neurons in the layer, and y_j^{t} is the true class label or parameter value for each training sample. When the predicted labels over all training samples in the final layer y_j^{M} are similar to the training sample true labels y_i^{t} over all training samples, the cost term approaches zero and is near a minimum (the absolute minimum, which for mean squared error is actually zero, is when all predictions are exactly correct).

The output of each neuron in every layer may then be defined recursively as a learned function expressed as

$$y_k^m = \sigma\left(\sum_j w_{jk}^m y_j^{m-1} + b_k^m\right), \tag{2.2}$$

where y_j^{m-1} is our given input from neuron j in previous layer $m-1$, w_{jk}^m is our tunable scalar weight parameter, b_k^m is our tunable scalar bias term, and σ is a non-linear function which rescales the output. Since we usually frame the binary classification problem such that a prediction of 0 represents the first class and a prediction of 1 represents the second class, we typically apply a sigmoid activation function (Fig. 2.3) since it limits the output to be between 0 and 1. Nonlinear activation functions are also applied throughout all layers of the network in order to allow our network to learn non-linear functions/features [173]. From Eq. 2.2 it can be clearly seen that each layer is dependent on the previous layer. For example, Eq. 2.2 can be used to explicitly write out a network which has 3 layers defined as

$$y_u^3 = \sigma\left(\sum_v w_{vu}^3 \sigma\left(\sum_k w_{kv}^2 \sigma\left(\sum_j w_{jk}^1 x_j + b_k^1\right) + b_v^2\right) + b_u^3\right), \tag{2.3}$$

where we see that our neural network is composed of many nested functions. It is clear that the more layers we use, the more complex our neural network approximated function (Eq. 2.3)

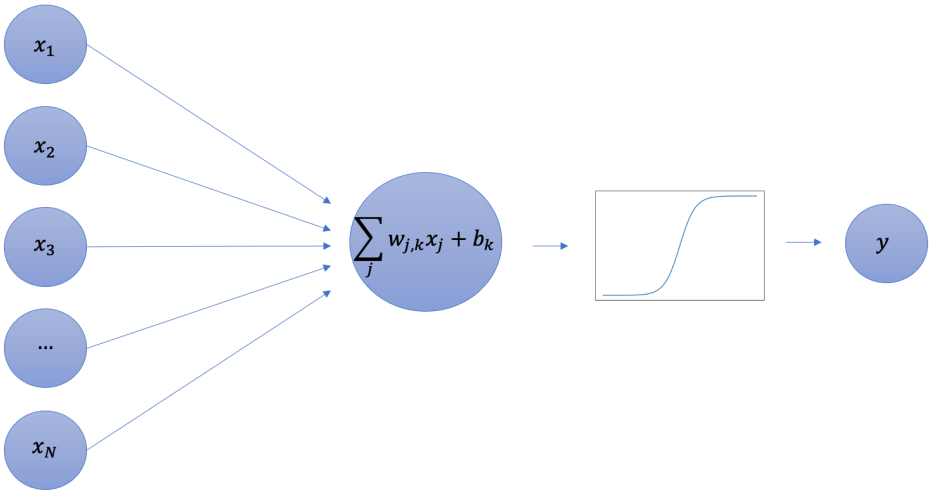


Figure 2.1: A perceptron network whose inputs $x_1, \ldots, x_N$ are each multiplied by a learned set of corresponding weights $w_1, \ldots, w_N$ and summed together. The summed value is then passed through an activation function to get the final output of the network, y.

becomes.

The reader's next question may rightly be, how does one go about choosing the right weights and biases for each neuron in the network? In other words, how much of an effect will changing the weights and biases have on the resulting cost function H. We can quantify the amount to change in weights and biases such that the cost is minimised through an operation known as gradient decent [174]. We will specifically be discussing a variant of this optimisation method called stochastic gradient decent.

Stochastic gradient decent is an iterative algorithm used to find the global minimum of a function. In general, the minimum of a function may be determined by computing where the derivative (or gradient) of that function is equal to zero. Specifically, we are interested in minimising the cost function defined in Eq. 2.1. This cost function is itself a function of the weights and biases of the neural network. The amount that each weight and bias term in the neural network needs to be adjusted in order to minimise the cost function may be defined as

$$g_1 = g_0 - \gamma \nabla H(g_0, x), \tag{2.4}$$

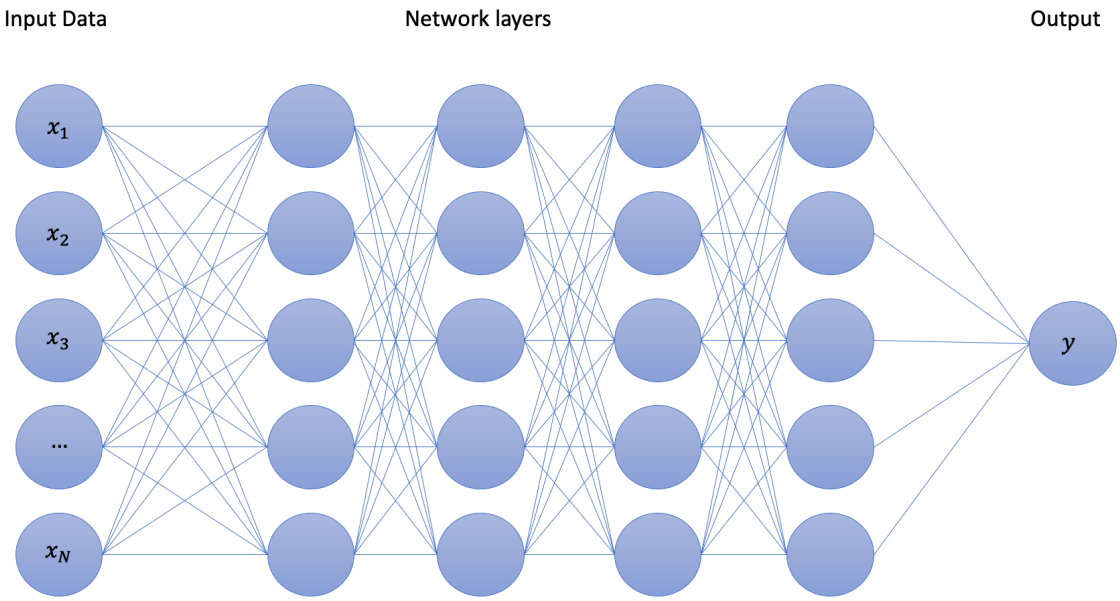


Figure 2.2: A deep fully-connected neural network. Samples $x_1 ... x_N$ are given as input to every neuron (blue unlabeled circles) in the first layer of the neural network. The outputs of each neuron are then passed through an activation function (See perceptron illustration in Fig. 2.1) on to the next set of nodes in the following layer (represented as blue lines). The outputs from the final layer of neurons pass through a single neuron which determines the class/value of the given input.

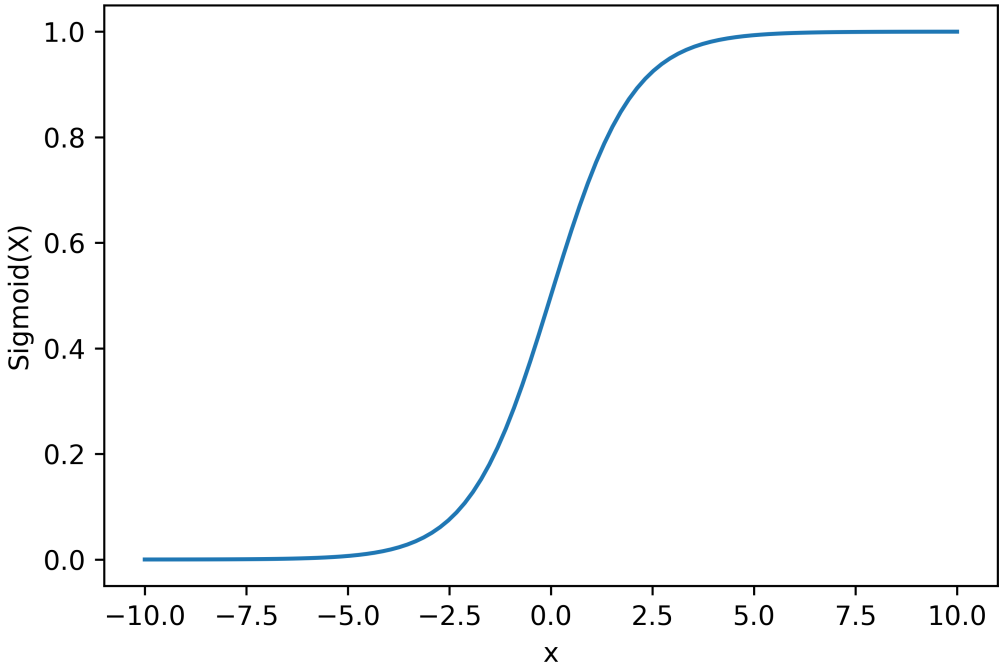


Figure 2.3: A sigmoid activation function plotted over the input range -10 to 10. Given an input, the activation function will rescale the input to be between the range of 0 to 1.

where γ is a tunable step size scale factor (learning rate), $\nabla H(g_0, x)$ is the gradient of the cost with respect to the weights and biases of the network, g_0 is the current value for all weight and bias terms, and g_1 is the updated value for all weight and bias terms. In order to compute the gradient, we must compute the partial derivative of the cost with respect to each weight and bias. For example, one can explicitly write out the partial derivative of the cost with respect to weight w_{11}^1 for our simple 3 layer network defined in Eq. 2.3 using the chain rule as

$$\frac{\partial H}{\partial w_{11}^1} = \frac{\partial H}{\partial y_1^3}\frac{\partial y_1^3}{\partial w_{11}^1} + \frac{\partial H}{\partial y_2^3}\frac{\partial y_2^3}{\partial w_{11}^1} + \ldots, \tag{2.5}$$

where we see that the derivative is a sum over the partial derivatives from each neuron in the third layer y^3. We now need to continue expanding backwards to the previous layers using the chain rule

$$\frac{\partial H}{\partial w_{11}^1} = \frac{\partial H}{\partial y_1^3}\left(\frac{\partial y_1^3}{\partial y_1^2}\frac{\partial y_1^2}{\partial w_{11}^1} + \frac{\partial y_1^3}{\partial y_2^2}\frac{\partial y_2^2}{\partial w_{11}^1} + \ldots\right) + \frac{\partial H}{\partial y_2^3}\left(\frac{\partial y_2^3}{\partial y_1^2}\frac{\partial y_1^2}{\partial w_{11}^1} + \ldots\right), \tag{2.6}$$

where it can be seen that we end up with an expanding sum of partial derivative products. This process of propagating the gradients back through the network is known as backpropagation. This is done for every weight and bias term in the network. Ideally, we would compute the gradient over more than just one training sample, but rather the whole training set during each tunable parameter update. However, computing the gradient over the whole network for all training samples is computationally expensive, thus the training set is normally split into batches of samples in order to reduce the computational cost. If we used all training samples then this would be called "gradient descent" but by using batches we include randomness, hence this is called "stochastic" gradient descent. We also note that all weight and bias values are typically initialised randomly prior to training and there are many schemes for choosing the optimal initialisation [175].

2.2 Training Best Practices (practical advice for the reader)

We were commonly told when first wading into the pool of deep learning that the practical implementation/training of deep learning models is largely a dark art. In this section, we would like to clear up the picture by offering some best practices from personal experience when training machine learning models in general.

2.2.1 Dataset Size, Pre-Processing and Augmentation

Training Data Size

When in doubt, the more training data you make available to yourself, the better [176, 177][1]. Time and again, during a large part of this thesis work, we would spend countless hours tuning and tweaking the neural network architecture only to run up against some seemingly insurmountable wall of impeded progress. Only then to increase the number of training samples by several factors and see a significant increase in performance. There is no hard rule for the exact number of training samples needed, as it is dependent on many factors associated with the problem being

[1] We note that if the user is employing an ML model which is complex with respect to the training parameter space that there is a risk of over-fitting when using a large training set.

addressed, including the structure of the data, the data space itself. There is some theoretical work using the concept of Vapnik-Chervonenkis (VC) Dimension [178] which may possibly inform the practitioner on a lower bound for the number of training samples to use (typically of order billions of training samples for a standard fully-connected network [178]). However, it should be noted that this concept is only applicable to a handful of standard algorithms and there is much disagreement in the computational science literature concerning the relevance of VC dimension in the practical implementation of machine learning algorithms [179, 180, 181]. Specifically, it is known that simple neural network models perform well using a lower number of training samples than the VC dimension would demand.

Data pre-processing

Commonly, input datasets may span a large dynamic range of several orders of magnitude. In order to help the network learn more efficiently, it is beneficial to perform some pre-procesing steps [182]. It's important to do this because if the input data has large values over a wide range, and our network is trying to learn a mapping from input to prediction, the learned weights may also end up being very large. Large learned variable weights can lead to large gradients which cause the network to update weights in large step sizes, making the whole network more unstable in the process. Normalisation can also help ensure equal importance is assigned to all features in the training set [183], i.e., greater numeric features do not dominate smaller numeric features. We can partially solve this issue by normalising our input dataset to be between the range of zero to one[2]. For example, one can normalise a series of given inputs by

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}, \tag{2.7}$$

where x' is our new normalised data, x is our original un-normalised data, x_{min} is our un-normalised data minimum value and x_{max} is our un-normalised data maximum value. It can also be advantageous to rescale our input data to a standard normal distribution if our data has

[2]There are multiple options for ranges one can use (i.e. -1 to 1).

widely varying scales by applying

$$x' = \frac{x - x_{\text{mean}}}{\sigma_x}, \tag{2.8}$$

where x_{mean} is the mean of our data and σ_x is the standard deviation of our data. In addition to rescaling our input data, we occasionally need to also rescale our training labels. This is especially relevant if the activation function in our final neural network layer is fixed to be between some values. For example, if we were using a sigmoid activation function where the values of the output of the sigmoid are only allowed to be between zero and one, then we would also want our training label values to lie between zero and one since the network would not be able to produce values outside of that range.

Data Augmentation

Neural networks can sometimes be composed of hundreds, thousands, even million of parameters to be tuned during training. Given the massive number of parameters, many complex networks also require a massive number of training samples. But, what do you do if you are only given a limited number of training samples? This is where data augmentation comes to the rescue. In order to generate more training data, we can apply simple changes to the existing limited dataset in order to greatly expand the number of training samples. Data augmentation can be accomplished in various ways, including the methods briefly mentioned here:

- Translation: Shifting the input data sample across n dimensions in the input data sample space. For example, if we had a dataset with images which were all centered in the x-y plane, we could shift each image by a random amount on either the x or y axis in order to translate it (the remaining space after translation could be filled with a constant value, or random Gaussian noise).

- Rotation: Primarily useful for 2-dimensional images, one can also rotate each image in a dataset about a fixed point in the x-y plane. All remaining space after rotation can again be filled with a constant value or random noise.

- Cropping: One can choose a random subsection of an n-dimensional input in a dataset,

remove all data outside of this subsection, rescale the cropped data to the original input size.

It should be noted that the augmentation applied has to be consistent with what we wants the network to be able to learn. For example, we wouldn't do rotation if we were identifying buildings in images because they are all usually upright. More directly relevant to GW data analysis, one can also make additional fake signals using existing data (as outlined in detail in Ch. 5, Sec. 6.4). We draw a subset of the parameters which characterise or signals and then use them to augment the existing training sample such that it is representative of those newly drawn parameters. If the training signals are buried in noise which is well-defined, new noise realisations may then be drawn and added to noise-free versions of the training signal in order generate further training signals.

2.2.2 Validation

In order to understand how well your model is performing with respect to the training data, one may plot the resultant average output of the loss function as a function of the number of training iterations. Ideally, we would like the loss to rapidly decrease and to then flatten out. If the loss flattens out, it *may* indicate that the network has reached a minimum in the loss space, much like one would find in the minimum of a parabola, thus it may be reasonable to terminate training. Although this will give us an indication of how well the network is performing on the training set, it does not inform us as to how well the model will generalise to a new testing set. In order to quantify the generalisation ability of the network, we may set aside a small portion of the training set prior to training as a validation set. During training, we can temporarily fix the weights and run the validation set through the model in order to compute a validation loss.

If we plot both the validation loss and the training loss as a function of time, we would preferably like to see both curves decrease as a function of training iteration, level out and trace each other . However, if we see that the training curve initially decreases and then continues to decrease while the validation curve initially decreases and then subsequently increases we may say that the model has overfit the training data. Overfitting the training set generally indicates

that either our model is too complicated and has essentially memorised the training set or that we do not have enough training data to sufficiently cover the entire parameter space. What is meant by "memorising the training set" is that the network is parameterised by so many parameters that it is essentially able to model each data point in the training set with a weight or bias coefficient. This is similar to if we had a polynomial composed of 3 coefficients and a data set of 3 data points, we could in principle perfectly fit the polynomial to the dataset such that there was zero loss, but obviously this means the model will not be able to generalise well to other data points from that parameter space. Overfitting can be mitigated by increasing the training set size, adding dropout connections (Sec. 2.3.1), batch normalisation (Sec. 2.3.2), early stopping of the training run if the model loss has flattened out and before it overfits to the training data [162] or reducing the complexity of the neural network [162].

2.3 Regularisation

Regularisation is used to help prevent a machine learning model from overfitting the training data. In the following subsections we will discuss various regularisation techniques which are used in this thesis.

2.3.1 Dropout

One of the easiest regularisation techniques to implement is dropout. The method is used in order to effectively simulate training and evaluating many different network architectures relatively cheaply. Dropout may be implemented across most types of neural network layers (i.e. fully-connected, convolutional filters, etc.). During training, if dropout is implemented, a randomly selected subset of the neurons in a layer will be switched off and not used when the gradient is computed and backpropogated through the network for a given training iteration. The percentage of neurons in a layer to switch off is a tunable parameter, though typically we have found through our own work it is best to set a value of no more than 50%. Due to the fact that dropout essentially forces neurons to take on more or less information from a given set of inputs variably during training, dropout may help the network to generalise better. Since a neuron in a given

layer is forced to not rely upon other neurons, which in a non-dropout network may specialise in learning a particular set of feature from the inputs and not any others [184].

2.3.2 Batch normalisation

Batch normalisation aims to standardise the inputs to a neural network at each layer of the network. As such, batch normalisation is essentially a normalisation computed on the output of each individual neuron within a neural network to have zero mean and unit variance. This normalisation in effect prevents weights or biases from becoming too large. This is formalised as

$$\hat{y} = \frac{y - \mu}{\sigma^2 + \varepsilon}, \tag{2.9}$$

where y is a given input to a network layer (i.e. output of a neuron), μ and σ^2 are the mean and variance of y computed over the batch and ε is a small constant. The epsilon parameter is used to avoid instances where the variance σ is very small and thus preventing from having large values. Rescaling of $\hat{y}$ is then applied through two learned variables (α and β)

$$y' = \alpha\hat{y} + \beta, \tag{2.10}$$

where α and β are learnable parameters, and y' is the normalised input to the next layer. If it turns out that the application of batch normalisation is not optimal, then the network may undo the above normalization in Eq. 2.9 through the optimization of (α, β) in each layer. We also note that the mean and variance (μ,σ) are also recorded during training through a running mean and variance updated at each training iteration . When going to test/validate the network, μ and σ in Eq. 2.9 use these running means/variances. For more information on other batch normalisation techniques, see [162, 185].

2.4 Hyperparameter Optimization

Given the increased complexity and time cost of training neural networks in recent years, despite the significant speed increase in graphics processing units (GPUs), choosing the optimal

hyperparameters and run settings is key to efficient resource/time management. For example, for a practitioner trying to train a standard model such as ResNet-101 [186], training the model to completion can take $O(24)$ hours and further tuning of hyperparameters to find the optimal ResNet-101 model can take several weeks [187]. Much of this work involves training the model multiple times with different settings and architectures. Hyperparameters may govern a number of model attributes, as well as model performance and may include: the number of neurons, number of layers, learning rate, activation functions, etc. In this section, we will describe three approaches that have been used during the course of this thesis work to optimise hyperparameter choices for a given model. In practical terms, the search space for each of these optimisation algorithms is composed of an n-dimensional hypercube, where each side of the cube is a range of possible values, or binary choices (i.e. turning on/off batch normalisation) for each tunable parameter.

2.4.1 The "Intuitive" Approach

The first technique which should be used to determine optimal hyperparameters is through what one might call the "intuitive" approach. In this approach, hyperparameters are manually tuned based on common-sense and experience. Start with a small number of network parameters and see if the network is performing better than random chance. Following this, slowly build up the complexity of the network by widening the layers, adding layers, iteratively increasing the training data volume, etc. This approach is in large part how different models were tested over the course of the work completed in this thesis.

2.4.2 Random Search

Probably the simplest of the three approaches, a random search determines model hyperparameters given a predefined prior distribution for each hyperparameter. The user may define whatever distribution they prefer whether that be a multivariate Gaussian distribution or a uniform distribution and may also choose the bounds of that distribution. Optimisation is carried out in a series of trials, whereby a random subset of hyperparameters is sampled from each hyperparameter dis-

tribution and the model is trained for a number of training iterations determined beforehand by the practitioner. In terms of discrete parameters, such as activation functions, one can simply randomly select an activation function from a list of activation functions for each layer (or all layers) during each trial. Since by definition a random search is sampling statistically independent hyperparameter realisations from a distribution, one gains some practical benefits from this statistical independence. One of those benefits being the fact that the optimisation process may be stopped at any time, since hyperparameters may be drawn from anywhere within the range defined by the distribution priors [188]. Further trials only enhance the coverage of the hyperparameter parameter space by the random search. Additionally, if a trial fails (i.e. say a mouse chews through your computer power cable), the trial may either be restarted or ignored without having any affect on the statistical significance on the trials already performed [188]. Through a random search, we are able to discover a set of hyperparameters which provides a fast avenue to achieve a low loss.

2.4.3 Grid Search

In a grid search approach we define both an upper and a lower limit for each hyperparameter. Additionally, we define a step size by which we increase the the hyperparameter value after each model evaluation (much like steps on a ladder). Each of these hyperparameter ladders are iterated over in an n-dimensional grid, where n is representative of n hyperparameters describing the model. Although a fine spacing in an n dimensional grid may make it likely that the practitioner is sufficiently sampling the hyperparameter space, a grid search suffers from the curse of dimensionality, whereby as the number of hyperparameters used (dimensions) increases, the number of evaluations required to perform trials across the entire grid increases geometrically [188]. Additionally, we can't necessarily stop a grid search in the middle of its process, like we can with a random search, since we would have only searched half of the parameter space at that point.

2.5 Convolutional Neural Networks

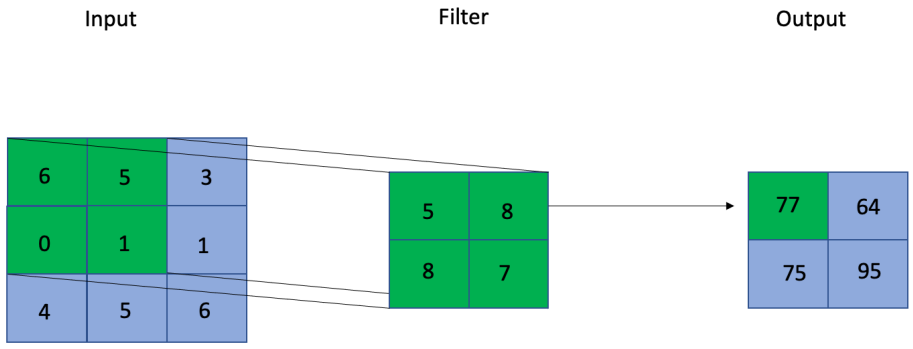
CNNs were first made popular in the late 1980s to early 1990s and one of the most famous examples is that of the LeNet architecture made by Yann LeCunn in 1998 [189]. In their paper, LeCunn illustrated that CNNs outperformed other simpler techniques like k-Nearest Neighbour [190] and fully-connected neural networks (Sec. 2.1) on a variety of tasks. In recent years CNNs have been applied to many more domains including image object detection [191], fraud identification [192], healthcare analysis [193], and many others. In this section we will explain how CNNs work, mechanisms for improving CNN performance, and why they are so powerful.

2.5.1 The Convolutional Filter

The basic building block of a CNN, convolutional filters, are analogous to neurons in a standard fully-connected neural network. A convolutional filter is typically made up of an $N \times M$ matrix, where N and M may be different sizes (for an illustrated example, see Fig. 2.4). There is also sometimes an additional dimension in the form of channels we'll denote here as D. Channels are typically other components of the input which represent different types of information. For example, the red, blue and green components of an RBG image could be considered as 3 separate channel dimensions, or the outputs from multiple GW detectors in a global network. The output for a single filter y^m_{ijk} in a given layer may be defined recursively as

$$y^m_{ijk} = \sigma\left(\sum_{a=-N/2}^{N/2}\sum_{b=-M/2}^{M/2}\sum_{c} h^m_{abck}\, y^{m-1}_{i+a,j+b,c} + b^m_{ck}\right), \tag{2.11}$$

where i and j index the x and y-axes of a 2-dimensional image and k indexes the output channel. The output from a given filter then becomes a separate channel in the output. The filter is denoted as h^m_{abck} where k denotes the k'th filter for the m'th layer and c'th input channel. The input to the filter is given as $y^{m-1}_{i+a,j+b,c}$ where a and b index the x, y input dimensions of the filter whose length is N, M in those dimensions. The summation over a, b represent the convolution of the filter sliding with respect to the indices i, j and the summation over c sums the contributions from the input channels. For each input channel c and output channel k there is a bias term b^m_{ck}.



$$(6 \times 5) + (8 \times 5) + (0 \times 8) + (7 \times 1) = 77$$

Figure 2.4: An illustration of a simple CNN filter. The filter is a 2×2 filter randomly initialised with a set of weights for each element of the filter. Each filter weight is multiplied by the corresponding input element over which it is currently covering. All multiplied values are then summed together to produce the output of the filter (explicitly defined in the equation in the bottom of the figure). The filter slides over one element (if stride is set to 1) and the computation is done again until the whole input space has been covered.

Values in the $N \times M$ matrix of the filter are initialised randomly with an added bias term [175]. The initialised values in the filter are the weights of the filter. During training, the filter is convolved with its given input where each element in the filter is multiplied by the corresponding overlapping element in the input. All the multiplied values are then summed together to produce the output of the filter. If there are multiple channels, the filter may also be convolved with each input channel representation and then summed over all channels, where there is a unique filter per input channel. A bias term, b, is also added to the output, where there is a bias for each input and output channel. We may then slide the filter over by 1 column and repeat the convolution process until the edge of the filter reaches the last column of the input. We then slide the filter back to the 1st column of the input and down one row and repeat the convolution process on all columns again. This is done until we have convolved the filter over the whole input. We note that in the case where the data input to the filter is 1-dimensional, we need only simply change the filter size to also be 1-dimensional (i.e. $1 \times M$), or just omit the summation over that particular feature axis and have all quantities described by 1 less dimension.

2.5.2 Pooling Layers

Pooling is a form of regularisation which typically takes the maximum value over a pre-defined local pool of neural network layer outputs. One can also take the average or the weighted average based on the distance from the central element of the local pool, which is generally used as a data reduction step. Pooling is an operation which generally enforces that the output from convolutional filters is invariant to small translations to the input data space. Invariance in this case meaning that if a given input to a filter were shifted by a small amount, the output of the filter would not change by a significant amount. This invariance property is useful if the network is being applied towards a problem where we do not care as much about the location of features in the input data space, but rather would like to place more importance on the features themselves. Pooling can also be useful when dealing with large data inputs to a given layer since a large local pool size provides a single summary statistic for a portion of the input which is equivalent to the local pool size [162].

2.5.3 Striding

In addition to pooling, another form of regularization is striding. Striding determines the number of elements a convolutional filter (or other operation such as pooling) moves after each convolution is performed. Standard techniques which use striding nominally employ a stride of 2 [162]. The two main reasons why striding is applied are to reduce the spatial complexity from one layer to the next and to reduce the overlap of the receptive field of the CNN [194]. The Receptive field is defined as a region of the input space to a given layer which affects the output of the filter being used. With regards to spatial complexity reduction, because we are skipping over some parts of the input when sequentially sliding the filter over each element of the input to the filter, we reduce the amount of computations required and thus the total memory usage of a CNN during training. In terms of reducing overlap of the receptive field, if we reference Fig. 2.5 we see that the input grid is now a 4×4 grid. We can see that with a stride of unity (one) from left to right, the elements in the corners of the of the input will only be convolved with the filter once, whereas the two elements in the middle will each be included in the convolution twice. With a stride of two both the middle two and outer two elements will only be convolved once, thus reducing the amount of repeated convolutions of a filter over some of the receptive field and giving more uniform importance to all areas of the input space.

2.5.4 The Fully-Connected Layers

Following the convolutional filter layers, we now have to convert the output of the CNN to predict a discrete set of classes or regress on some set of parameters. Typically, a flattening operation is applied to the output of the last convolutional layer where the concatenated output from the previous layer's filters are flattened into a 1-dimensional string of elements. This 1-dimensional string of elements are then given as input to a fully-connected layer of neurons. One can then add additional fully-connected layers and a final layer equivalent to either the number of classes to predict, or the number of parameters to regress on. A logical question may be why add this extra fully-connected complication? The primary reason for the addition of fully-connected layers is that we want to unify the learned features across all CNN filters and

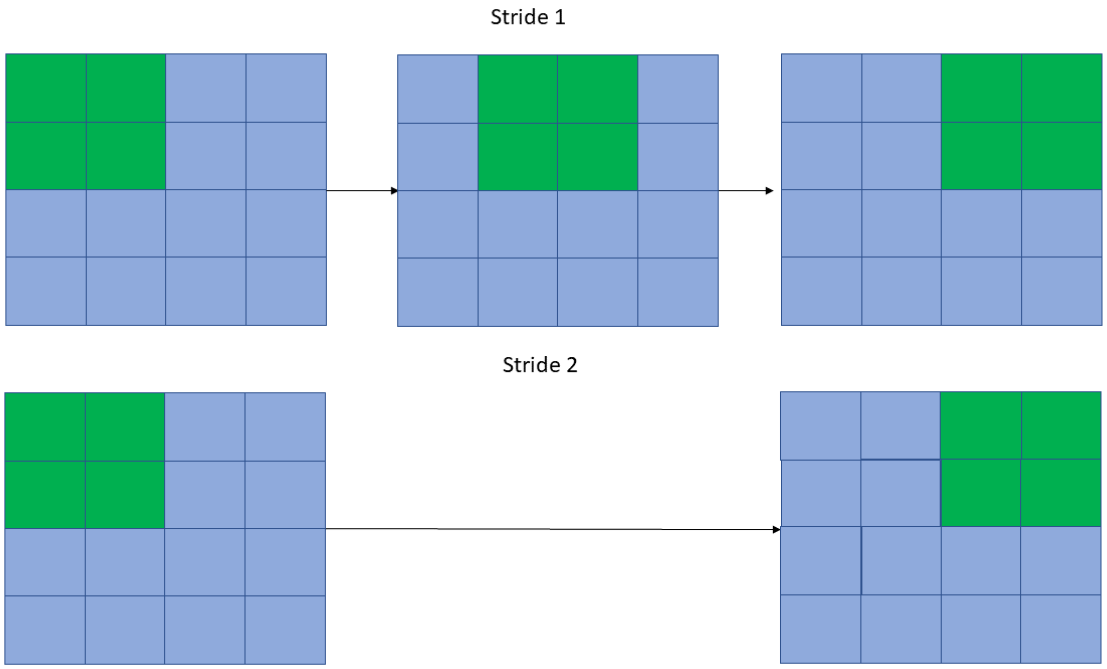


Figure 2.5: An illustration of striding a 2×2 CNN filter (green) with a given data input (blue). The top row is an example of striding the filter across all columns of the input from left to right with a stride of 1. The bottom is an example of striding the filter across all columns with a stride of 2.

use those features as a combined whole to learn how to either classify or regress.

2.6 Conditional Variational Autoencoders

In this section we will describe the network architecture of a CVAE. This will be done by first describing an AE network, then a VAE network and finally a CVAE network.

2.6.1 Autoencoders

An AE is comprised of two neural networks called an encoder and decoder. The encoder is given a data input from a pre-generated training set. The predicted numbers from the encoder representing the latent space are then given as input to the decoder network. The decoder then tries to reconstruct the given input to the encoder network (Fig. 2.6). We also note that the output of the encoder network is typically smaller than the dimension of the input, essentially forming a bottleneck of the data flowing from the encoder to the decoder network. We call the output of the encoder the latent space representation. We can measure how well the network is performing through the same mean squared error function given in Eq. 2.1 but now expressed as

$$H_{\text{AE-MSE}} = \frac{1}{n}\sum_{i}^{n}(y^m(x^{(i)}) - x^{(i)})^2, \tag{2.12}$$

where y is now the output from the decoder network on a given training/validation/testing sample of the AE and $x^{(i)}$ is the input training/testing sample. During training, $y(x^{(i)})$ is trained to be as similar to $x^{(i)}$ as possible in order to minimise $H_{\text{AE-MSE}}$. The lower the value $H_{\text{AE-MSE}}$ is, the better the network is performing. The loss $H_{\text{AE-MSE}}$ is then backpropagated (Sec. 2.1) through the network adjusting the weights and biases in order to further minimise the loss. Through training, the AE learns a latent space representation which is essentially the compressed form of the data.

AEs can be used across a variety of applications including: dimensionality reduction (also known as compression) [195], image denoising [196] and feature extraction [197]. Although powerful, one of the limitations of an AE is the way it represents the latent space, which has

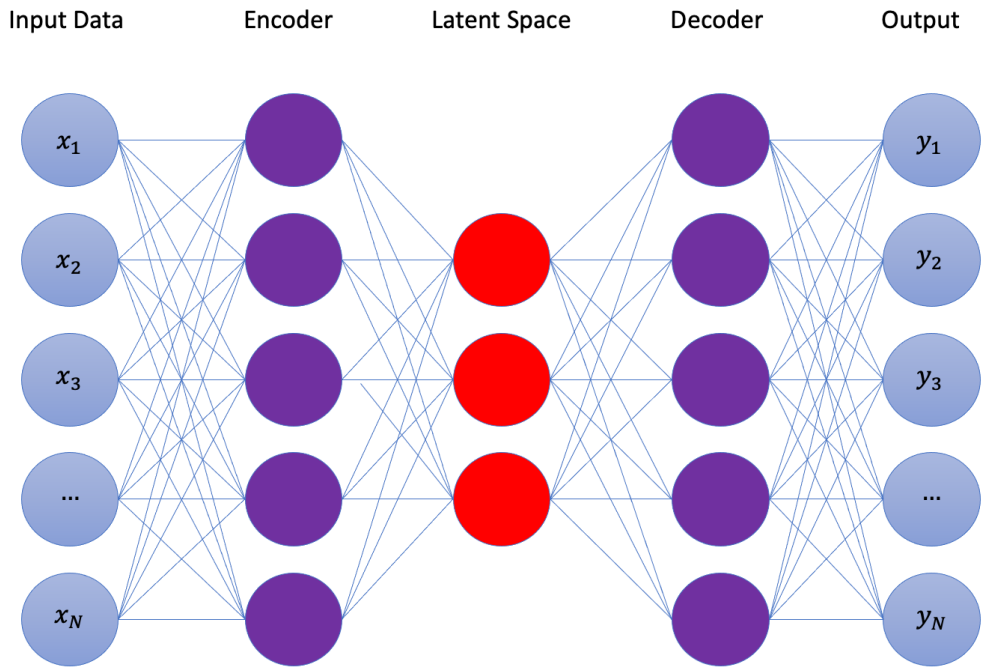


Figure 2.6: An autoencoder network composed of two neural networks defined as the encoder and the decoder networks. Here, both networks are represented as fully-connected networks, but could also be any number of other network architectures such CNNs and LSTM networks. Both networks can also be of any depth or shape. The encoder network takes as input a set of data and compresses the data through a bottleneck called the latent space. The latent space representation is then given as input to the decoder network which tries to reconstruct the given input $x_1 ... x_N$.

implications for their usefulness as generative models. A generative model is one which can produce predictions from the training set parameter space. Within the context of AEs as generative models, the reader would be forgiven for thinking that, if properly trained, one could just simply sample from the latent space uniformly in order to generate unique predictions from the training set parameter space. However, due to the fact that the network is not required to distribute learned latent space representations during training (features are allowed to be encoded anywhere in the latent space), it is not guaranteed that randomly drawn latent space samples will be from a learned part of the latent space.

2.6.2 Variational Autoencoders

A VAE (see illustration in Fig.2.7) is nearly identical to an AE, except for how the latent space is represented and how the loss function is constructed. Considering that the encoder is given an input sample; instead of having the encoder network output a single predicted value for each dimension in the latent space, we now have it output both a predicted mean and standard deviation value describing a Gaussian for each dimension[3]. Samples are then drawn from the predicted distributions and given as input the decoder network. The decoder network produces estimates trying to reconstruct the given input to the encoder network. Through making our encoder network generate latent space samples drawn from a probability distribution, we are essentially forcing the encoder network to make a continuous and smooth latent space representation. If the latent space is smooth and continuous (i.e. two similar classes being "close" in their latent space representation) then the decoder may be able to more easily reconstruct a given input. The loss for a VAE is also slightly different and is represented as

$$H_{\text{VAE-MSE}} = \frac{1}{n}\sum_{i}^{n}(y(x^{(i)}) - x^{(i)})^2 + D_{\text{KL}}[\mathcal{N}(\mu_i, \sigma_i), \mathcal{N}(0,1)], \tag{2.13}$$

where y is the predicted output from the decoder, $x^{(i)}$ is the given input to the encoder, μ_i and σ_i are predicted means and standard deviations describing Gaussians for each latent space dimen-

[3]In principle they don't have to be Gaussians nor do they have to be separate distributions for each latent space dimension.

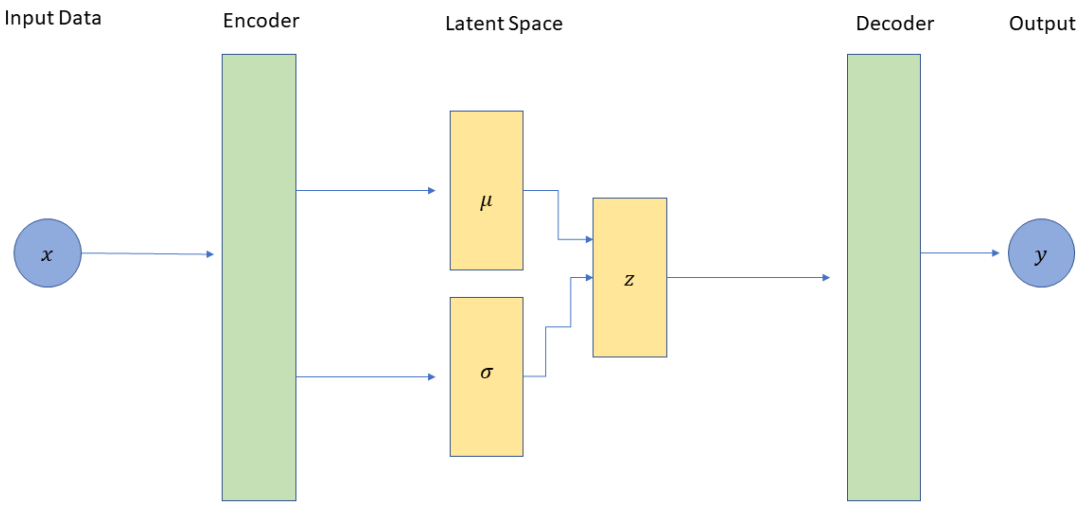


Figure 2.7: A simplified diagram of a VAE network. The input data x is passed to the encoder neural network which produces predictions on the means μ and standard deviations σ of multivariate Gaussian distributions describing the latent space. Samples, z, are drawn from the predicted distributions described by μ and σ in order to get samples from the latent space. These latent space samples are then passed to the decoder network which attempts to reconstruct the given input x.

sion, D_{KL} is the Kullback–Leibler (KL)-divergence between latent space samples from predicted Gaussians $\mathcal{N}(\mu_i, \sigma_i)$ produced by the encoder network and samples from a mean zero unit variant Gaussian distribution $\mathcal{N}(0,1)$.

We can derive this loss function by first assuming that we want to approximate the true posterior in the latent space. We are assuming here that there is a true posterior in the latent space that is a faithful representation of the input data given a limited latent space dimensionality. We are also limited by the capacity of the encoder/decoder in conjunction with finite training time/data representation using a neural network. We will be directly following a similar derivation done in [198]. We can define the difference between the approximated distribution of the latent space and the true distribution of the latent space as the KL-divergence defined in Eq. 2.13 as

$$D_{\mathrm{KL}}(q_\theta(z|x)||p(z|x)) = -\int q_\theta(z|x)\log\left(\frac{p(z|x)}{q_\theta(z|x)}\right)dz, \tag{2.14}$$

where $q_\theta(z|x)$ is the approximate distribution of the latent space given data x and $p(z|x)$ is the true representation given data x. The approximate , $q_\theta(z|x)$, is parameterised by an encoder neural network whose tunable parameters θ are learned during training. We can then substitute Bayes theorem (Eq. 5.1) in for $p(z|x)$ as

$$D_{\mathrm{KL}}(q_\theta(z|x)||p(z|x)) = -\int q_\theta(z|x)\log\left(\frac{p_{\theta_2}(x|z)p(z)}{q_\theta(z|x)p(x)}\right)dz \tag{2.15}$$

where $p_{\theta_2}(x|z)$ is in practice modelled by the decoder network whose tunable parameters we denote here as θ_2. We now separate out the division using the rules of logarithms and distribute the integral

$$D_{\mathrm{KL}}(q_\theta(z|x)||p(z|x)) = -\int q_\theta(z|x)\log\left(\frac{p_{\theta_2}(x|z)p(z)}{q_\theta(z|x)}\right)dz + \int q_\theta(z|x)\log(p(x))dz. \tag{2.16}$$

We also know that by definition the KL-divergence must be greater than zero, so we can can therefore define the inequality

$$-\int q_\theta(z|x)\log\left(\frac{p_{\theta_2}(x|z)p(z)}{q_\theta(z|x)}\right)dz + \int q_\theta(z|x)\log(p(x))dz \geq 0. \tag{2.17}$$

Since $\log(p(x))$ is not a function of z, we can move it outside the integral on the right-hand side. We also know that because $q_\theta(z|x)$ is a probability distribution, we can state that the integral of $q_\theta(z|x)$ with respect to z is equal to 1. Moving the integral over to the right-hand side to isolate $\log(p(x))$ we get

$$\log(p(x)) \geq \int q_\theta(z|x) \log\left(\frac{p_{\theta_2}(x|z)p(z)}{q_\theta(z|x)}\right) dz. \tag{2.18}$$

Using the rules of logarithms we can write

$$\log(p(x)) \geq \int q_\theta(z|x) \left[\log\left(\frac{p(z)}{q_\theta(z|x)}\right) + \log(p_{\theta_2}(x|z))\right] dz. \tag{2.19}$$

Distributing the $q_\theta(z|x)$ term and the integral we get

$$\log(p(x)) \geq \int q_\theta(z|x) \log\left(\frac{p(z)}{q_\theta(z|x)}\right) dz + \int q_\theta(z|x) \log(p_{\theta_2}(x|z)) dz. \tag{2.20}$$

We can now see clearly that the first expression on the right-hand side of the above inequality is equivalent to the negative KL divergence between our approximate latent space representation $q_\theta(z|x)$ and our prior on the latent space $p(z)$ such that

$$\log(p(x)) \geq -D_{\mathrm{KL}}(q_\theta(z|x)||p(z)) + \int q_\theta(z|x) \log(p_{\theta_2}(x|z)) dz. \tag{2.21}$$

Conveniently, we also see that the right most term on the right-hand side of the inequality can be approximated as the expectation value of the log probability of the data x given z assuming z is drawn from $q(z|x)$. Hence

$$\log(p(x)) \geq -D_{\mathrm{KL}}(q_\theta(z|x)||p(z)) + E_{\sim q_\theta(z|x)}[\log(p_{\theta_2}(x|z))], \tag{2.22}$$

where $z \sim q(z|x)$. This term now gives us an evidence lower bound (ELBO), which places a lower bound on the log-evidence of the data $\log(p(x))$. If we assume that the output of the VAE decoder, $p_{\theta_2}(x|z)$ has a Gaussian distribution, then the log-probability of the output x will be equivalent to the log of a Gaussian. Thus the expectation value on the right reduces to the mean squared error between predictions on x by a neural network called the decoder which is

given latent space samples from a multivariate Gaussian distribution whose mean and variance are predicted by the encoder network $q_\theta(z|x)$. Assuming a Gaussian prior on our latent space, we can also write the KL-divergence term as the KL-divergence between samples drawn from a mean zero unit variate Gaussian distribution and those from a multivariate Gaussian distribution whose mean and standard deviation is predicted by the encoder network q. The KL term essentially acts to constrain the form of the approximate posterior $\log(p(x))$ according to the chosen prior on the latent space representation $p(z)$ across the whole input parameter space. Furthermore, the KL term helps the model to learn a well-formed latent space and reduce the likelihood of the posterior being too different from the prior [199]. Thus Eq. 2.22 can be rewritten in the form of a cost function to be minimised given earlier by Eq. 2.13.

2.6.3 Conditional Variational Autoencoders

Through our derivations in the previous sections we have now formulated a loss function which can be trained to maximize the expected log-probability of our decoder's prediction, while also regularising its model within the latent space to a smooth representation that matches a predefined Gaussian prior. However, there are some minor drawbacks to the VAE which make it unsuitable for some generative tasks. Generative in this case means to draw samples from the latent space on a pre-trained encoder network and then feed those samples to a pre-trained decoder network in order to generate new data samples from the prior distribution defined by the training set.

For example, if we wanted to train our VAE on the fashion Modified National Institute of Standards and Technology dataset (MNIST) dataset [200][4] , a database of digital images representative of various classes of fashion items (shoes, shirts, trousers), in order to generate new images of fashion items we would run into a problem. This problem is most clearly illustrated when taking a closer look at the VAE cost function in Eq. 2.13. In the VAE cost function, it can seen that the encoder is solely conditioned on inputs from the training space x, but not on what class/type of image x is. Similarly, the decoder is solely conditioned on the latent space

[4]Each image in the sets are normalised to be 28×28 pixels in size and are grayscale, such that each pixel is represented by a single scalar.

z. In order to produce a particular class of input from the decoder, we would have to know in which part of the latent space each class lives and to then draw from the region of the latent space that our class of interest resides. Knowledge of class location is infeasible to obtain after training because the KL-divergence term in Eq. 2.13 enforces that on average across the whole training set that the predicted latent space locations are representative of a mean zero Gaussian distribution, but does not constrain where individual classes of input may reside.

The natural question then arises, is it possible to condition the network on classes of interest? It turns out that this is possible to do by starting from first principles (as was the done for the VAE in Eq. 2.13) and then deriving an expression through modeling the joint distribution on x and c such that

$$\log(p(x,c)) \geq -D_{\mathrm{KL}}(q_\theta(z|x,c)||p(z,c)) + E_{\sim q_\theta(z|x,c)}[\log(p_{\theta_2}(x|z,c))]. \tag{2.23}$$

Here, we now condition the encoder $q_\theta(z|x,c)$ on both the training data x and the class of said images c (see Fig.2.8). In the decoder $p_{\theta_2}(x|z,c)$ we condition on samples from the latent space z and the class c. For classification purposes, the class may be parameterised as a single scalar number which is appended to the input to the encoder and decoder networks. Now, if we wanted to use the decoder as a fashion image generator, we would only simply need to draw a random sample from the latent space from the prior on z, choose a class label, and then feed both as inputs to the decoder network. In Ch. 5 we will show how we use a form of CVAE in order to perform low-latency GW Bayesian parameter estimation.

2.7 Summary

In this chapter we introduced the concept of machine learning starting from one of the simplest neural networks, a perceptron. We showed how perceptrons may be used to provide predictions on a given input in the form of classification or regression tasks. We then built on the concept of perceptrons discussing how perceptrons can be strung together to form a layer, and from there layers can be stacked to form a deep fully-connected neural network. The cost function for a

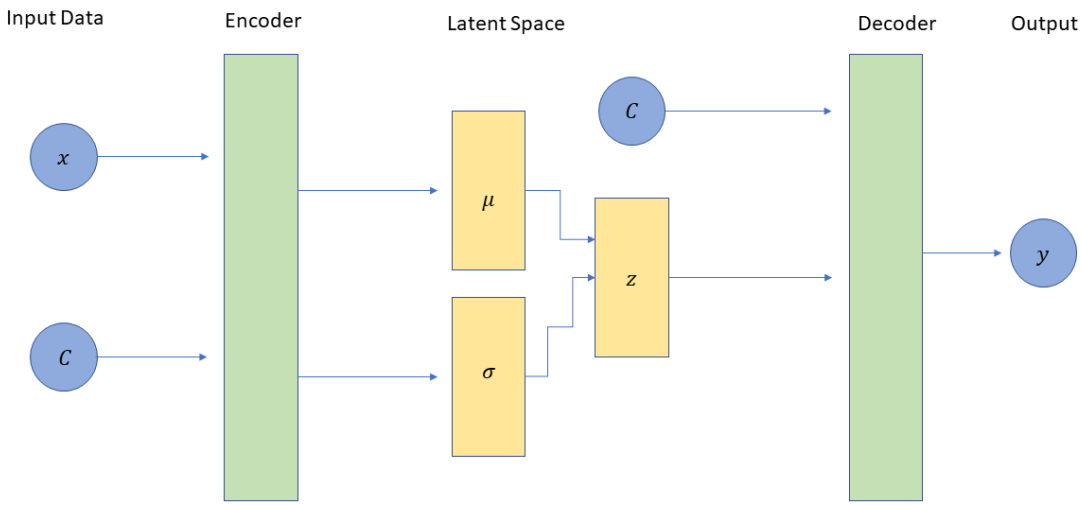


Figure 2.8: A simple CVAE diagram consisting of an encoder and decoder network. x data is given as input to the encoder and latent space samples are given as input to the decoder as was the case with the VAE diagram in Fig.2.7. However, now we additionally condition the encoder on the class of x by appending the class label in numerical form to x. We also condition the decoder on class c by appending it to the latent space samples z.

deep network was described, along with how that cost is used to update the tunable parameters (weights and biases) of a neural network through a mechanism known as backpropagation. We also provided some best practices on hyperparameter optimisation and training procedures such as dropout, batch normalisation and data augmentation. We additionally introduced the concept of CNNs, AEs, VAEs, and CVAEs. We mathematically expressed how the cost function for each method may be described and discussed how all models are used in practice. Two of the networks introduced in this chapter (CNNs, CVAEs) are the primary methods used in this author's thesis work. In the following chapter (Ch. 3), we will discuss how many of the ML techniques discussed in this chapter (including others not mentioned) are being applied across a variety of domains in GW astronomy to great success.

Chapter 3

Machine Learning in Gravitational Wave Astronomy

It is expected that in the coming years the global advanced detector network will see an increase in sensitivity. With this increase in sensitivity also comes an increase in the number of CBC signals which will be detectable by the collaboration. The number of signals per year is expected to be anywhere on the order of 100s to 1000s (depending on the source type) [18]. We also note that the time to compute full Bayesian posteriors on the sky location of such events can take upwards of weeks to months, where delays in sky location alerts to EM partners can mean missed EM signatures [21] (with possible adverse future science discovery impacts). As such, it is imperative that we develop new techniques to deal with this massive influx of detectable sources. We also note that some signals [201] and noise cannot be accurately modelled using traditional approaches, thus there is also a need to develop flexible tools to solve this problem. ML has been proven to be an excellent resource for tackling many problems in the GW community. Over the past several years, there has been a boon in the use of ML methods across a variety of applications in detection, parameter estimation and detector characterization algorithms (among other domains). This surge in applications has seen marked success not only in proof-of-principle studies, but even in studies involving the use of actual LVK data. Going forward, it is the hope of this author and surely other ML GW practitioners, that after rigorous testing, these methods continue to be more widely used and for some, eventually implemented in realtime during

observing runs in order to further enhance the capabilities of future GW detectors.

In this chapter we will provide an overview of recent advances in ML approaches in GW astronomy in signal detection, parameter estimation, population inference and detector characterization.

3.1 Machine Learning for Gravitational Wave Detection

Algorithms which search for GWs are typically concerned with four types of sources: CBC, Burst, CW and stochastic GWs (Sec. 1.6). CBC sources being composed of BBH, BNS and NSBH signals (Sec. 1.6.1), CW sources composed of fast spinning NSs (Sec. 1.6.3), Burst are unmodelled/poorly modeled signals like supernovae or even as yet unknown sources (Sec. 1.6.5), and stochastic GWs result from left over remnants of the Big Bang along with poorly resolved CBC signals from extreme distances (Sec. 1.6.7). In this section we will outline several recent and interesting studies which have used ML to directly detect GWs from a variety of the aforementioned sources listed above. We also note that there a number of papers which have been applied to this area and have thus not mentioned all for the sake of brevity.

3.1.1 Compact Binary Coalescence Detection Studies

Deep learning algorithms were first applied to the task of detecting simulated GWs by George & Huerta [202] and Gabbard et al. [1]. In our work (Ch. 4), our ML algorithm was trained on two distinct timeseries classes, simulated BBH waveforms buried in Gaussian noise, as well as purely Gaussian noise timeseries[1] in order to perform a classification task. Gaussian noise was used because it is known that matched filtering performs efficiently under Gaussian noise conditions [67] and the analysis of signal in Gaussian noise was a simple first step towards trying to match the efficiency of matched filtering. We note that it is known that matched filtering is not the optimal approach when searching over discrete waveform templates [203, 204] and ML has in fact since been shown to even exceed matched filtering sensitivity in some circumstances [204]. Using a CNN, we were able to show that deep learning could match the efficiency

[1]The noise generated was not white and was simulated from a realistic detector PSD

of matched filtering. In [202], they also show that a similar CNN approach ("deep filtering") could reproduce predictions from matched filtering. In addition to signal detection, they were able to show that a CNN, for the first time, could perform point-estimate parameter estimation on both BBH and eccentric BBH waveforms, closely matching the best-fitting templates predicted by matched filtering. It should be noted here that "close" is a relative and somewhat meaningless term if there are no uncertainties on those estimates. They additionally compare their ML approach to other common ML approaches including: K-nearest neighbor, support vector machines and logistic regression (among others) showing their CNN to have superior results.[2]

George & Heurta use two network architectures in their study: one for parameter estimation and one for signal detection. The parameter estimation network is composed of 4 convolutional layers, 4 max pooling layers, 1 flattening layer, 3 fully-connected layers and rectified linear unit activation functions on all fully-connected/convolutional layers, whereas the signal detection network is slightly smaller with 3 convolutional layers, 3 pooling layers, 1 flattening layer and 2 fully-connected. Gabbard et al. on the other hand used a slightly deeper network with 6 convolutional layers, 3 max pooling layers, 1 flattening layer, 2 fully-connected layers and exponential linear unit activation functions on all fully-connected/convolutional layers. In both studies, the number of filters in each convolutional layer increases up to the flattening layer, while conversely the kernel size decreases. The time to train both George & Huerta networks was shown to be a few hours and evaluation of a single test sample took ~ 6.7ms for signal detection and ~ 85ms using the parameter estimation network. Gabbard et al.'s network took ~ 1 hour on a single GPU and could produce signal detection statistics for a test sample on the order of milliseconds.

It was first shown by Krastev et al. [206] that using artificial neural network (ANN)s one could accurately recover BNS signals in Gaussian noise. The problem was posed as a trinary classification task whereby the ANN model was tasked with distinguishing between BBH, BNS and Gaussian noise alone timeseries segments. Their neural network was limited to analysing data selected from a sliding window of ~ 10s due to computational expense associated with

[2] An alternative approach to CNNs that may be interesting to explore are wavelet convolutional neural networks which have been shown in some circumstances to have similar performance with respect to CNNs with the added advantage of significantly lower computational expense [205].

analysing long timeseries segments. They also train their network through curriculum learning whereby the network is initially trained on easy-to-detect high SNR signals and then gradually given more and more low SNR signals as training progresses. Their results are quantified by comparing the true alarm probability (TAP), the fraction of signals correctly identified and FAP of predictions from their neural network classifier for different classes of signal (i.e. BBH, BNS, noise-alone) (also known as a receiver operator characteristic (ROC) curve [207]). Their results show that that their network is more sensitive to detecting BBH events than for BNS events. The authors show that their ANN performs more poorly on BNS signals than on BBH signals. To expand, 100% of BBH events are detected above an SNR of 10, whereas 100% of BNS events are only detected above an SNR of 18. They additionally quantify their results by plotting the TAP as a function of SNR at different FAP values for both BBH and BNS neural network predictions. Their results for the BBH case reach similar levels of sensitivity as those achieved by Gabbard et al. [1]. They state that the sensitivity difference between BNS and BBH neural network predictions could be due to a number of reasons including: sliding window size, network architecture choice and signal complexity. The authors do not compare their approach to any other existing currently used signal detection method such as matched filtering.

In Marlin et al. [208], the authors perform a more in-depth study on ML for BNS detection and use a different neural network architecture. In order to reduce the overall complexity of the input space they deal with 32s long BNS signals and vary the sample rate as a function of different segments in time of the signal. The authors justify this sampling approach by describing that most of the SNR of the signal is contained in the final few seconds. Additionally, if one looks at the frequency evolution of a BNS signal, we see that, for example, at 16 s prior to merger for a 1.35-1.35 system, the GW frequency is 48.7 Hz. Hence prior to this, a sampling frequency of at least 100Hz would theoretically suffice. They state that using a varying sample rate also reduces the size of input signal space by a factor of ~ 9, thus decreasing the computational expense needed to run their approach. Their approach was also the first to probe ML-based detection algorithm sensitivities down to FARs of about 0.5 per month. The authors also improve on previous architecture approaches by employing inception modules [209], temporal convolutions for signal amplification, [210] and tailoring inception modules for each different sampled section

of the input time series [208]. Results from their study show an overall improvement in BNS detection sensitivity with respect to results from Krastev et al. [206]. They quantify this by comparing results from their study with Krastev's study by plotting the TAP as a function of SNR for both studies. Marlin et al. are able to produce predictions with their neural network on a given input with a latency of ~ 10.2s. The authors compare their results to traditional techniques like matched filtering (i.e. the PyCBC live event trigger generator [6]) using a quantity known as the sensitivity distance [208] at fixed FARs. The authors find that their ML approach has a sensitivity distance which is ~ 6 times lower (less sensitive) than that of results from PyCBC [6] at a FAR of 0.6 per month. The authors also compare their results to Krastev et al. [206] and see that for a given fixed FAR, Marlin et al.'s network does not generalise well to high SNR signals to the same sensitivity as Krastev et al. [206] in terms of TAP values. On the other hand, the results of Marlin et al. show a ~ 4 times increase in TAP sensitivity over Krastev et al. [206] in the lower SNR regime ($8 \leq \text{SNR} \leq 15$). The authors mention that some general improvements could be made including: lowering the latency of their ML approach by reducing the complexity of the network, the addition of real noise to the training/ testing sets, and using results form their approach as input to follow-up analysis by a traditional matched filtering algorithm with a heavily reduced template bank.

We also note here that there are many other studies that have been done within the context of CBC detection using ML that we do not have the space to mention in this chapter. Some other notable examples include: using scalable autoencoders for CBC GW detection [211], fully-convolutional neural networks for GW signal detection [212, 213], genetic-algorithm-optimized neural networks for CBC GW detection [214], using CNNs given time-frequency input representations to detect CBC GWs [215], and detecting the early inspiral of CBC GW signals using CNNs [216]. We will now discuss how ML is being applied towards the detection of burst GW signals.

3.1.2 Burst Detection Studies

The first implementation of a ML algorithm for the purpose of identifying burst-like signals was by Astone et al. in 2018 [217]. In their paper they build upon the already existing Coherent

WaveBurst (CWB) pipeline [136] used by the LVC. Specifically, they are interested in searching for core-collapse supernovae events [128]. Their method can be summarised in two steps. First, they prepare time-frequency images of detector data using the CWB event trigger generator, which searches for periods of excess power and combines these periods coherently among the various detectors (for further details on coherent searches see either Ch. 1 or [136]). Once periods of excess power have been identified, a time-frequency spectrogram image is constructed for each active detector. In their approach, they use three detectors and combine them in such a way that each detector essentially acts as a colour channel, akin to a red-green-blue (RBG) colour scheme. The authors then use a CNN to classify the identified time-frequency image into either noise or signal+noise classes. They test their network on a range of signals spanning SNR values from 8 up to ~ 40. They show that their method is overall more efficient at detecting core-collapse supernovae signals than the standard detection pipeline CWB at a FAR of around 7×10^{-5}Hz.

It was also shown in Chan et al. [218], that CNNs may be used to detect GWs resulting from core collapse supernovae events. In their work they find that their approach would be able to detect core collapse supernovae events out to a distance of 60kpc with a TAP of $\sim 54\%$ at a FAP of 0.1%, well within the distance to the large and small Magellanic clouds. Additionally, McGinn et al. [201] applied a ML algorithm known as a generative adversarial network (GAN) [219] in order to produce generalised GW burst waveforms in the time domain. Their network was trained over 5 different classes of waveforms commonly used by burst searches and find that, once trained, their GAN model is able to accurately produce waveforms on-command which are similar in waveform characteristics those it was trained over, as well as generalise to other waveforms which act as hybrids between the 5 training classes. They test the practical aspects of their work by first training a CNN solely using the 5 well-defined classes of waveforms used by their GAN during training and then training another identical CNN using a training set which is generated using waveform timeseries produced by their GAN. The authors find that the CNN trained with waveforms generated by the GAN had superior performance over the CNN trained with the 5 classes alone and quantify this through efficiency curves which measure the TAP as a function of SNR at fixed FAP values.

Finally, we note the recent study by Skliris et al. [220] which showed for the first time that a multi-component CNN architecture was able to classify unmodelled bust signals. They train their model using white-noise burst signals [135] buried in Gaussian noise and Gaussian noise-alone timeseries. Their network architecture is unique in that it employs two CNNs: one for identifying signals which are coincident between detectors and the other for detecting correlations between detectors. The first CNN was derived from the same architecture as Gabbard et al. [1] and is given as input a timeseries and produces an output between 0 and 1, denoting the probability of the timeseries containing either a signal or noise respectively. The second network takes as input again a timeseries, but is also given the Pearson correlation [220] for each pair of detectors and outputs values between 0 and 1, where 0 indicates low correlation and 1 indicates high correlation between detectors. This dual CNN setup is inspired by the operation of standard burst GW detection pipelines which require that a signal is both coincident and correlated between detectors [220]. They test their method under a variety of waveform morphologies and illustrate their sensitivities through computing TAP values at fixed FAPs across a range of SNR values.

3.1.3 Continuous Wave Detection Studies

In terms of CW searches, the standard coherent or semicoherent searches can be computationally expensive, sometimes taking well over a month to complete [221]. Dreissigacker et al. [221] try to decrease this computational time with respect to standard CW detection approaches using ResNet neural networks [186] in the first application of deep learning for the CW search. They frame the problem such that their training set consists of two types of input. One, being long 10^6s signals and shorter 10^5s signals. Rather than doing a search over the entire frequency range, they limit their test cases to a discrete set of frequency bands, 50mHz in width, at (20Hz, 100Hz, 200Hz, 500Hz and 1000 Hz). They compare their ML results to a "gold standard" CW search method pipeline `WEAVE` [222]. The pipeline sums coherent F-statistics [118] over semicoherent segments on lattice-based GW template banks [223], although their study employs the fully coherent search version of the code. The reason there is both a semicoherent and a fully coherent version of the `WEAVE` code is related to the computational cost of performing a fully coherent

search. Given that CW signals are weak in amplitude, one needs to analyse long periods of data in order to accumulate enough SNR to make a detection. However, the computational expense of performing the optimal fully coherent search scales as T^n where $n \gtrsim 5$ and T is the length of the data analysed (this is especially problematic at high frequencies [224]). Semicoherent methods for performing a CW search were thus developed with the purpose of making it possible to analyse longer stretches of data with reduced computational cost.

For the training set, Dreissigacker et al. generate over 10^4 training waveforms in the frequency domain with half containing Gaussian noise and the other half containing signal+noise. They found that using a set greater than 10^4 training signals provided little gain in efficiency. After training, the authors found that at a fixed FAP, their deep learning approach appears to be competitive ($88-73\%$ detection probability) with the matched filtering search ($\sim 90\%$ detection probability) on short time scale inputs (10^5s). However, when tasked with longer time scale inputs (10^6s), the neural network performs at a diminished detection probability ($69-13\%$). In terms of computational speed, it was also shown that their neural network outperformed the matched template search, taking only on the order of a few milliseconds ($3-10$ms) as opposed to the 10^6-10^9s of the matched filter method. The authors do also note however that the network requires $\sim 1-10$ days to train depending on the length of the observed data, T, used. As the authors state, there is much work to be done in order to improve the efficiency of deep neural networks within the context of CW searches. There is also much promise given the quality of results achieved thus far. In a follow-up to [221], Dreissigacker et al. show in [225] that using an Inception-ResNet [226] architecture they were able to train a deep neural network to classify the presence of CW signals given data from 2 detectors simultaneously. They also showed that their network code was able to show an approximately equivalent sensitivity under both targeted search and all-sky search CW scenarios. We also briefly note other recent ML CW studies such as Morawski et al. who showed an application of CW signal detection using CNNs [227], as well as Beheshtipour and Papa who used a Mask R-CNN [228] architecture for CW signal candidate clustering [229].

3.2 Machine Learning for Gravitational Wave Bayesian Parameter Estimation

For many, GW parameter estimation was the "holy grail" of ML in the field of GW astronomy. Although prior to 2019 there were some attempts to compute point estimates for GW source parameters directly, as well as to incorporate ANNs directly into subsections of existing Bayesian inference algorithms [159], there had up to that point been no studies done using an ML algorithm which went from directly from strain to computing samples from the Bayesian posterior on GW source parameters. In this subsection, we will outline some recent studies on incorporating artificial intelligence (AI) into existing GW parameter estimation pipelines, ML for point estimate parameter estimation, and finally ML for directly computing samples from the Bayesian posterior.

ML was first used in the context of GW parameter estimation by Graff et al. [159] in their modified nested sampling algorithm `BAMBI`. In their algorithm, the authors exploit the tenants of the universal approximation theorem [230], which implies that even a properly optimized ANN should be able to approximate any sufficiently complicated likelihood function. This is done by essentially replacing the computationally expensive likelihood calculation in the `MultiNest` [231] nested sampling algorithm (Sec. 1.7.2). After the nested sampling algorithm has produced a sufficient number of samples, their ANN is trained on those samples. The inputs to the ANN are the sample parameter values and the output is a single scalar which represents the likelihood values for each of those samples. The ANN is then trained such that it's predictions for the likelihood values match those computed within the nested sampling algorithm. The advantage of having such a ANN is that once trained, the ANN can provide low-latency likelihood evaluations. In order to quantify when the ANN has been trained sufficiently to fully replace the standard likelihood function, a tolerance level is calculated. The tolerance is defined as the standard deviation of the difference between the true log-likelihood values from nested sampling and the predicted values from the ANN. The actual value of this tolerance level may be specified by the user. Once the network has reached an acceptable tolerance level of performance it then replaces the original likelihood function calculation. Periodic tolerance checks

are made throughout the rest of the nested sampling iterations to ensure that the ANN is still providing solid predictions.

The authors find that their algorithm performs well across a variety of toy model cases (Gaussian shells, Rosenbrock functions [232], eggbox [231]). They also compare their approach (`BAMBI`) to unmodified `MultiNest` on a few real physics examples including predicting the posteriors of 8 cosmological parameters using data from cosmic microwave background (CMB) observations and CMB observations plus Hubble Space Telescope constraints on H_0, the Sloan Digital Sky Survey and Type 1a Supernovae data (see [159] for more details on datasets used). The gain in speed goes from taking on the order of seconds per likelihood calculation, down to milliseconds, a three order of magnitude increase in computational performance with accurate evidence predictions and posterior probability distributions [159]. It has also been shown within the context of GW BNS signals that `BAMBI` can produce posteriors which are consistent with traditional nested sampling algorithms, generating samples from the posterior ~ 100 times faster [233].

One of the first implementations to use a purely ML-based algorithm to produce Bayesian posteriors was performed by Gabbard et al. in [2]. In our study (also discussed in detail in Ch. 4), we show that using a particular type of CVAE, given a timeseries, we can produce Bayesian posteriors over the full GW BBH parameter space. The network is trained over BBH GW waveforms in Gaussian noise with a component mass range of $35-80\,\mathrm{M}_{\odot}$, as well as the source parameters which characterise the BBH waveforms. Once trained, our network can produce $\sim 10^4$ posterior samples per given timeseries in ~ 0.1ms. We compare our results against four benchmark Bayesian samplers (`Dynesty`, `ptemcee`, `emcee`, `CPNest`) and find that our ML model is generally consistent with other benchmark approaches. Consistency is quantified through a combination of JS-divergence and p-p plot tests, as further outlined in Ch. 4.

Released at the same time as Gabbard et al. [2], Chua et al. [234] also aimed to show that ML methods could accurately approximate Bayesian GW posteriors. Their method used fully-connected neural networks (Ch. 2). They represent their training waveforms by first training a neural network to predict coefficients which describe a reduced order framework [235]. Once

trained, the network is able to quickly produce training GW waveforms. Their training GW waveforms are represented in the frequency domain, whereby there are both real and imaginary components of the wavefrom. The authors use two networks which produce the real and imaginary components of the waveform separately. To produce posteriors, Chua et al. again use a fully-connected neural network which takes as input during training the training waveforms (in Gaussian noise), as well as the source parameter values of those training waveforms. Their model then outputs a 1 or 2−dimensional posterior in the form of a multivariate Gaussian whose means and covariance matrix are predicted by the neural network. When testing, the model is constructed such that it only needs as input the test sample waveform alone. One can then produce samples from the posterior for a given input waveform by sampling from the predicted multivariate Gaussian.

Chua et al. [234] test and train their model on BBH waveforms, expected to be seen by space-based detectors such as LISA, in Gaussian noise with masses ranging from $1.5 \times 10^5 - 10 \times 10^5$ with aligned spins between -1 and 1. They then train their model to predict both the chirp mass and the symmetric mass ratio . Their results show good agreement between the posteriors produced by their machine learning approach and traditional Bayesian sampling approaches [234, 235]. They more formally quantify the accuracy of their results by computing the sample covariance matrix of the posterior predicted by their neural network, as well as the covariance matrix of the true Bayesian posterior and then comparing both through a log ratio expression given in Eq. 9 of [234].

Following on from Chua et al. [234] and Gabbard et al. [2], Green et al. [236] wanted to tackle two outstanding problems which both Chua et al. and Gabbard et al. had yet to solve at the time. With regards to Chua et al.'s approach, it was to expand upon the limited number of source parameter dimensions that their algorithm could produce posteriors on at a time (no more than 2). With regards to Gabbard et al., it was the initial difficulty of dealing with multimodal posteriors (which was later rectified in subsequent updates to our paper). In Green et al.'s first paper [236], they implemented a new ML approach using a combination of normalising flows [237] and CVAEs (Ch. 2).

A normalising flow is a mechanism for mapping simple distributions to more complex distri-

butions through a series of transformations, while also retaining the property of being invertible. Invertibility meaning that in order to sample from the complex distribution, we must also still be able to sample from the simple distribution. As stated in [237], a normalising flow may in fact be represented as a neural network where the parameters which characterise the series of transformations in a flow, are learned by the neural network over the course of training [233].

In their paper, Green et al. use three distinctly different models to produce Bayesian posteriors: a CVAE, a masked autoregressive flow (MAF) [238] and a combination of a CVAE and a MAF. The combined CVAE+MAF network, is constructed by appending flows to the end of both encoders and the decoder network of the CVAE in order to better model more complex distributions in the latent space of the CVAE. The authors test their methods on 1 s long BBH signals in Gaussian noise sampled at 1024Hz and try to predict a 5 parameter case $(m_1, m_2, \phi, t_0, d_{\rm L})$ in order to directly compare with Gabbard et al. [2], and an eight dimensional case $(m_1, m_2, \phi, t_0, d_{\rm L}, \chi_1, \chi_2, \iota)$ which uses an aligned spin model. Both of these cases are using a single detector with fixed sky location. Prior to training, they perform a unique pre-processing step whereby they transform their training samples (GW waveforms in Gaussian noise) via principal component analysis into 100 element long series of principal components, thereby reducing the dimensionality of the input and vastly improving the overall computational cost of training/testing their network. They find that all three methods perform reasonably well, although the basic CVAE alone method is not able to resolve the multi-modal source parameters such as phase. They validate their results quantitatively through a combination of p-p tests and computing the KL divergence between multi- dimensional posterior distributions from their ML approaches and those from a benchmark Bayesian sampler, `emcee`, with KL values of ~ 0.3. For context they also computed the KL divergence between samples which are drawn from identical posteriors and found that value to be ~ 0.1.

In a subsequent paper, Green et al. [239] used a pure MAF flow network with spline couplings [240]. The authors train their model and then test on a known GW signal in Gaussian noise (GW150914 [3]). After training, they find that their approach is able to to closely match a standard Bayesian sampler (`Dynesty` [160]), according to corner plots, JS-divergence values and p-p plot tests. They find minor inaccuracies on the inclination angle, where their ML

approach puts more imbalance in the posterior on one of the two modes in the posterior space.

Using normalizing flows, Williams et al. [241] were able to replace the computationally costly task of drawing samples from the prior constrained by the likelihood contour of the lowest (worst) live point in an existing nested sampling algorithm with flows. The flow tries to model the multi-dimensional multi-modal likelihood contour defined by the worst live point. Classically, the nested sampling algorithm has the issue of trying to efficiently sample from inside the final likelihood contour, where as the analysis progresses and the contour becomes smaller, basic nested sampling algorithms become less efficient. The authors use an algorithm, dubbed Nested Sampling with Artificial Intelligence (`NESSAI`), to try and solve this problem with a specific type of flow, coupling flows, because of their computational cheapness (although with the added disadvantage of tending to be slightly less flexible than autoregressive flows). The flow is trained on the current set of live points through a loss function defined in Appendix A of [241]. Once trained, the flow is then used to generate a population of new proposal points. A point is randomly drawn from this proposal population and its likelihood is calculated. The proposed point replaces the old worst live point if the proposed point likelihood is greater than the worst live point likelihood. This process is repeated until one of a number of criteria are met: the proposal population generated is depleted, acceptance rate of points drawn from proposal population is too low, a pre-determined number of new live points have been accepted, some user-defined nested sampling convergence threshold is passed. This is all further outlined in Sec. 3B of [241]. The authors find that when compared to the Bayesian sampler `Dynesty`, their approach is between ~ 2.32 (without distance marginalisation) and ~ 1.40 (with distance marginalisation) times faster. The majority of the computational cost of the `NESSAI` algorithm is expended through the proposal population generation stage (36% of the total cost for a given injection), while flow training only accounts for a small percentage (9%). The proposal point likelihood evaluation stage cost decreases as the number of CPU threads used in the analysis increases, falling to approximately 9% of the total cost when using 16 threads. We also mention that there are many other works in ML for Bayesian parameter estimation which we do not have the space to discuss in more detail here. We highlight a few additional studies where for example it was shown that CNNs could be used to model the likelihood-to-evidence ratio in an

MCMC sampling algorithm [242]. It was demonstrated in [243] that a CVAE (using a similar scheme as Gabbard et al. [2]) could be trained by also conditioning on the ASD, thus allowing their network to generalise to a time varying ASD. Finally, it was also illustrated in [244] that a cross-residual network [245], given spectrogram data, could produce probability distribution of estimates on GW source parameters[3]. We will now discuss in the next section some new work on using ML for GW population inference.

3.3 Machine Learning for Population Inference

Given the recent accumulation of GW detections over the past several LVC observation runs, we are now able to perform more detailed studies across a population of GW events. It is also expected that as detector upgrades are made and new detectors come online, that we will start to see hundreds to thousands of GW events per year [18]. Such abundance of signals will only enhance population studies which aid us in understanding GW source formation channels, the general properties of GW source parameters (masses, redshifts, spins etc.), progenitor merger rates, as well as possible modifications of GR [247]. In order to constrain certain phenomenological models, it is common to employ the use of Bayes factors (Ch. 1, Sec. 1.7)) comparing different model assumptions. Employing such Bayes factors commonly involves the production of computationally costly simulations of synthetic populations of GW events and then comparing those synthetic populations to real-world data [248]. One way of circumventing this is through ML.

Wong et al. [249] showed for the first time that a hybrid normalizing flow—hierarchical Bayesian model framework was able to accurately constrain GW population models [249]. Specifically, the authors try to get a normalising flow to estimate the population likelihood, $p(\boldsymbol{\theta}|\boldsymbol{\lambda})$ (See Eq. 1 in [249]), where $\boldsymbol{\lambda}$ are a set of hyperparameters which describe the general properties of a set of GW events. Parameters which characterise GW events from the GW event set are given as $\boldsymbol{\theta}$. The population likelihood $p(\boldsymbol{\theta}|\boldsymbol{\lambda})$ is needed in order to calculate the

[3]We note that the probability distribution generated is not a Bayesian posterior, but rather a distribution which models the uncertainty due to the stochastic processes of the neural network itself. This is done through a process known as Monte Carlo Dropout [246]

likelihood of observing a particular set of GW events given population hyperparameters $p(\boldsymbol{d}|\boldsymbol{\lambda})$ (See Eq. 1 of [249]) through hierarchical Bayesian model selection. Their normalising flow is given as input parameters $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$ which characterise simulated GW event catalogue sets and produces as output the population likelihood $p(\boldsymbol{\theta}|\boldsymbol{\lambda})$. Wong et al. show that their technique performs well over what were once considered highly intractable models. They compare their results to simulations using an analytic phenomenological model, which provides an exact analytic expression for $p(\boldsymbol{\theta}|\boldsymbol{\lambda})$ which they can directly compare their predictions of $p(\boldsymbol{\theta}|\boldsymbol{\lambda})$ from the normalising flow against. Although the authors neglect to take into account event uncertainties and selection bias, their ML technique is shown to scale well, with up to 3000 GW events in a set characterised by 6 GW event parameters $\boldsymbol{\theta}$ and 4 population hyperparameters $\boldsymbol{\lambda}$, producing 100 samples from $p(\boldsymbol{\theta}|\boldsymbol{\lambda})$ in ~ 0.1s. Given that their model performs just as efficiently as other analytic approaches in a fraction of the time, it is likely that normalizing flows will play a crucial role in testing different state of the art models and even families of models in future GW event catalogue releases. As is the case with most ML approaches, more data and greater ML model complexity may also improve results. We will now discuss in the next section how ML has been used for the purposes of identifying and characterising non-astrophysical noise transients and glitches.

3.4 Machine Learning for Detector Characterisation

Non-astrophysical noise sources can affect the data quality of the detectors adversely. Poor data quality can not only be a source of confusion when doing data analysis, but can even entirely corrupt segments of data. It is the job of detector characterisation experts to identify and mitigate such noise disturbances. If we are able to identify and prioritize classes of glitches (non-astrophysical transients), this is an important first step in mitigating such events. In this subsection I will describe some recent efforts to use ML to identify and mitigate noise in the ground based GW detectors.

In order to better classify known and unknown classes of glitches, Zevin et al. [86] use a unique combination of human learning and ML in a pipeline called `Gravity Spy` [250]. In

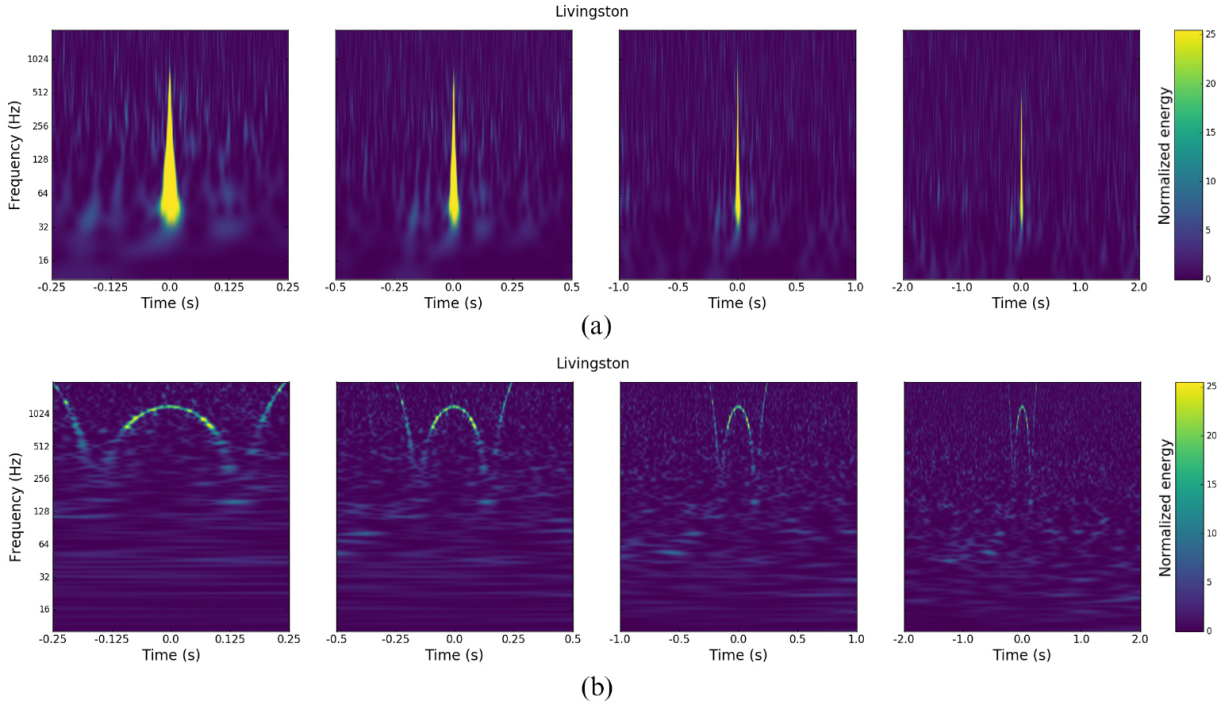


Figure 3.1: In this figure we see examples of two glitch types commonly identified by Gravity Spy. The top panels show a blip glitch in the time-frequency plane at 4 different time windows and is colorised by the normalised energy, which is essentially a measure of loudness. The bottom panels are an example of a whistle glitch, again at 4 different time windows. This figure was produced by the authors of [86].

`Gravity Spy` a set of glitch triggers (Ch. 1, Sec. 1.4.2) are chosen for training which occur in "lock" (meaning the detector was in a proper state to search for GWs, i.e. a state when it's sensitive to GWs). The glitches themselves are represented through Omega scans [251], which essentially search over sine-Gaussian waveform templates characterised by Q (quality) factor. Each template is represented as a time-frequency tiling and the Omega Scan searches for templates which most closely match a given piece of data where a good match is defined as having a high SNR value of the data with the template. A time-frequency spectrogram is finally generated for the most significant tile of the best matching template. See Fig. 3.1 for an example of some glitches represented as Omega Scans.

Their initial training set was generated from a set of 10^5 Omicron [252] glitch triggers, from which about 100 glitches per class were identified by eye from trained detector characterization experts. Approximately 20 classes were identified, where each were characterised by their corresponding waveform morphology. These 100×20 "gold-standard" glitches were then used to

train a preliminary neural network (a basic deep CNN model (Ch. 2)) to identify the remaining glitches in the rest of the 10^5 Omicron triggers. These triggers are then uploaded to the Zooniverse website [250] where volunteer citizen scientists are able to try classifying the trigger spectrograms by eye. Volunteers are initially trained on "gold standard" glitch images and other well known glitches which have a high probability of belonging to a specific class. After successive successful classifications by volunteers, the difficulty of glitches presented to the volunteers is ramped up. If a glitch is identified enough times by both the ML algorithm and volunteers to a high degree of confidence, it is then "retired" whereby it is taken out of the volunteer glitch lookup pool and then added to the ML labeled training set to improve the accuracy of the ML model.

Overall, `Gravity Spy` has been an incredibly successful use of ML and a unique example of use citizen scientists for great gain. In addition, `Gravity Spy` is an excellent example of outreach which energises the general public to get more interested in GW astronomy. Citizen scientists were able to identify more than $45,000$ glitches and several new glitch classes such as "Paired Doves" and "Helix" glitches. The "Paired Doves" class was a particularly useful identification as it closely resembled that of a company binary inspiral signals, which means that it was a particularly problematic glitch type, since it mimics the general structure of a known GW waveform.

One of the downsides of many ML algorithms (and a topic this author is personally very interested in) is the idea of interpretability. Many ML models such as CNNs, GANs, CVAEs, etc., while powerful, are composed of millions of model parameters which interact in complicated non-linear ways. Trying to interpret why a particular network configuration works over another is an active area of ongoing research in the ML community. Genetic algorithms are unique in that they are more easily interpretable than many other approaches.

A genetic algorithm operates by first generating an initial population of programs. Each program may be parameterised as a mathematical function whereby operands/variables and operators of the function are learned during training. Variables are representative of different types of data input to the algorithm. The program functions themselves are commonly represented as syntax trees, which are similar to decision trees [253], where the GP syntax tree is made up

of three components: nodes, branches and leaves. Nodes are representative of tests on a given input, branches are the outcomes of those tests, and leaves are the final predictions for a given input to the tree.

Each program/function may then be used to produce a prediction on a given input. The accuracy of the program's prediction with respect to the true value associated with the given input is quantified as the fitness score. Predictions from the program can be in the form of either a regression or a classification. A new generation of programs is then composed based on the fitness scores of all current programs, where the higher the fitness score, the more likely that program will be used to produce the next generation of programs. New programs are primarily produced through 2 avenues: reproduction and crossover mutation. Crossover mutation finds programs above a user predefined fitness score and then pairs them off with another chosen program above a predefined fitness to produce offspring programs to be used in the next generation (paired off programs are known as parents). Offspring are produced by choosing a crossover point (usually randomly) in each parent syntax tree, which essentially equates to a sub-portion of each parent tree (could also just be one single operator/operand). The two sub-portions of each parent tree are then swapped between the parent trees to produce 2 new offspring trees. These offspring trees are then used as part of the next generation of programs. If the fitness score is high enough for a program in the current generation, one can also simply copy that program to be used in the next generation; this is known as reproduction. For further details on training and the operation of genetic programs, see [254].

Genetic algorithms were first used in the LVC by Cavaglia et al. in [255]. This was done by training a genetic algorithm over many auxiliary channels in the detector, where an auxiliary channel is defined as a monitor or data channel which measures the internal state of the instrument, or the physical environment surrounding the detectors. Once trained, their genetic algorithm produces a final set of functions where each variable in the function is representative of different auxiliary data channels. They train their algorithm using periods of observing data where there were known glitch classes (for comparison, a random forest algorithm was also run over the same dataset). They tested their algorithm on two glitch classes from both the first observing run and the second LVC observing run [7]. The first set are magnetometer glitches from

O2 and the second set or those from an air compressor seen in O1. Their training set consists of 2000 Omicron triggers (greater than SNR 5.5) and 749 auxiliary channels for the magnetometer glitches and 16 triggers and 429 auxiliary channels for the compressor glitches. When testing, the authors find that the genetic algorithm determines that 7/10 (9/10 for the random forest algorithm) of the most likely auxiliary channels for the magnetometer glitches result from the correct magnetometer LIGO detector subsystem (as identified by detector characterisation experts). The most likely auxiliary channel for a given glitch type was determined by counting the number of times a variable associated with a particular auxiliary channel was seen across the final multivariate expressions produced by the genetic algorithm. Both the genetic programming and the random forest algorithms were shown to generally be able to identify the source of the most likely auxiliary channels for both the air compressor glitches and magnetometer glitches.

What the work of Cavaglia et al. [255] shows is that both random forest (RF) and genetic programming (GP) have the ability to work well with complex GW channel data to identify complicated relationships between GW subsystems in a human readable fashion. The clear advantage of using such algorithms is in their interpretability, where in genetic algorithms one can look at the multivariate expressions chosen at each generation to see how the population was evolved exactly. The difficult to interpret internal nature of many deep ML algorithms is an interesting area of research not only in ML, but also in ML for GW astronomy. Going forward, this author believes that more work will need to be done in order to understand what features ML algorithms determine to be most important and in what instances an ML algorithm may be fooled into returning false positives, which this author believes algorithms like GP may have an important role to play.

We also note that ML has been used in a wide variety of other detector characterisation tasks including: transient classification using difference boosting networks [256], noise subtraction and denoising [257, 258], glitch classification using random forests, ANNs and support vector machines [259], and many other studies not listed here. In the next section we will summarise the topics covered and provide some final concluding thoughts on the state of ML in GW astronomy.

3.5 Summary

In this chapter we have provided an overview of recent advances in ML applications for GW astronomy. In the first section, we discussed how ML is being used for the detection of GWs from several sources including: BBH, NSBH, BNS, burst and CW searches. Algorithms used in these studies range from simple algorithms such as ANNs and CNNs, to more complex algorithms such as normalising flows and CVAEs. It was found in most of these studies that ML approaches are able to match the sensitivity of existing detection techniques. We also discussed recent work in ML for population inference, as well as ML for Bayesian parameter estimation. It was shown that algorithms developed by Gabbard et al., Chua et al. and Green et al. were able to reproduce Bayesian posteriors for CBC signals detected by both ground-based detectors (Gabbard et al.,Green et al.) and space-based detectors (Chua et al.). In all approaches, it was found that ML methods were faster when compared to existing inference techniques.

We additionally showed how ML is being used for detector characterisation to both classify non-astrophysical noise transients and mitigate such sources. We discussed the classification algorithm (`Gravity Spy`) and how it has been used in the LVK for several years and has been widely successful in classifying both known glitches (e.g. “Scattering”, “whistle”, etc.), as well as identified new glitch types (e.g. “Helix”, “Paired Doves”, etc.). We also described work by Cavaglia et al. which showed how interpretable algorithms, such as genetic algorithms, may be used for noise source hunting.

Given all of the many advances using ML mentioned in this chapter across a wide variety of domains in GW astronomy, it is clear to this author that ML will have a key role to play going forward. One of the primary benefits discussed many times throughout this chapter is that of speed. In the subsequent chapters we will show how we have used two ML methods, CNNs and CVAEs in order to tackle two outstanding problems in GW data analysis, low-latency signal detection and low-latency Bayesian parameter estimation. In our work we will show how these methods can not only be used in low-latency, but can also match the efficiency of existing techniques.

Chapter 4

Matching matched filtering using deep networks for gravitational wave astronomy

We note to the reader that this text is a modified version of the published paper here [1]. Modifications include: the addition of several diagnostic plots and some textual expansion on the background of techniques used.

In brief: we report in this chapter on the construction of a deep convolutional neural network that can reproduce the sensitivity of a matched-filtering search for binary black hole gravitational-wave signals. The standard method for the detection of well-modeled transient gravitational-wave signals is matched filtering. We use only whitened time series of measured gravitational-wave strain as an input, and we train and test on simulated binary black hole signals in synthetic Gaussian noise representative of Advanced LIGO sensitivity. We show that our network can classify signal from noise with a performance that emulates that of match filtering applied to the same datasets when considering the sensitivity defined by receiver-operator characteristics.

4.1 Introduction

The field of GW astronomy has seen an explosion of CBC detections over the past several years. The first of these were BBH detections [3, 4, 5, 7, 260], then the first detection of a binary

neutron star system [9], as well as the first NSBH detections [15]. This BNS event was seen in conjunction with a gamma-ray burst [10, 261, 262] and multiple post-merger electromagnetic signatures [21]. These detections were made possible by the detectors of the LVK. Over the coming years many more such observations, including BBHs, BNSs, NSBHs, as well as other more exotic sources are likely to be observed on a more frequent basis. As such, the need for more efficient search methods will be more pertinent as the detectors increase in sensitivity.

The algorithms used by the search pipelines to make detections of CBC signals [6, 263, 264] are, in general, computationally expensive. The methods used are complex, sophisticated processes computed over a large parameter space using advanced signal processing techniques. The computational cost to run the search analysis is due to the large parameter space, as well as analysis of the high frequency components of the waveform where high data sample rates are required. Distinguishing noise from signal in these search pipelines is achieved, in part, using a technique known as template based matched-filtering (Sec. 1.6.2).

Matched-filtering uses a bank [103, 104, 105, 264, 265] of template waveforms [71, 94, 106, 266] each with different component mass and/or spin values. A template bank will span a large astrophysical parameter space since we do not know a priori the true GWs parameter values. Waveform models that cover the inspiral, merger, and ringdown phases of a compact binary coalescence are based on combining post-Newtonian theory [69, 70, 71, 72], the effective-one-body formalism [73], and numerical relativity simulations [76].

Deep learning is a subset of machine learning which has gained in popularity in recent years [163, 209, 219, 267, 268, 269] with the rapid development of GPU technology. Some successful implementations of deep learning include image processing [163, 270, 271], medical diagnosis [272], and microarray gene expression classification [273]. There has also been success in the field of gravitational-wave astronomy in the form of glitch classification [86, 256, 274] and notably for signal identification [212, 275] where it was first shown that deep learning could be a detection tool [275]. See Ch. 3 for an overview of other studies where ML is being used for GW astronomy. Deep learning is able to perform analyses rapidly since the method's computationally intensive stage is pre-computed during the training prior to the analysis of actual data [162]. This could result in low-latency searches that have the potential to be orders of

magnitude faster than other comparable classification methods.

A deep learning algorithm is composed of stacked arrays of processing units, called neurons, which can be from one to several layers deep. A neuron acts as a filter, whereby it performs a transformation on an array of inputs. This transformation is a linear operation between the input array and the weight and bias parameters associated to the neuron. The resulting array is then typically passed to a non-linear activation function to constrain the neuron output to be within a set range. Deep learning algorithms typically consist of an input layer, followed by one to several hidden layers and then one to multiple output neurons. The scalars produced from the output neurons can be used to solve classification problems, where each output neuron corresponds to the probability that an input sample is of a certain class. We refer the reader back to Ch. 2 for a more in-depth description of machine learning principals/concepts.

In this chapter we investigate the simplest case of establishing whether a signal is present in the data or if the data contains only detector noise. We propose a deep learning procedure requiring only the data time series as input with minimal signal pre-processing. We compare the results of our network with the widely used matched-filtering technique. We show how a deep learning approach can be pre-trained using simulated datasets and applied in low-latency to achieve the same sensitivity as established techniques.

4.2 Simulation Details

In order to make a clean comparison between deep learning approach and matched-filtering, we distinguish between two cases, BBH merger signals in additive Gaussian noise (signal+noise) and Gaussian noise alone (noise-only). We choose to focus on BBH signals rather than including binary neutron star systems for the reason that BBH systems are higher mass systems and have shorter duration signals once the inspiralling systems have entered the Advanced detector frequency band. They typically then merge on the timescale of $\mathcal{O}(1)$s allowing us to use relatively small datasets for this study.

The input datasets consist of "whitened" simulated GW timeseries where the whitening process (see Sec. 1.6.2 of Ch. 1) uses the detector noise PSD (see Sec. 1.4.2 of Ch. 1) to rescale the

noise contribution at each frequency to have equal power. Our noise is initially generated from a PSD equivalent to the Advanced LIGO design sensitivity [276].

Signals are simulated using a library of GW data analysis routines called `LALSuite`. We use the IMRPhenomD type waveform [74, 75] which models the inspiral, merger and ringdown components of BBH GW signals. We simulate systems with component black hole masses in the range from $5M_{\odot}$ to $95M_{\odot}$, $m_1 > m_2$, with zero spin. Training, validation and testing datasets contain signals drawn from an astrophysically motivated distribution where we assume $m_{1,2} \sim \log m_{1,2}$ [277][1]. Each signal is given a random right ascension and declination assuming an isotropic prior on the sky, the polarization angle and phase are drawn from a uniform prior on the range $[0, 2\pi]$, and the inclination angle is drawn such that the cosine of inclination is uniform on the range $[-1, 1]$. The waveforms are then randomly placed within the time series such that the peak amplitude of each waveform is uniformly randomly positioned within the fractional range $[0.75, 0.95]$ of the timeseries.

The waveform amplitude is scaled to achieve a predefined optimal SNR defined as

$$\rho^2_{\mathrm{opt}} = 4 \int_{f_{\mathrm{min}}}^{\infty} \frac{|h(\tilde{f})|^2}{S_{\mathrm{n}}(f)} df, \tag{4.1}$$

where $h(\tilde{f})$ is the frequency domain representation of the GW strain and $S_{\mathrm{n}}(f)$ is the single-sided detector noise PSD [97] (Sec. 1.4.2 of Ch. 1). The simulated time series were chosen to be 1 s in duration sampled at 8192 Hz. Therefore we consider f_{min} as the frequency of the GW signal at the start of the sample timeseries hence each signal will have a mass dependent and merger time dependent minimum frequency. An example timeseries can be seen in Fig. 4.1.

Due to the requirements of the matched-filtering comparison it was necessary to add padding to the edges of each timeseries in the time domain so as to avoid non-physical boundary artefacts from the whitening procedure[2]. The Gaussianity of the noise and smoothness of the simulated advanced LIGO PSD allows the use of relatively short padding. Therefore each 1 s timeseries has an additional 0.5 s of data prior to and after the signal. The signal itself has a Tukey win-

[1] We note here that the inferred mass distribution from Observation run 3 does in fact differ from this distribution, but is not known at the time of writing this chapter.

[2] We note that for matched-filtering, whitening is typically done over a far longer segment.

dow ($\alpha = 1/8$) applied to truncate the signal content to the central 1 s, where the window is constructed such that it has no attenuation within the central 1 s. The CNN approach only has access to this central 1 s of data. Similarly, the optimal SNR is computed considering only the central 1 s.

Supervised deep learning requires datasets to be sub-divided into training, validation, and testing sets. Training sets are the data samples that the network learns from, the validation set allows the developer to verify that the network is learning correctly, and the test set is used to quantify the performance of the trained network. In a practical scenario the training and validation sets are used to train the network prior to data taking (See Ch. 2 for further details). This constitutes the vast majority of computational effort and is a procedure that needs to be computed only once. The trained network can then be applied to test data at a vastly reduced cost in comparison to the training stage [162]. Of the dataset generated, we use 90% of these samples for training, 5% for validation, and 5% for testing. A dataset was generated for each predefined optimal SNR value ranging from 1–10 in integer steps.

Our training datasets contain 5×10^5 independent timeseries with 50% containing signal+noise and 50% noise-only. For each simulated gravitational-wave signal (drawn from the signal parameter space) we generate 25 independent noise realizations from which 25 signal+noise samples are produced. This procedure is standard within machine learning classification and allows the network to learn how to identify individual signals under different noise scenario (See [278] and Ch. 2). Each noise-only sample consists of an independent noise realization and in total we therefore use 10000 unique waveforms in the m_1, m_2 mass space. Each data sample timeseries is then represented in the form of a 1×8192 pixel image with the gray-scale intensity of each pixel proportional to the measured GW strain.

4.3 The Deep Network Approach

In our model, we use a variant of a deep learning algorithm called a CNN [189] composed of multiple layers (See Ch. 2). The input layer holds the raw pixel values of the sample image which, in our case, is a 1-dimensional timeseries vector. The convolutional filters of the network

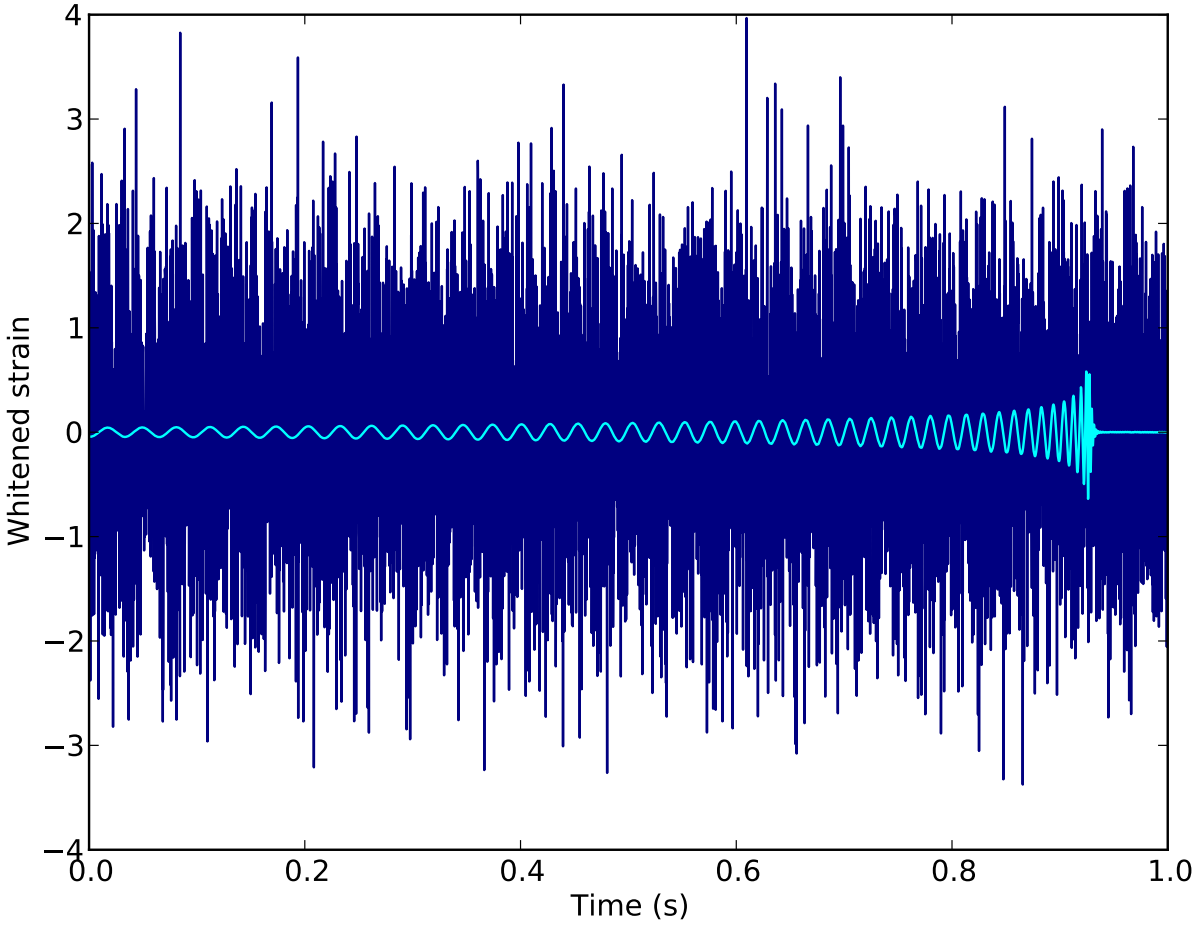


Figure 4.1: A whitened noise-free timeseries of a BBH signal sampled at 8192 Hz with component masses $m_1 = 41.86\mathrm{M}_\odot$ and $m_2 = 6.65\mathrm{M}_\odot$ with optimal SNR $= 8$ (cyan). The dark blue timeseries shows the same gravitational-wave signal with additive whitened Gaussian noise of unit variance. This latter timeseries is representative of the datasets used to train, validate, and test the deep neural network.

are also in 1-dimensional vector form. This is opposed to the traditional 2-dimensional form more commonly used by ML practitioners when applying CNNs towards 2-dimensional image analysis. We do not use a 2-dimensional form because our inputs are 1-dimensional, thus we would ideally like our CNN filters to reflect that reality. Each neuron in the convolutional layer computes the convolution between the neuron's weight vector and the outputs from the layer below it, and then the result is summed with the bias vector. Neuron weight vectors are updated through an optimisation algorithm called back-propagation [165]. Activation functions apply an element-wise non-linear operation rescaling their inputs onto a specific range and leaving the size of the previous layer's output unchanged. Pooling layers perform a downsampling operation along the spatial dimensions of their input. Finally we have a hidden layer connected to an output layer which computes the inferred class probabilities. These values are input to a loss function, chosen as the binary cross-entropy [279], defined as

$$f(p,t) = -\sum_{i\in \mathrm{S}} t_i^{\mathrm{S}} \log(p_i^{\mathrm{S}}) - \sum_{i\in \mathrm{N}} t_i^{\mathrm{N}} \log(p_i^{\mathrm{N}}), \tag{4.2}$$

where $p_i^{\mathrm{S/N}}$ is the predicted probability of the class signal+noise (S) or noise-only (N) and $t_i^{\mathrm{S/N}}$ is the true value for the i'th training sample. The loss function is minimised when input data samples are assigned the correct class with the highest confidence.

In order to optimise a network, multiple hyper-parameters must be tuned. We define hyper-parameters as parameters that we are free to choose. Such parameters include the number and type of network layers, the number of neurons within each layer, the size of the neuron weight vectors, the max-pooling parameters, the type of activation functions, the preprocessing of input data, the learning rate, and the application (or otherwise) of specific deep learning techniques. We begin the process with the simplest network that provides a discernible level of effective classification. In most cases this consists of an input, convolutional, hidden, and logistic output layer. The optimal network structure was determined through multiple tests and tunings of hyperparameters by means of trial and error[3].

[3] We have also used multiple other hyperparameter optimisation techniques, as introduced in Ch. 2, in subsequent projects. We tried some of these optimisation schemes in Ch. 4, but in the end settled on a network design which was determined through random trial and error.

Within our optimisation process we experimented with rescaling the input data, which we found to have minimal effect on the network performance. The reason for this is that our input data is whitened and our signals are buried beneath the noise. Therefore our data is effectively prescaled on a range $\pm\mathcal{O}(1)$ due to the natural variation of Gaussian noise. We also experimented with using transfer learning [280] where networks pre-trained on high SNR datasets are used as starting points for application to successively lower SNR datasets. We found that there were no performance benefits in using this approach compared to training the network solely on each SNR dataset separately. The network depth was adjusted between 2 and 10 convolutional layers. The inclusion of dropout (See Sec. 2.3.1 of Ch. 2) was used within the final 2 hidden layers as a form of regularisation to avoid overfitting.

During the training stage an optimisation function (back-propagation) works by computing the gradient of the loss function (Eq. 4.2) with respect to the weights of the network for a given training sample, then attempting to minimize that loss function. The value of the loss is propagated back through the network by taking the partial derivative of the loss with respect to the weights of the network and applying the chain rule. Using the calculated partial derivatives, one can then update the weight and bias terms of the network such that the loss is minimised. Back propagation is done over multiple iterations where at each iteration the gradients are computed all at once over a batch of training samples. We also use Nesterov momentum [281], which is described by

$$v_i = \mu v_{i-1} - \eta \nabla f(\theta_{i-1} + \mu v_{i-1}), \tag{4.3}$$

$$\theta_i = \theta_{i-1} + v_i, \tag{4.4}$$

where θ_{i-1} are the parameters of the neural network from the previous layer, θ_i are the parameters of the current layer of the network and v_i is the momentum for the current layer. We also have μ which is a scalar constant term which determines the amount of momentum to apply per gradient update (the higher the number, the more momentum), v_{i-1} is the momentum term from the previous layer, η is the learning rate, $\nabla f(\theta_{i-1} + \mu v_{i-1})$ is the gradient of the model parameters with respect to the previous layer (including the momentum term from the previous

layer (μv_{i-1})). There are a variety of initialisation schemes for the momentum term in each layer, but prior to training, the momentum in each layer is nominally chosen from a uniform distribution between 0 and 1. We use a learning rate which alternates between $\eta = 10^{-3}$ and $\eta = 5 \times 10^{-5}$, and the Adam optimiser [282] which is parameterised by a set of user-defined hyperparameters given as: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$ and a momentum schedule of 0.004 (See [282] for further details on the function of these Adam hyperparameter values). We outline the structure of the final neural network architecture in Table 4.1. We also note that the same network structure was used for each SNR, but that a separate neural network was trained for every SNR value.

The final ranking statistic that we extract from the CNN analysis is taken from the output layer, composed of 2 neurons, where each neuron gives the inferred probability that the input data belongs to the noise or signal+noise class respectively. Both neurons will produce a probability value between 0 and 1 with their sum being unity, which is the default behaviour of the softmax activation function given by

$$\text{Softmax}(\text{x}_\text{i}) = \frac{e^{x_i}}{\sum_j e^{x_j}}, \tag{4.5}$$

where e^{x_i} is the exponent of the value of one of the 2 output layer neurons for class i and $\sum_j e^{x_j}$ is a summation over the exponent of both neuron output layer values, representing the predictions for all classes. The computational time spent on training the network for each SNR is $\mathcal{O}(1)$ hour on a single GPU[4]. This one-time cost can be compared to the $\mathcal{O}(1s)$ spent applying the trained network to all $25,000$ 1 s test data samples also using a single GPU. Therefore at the point of data taking this particular analysis can be run at 10^4 times faster than real-time.

In Fig. 4.2, we plot three different quantities as a function of training epoch for one of our networks trained on SNR 8 signals. The top panel shows the loss as a function of training epoch, which we see decreases rapidly initially and then levels out as training progresses. In the middle panel, we show the detection probability (fraction of GW signals correctly identified

[4]It should be noted that this study was carried out in 2017 and as such GPUs have become more efficient since then. If the study were carried out today, it is possible that the speed of the network could be improved by some factor.

Parameter	Layer								
(Option)	1	2	3	4	5	6	7	8	9
Type	C	C	C	C	C	C	H	H	H
No. Neurons	8	8	16	16	32	32	64	64	2
Filter Size	64	32	32	16	16	16	n/a	n/a	n/a
MaxPool Size	n/a	8	n/a	6	n/a	4	n/a	n/a	n/a
Drop out	0	0	0	0	0	0	0.5	0.5	0
Act. Func.	Elu	Elu	Elu	Elu	Elu	Elu	Elu	Elu	SMax

Table 4.1: The optimised network consisting of 6 convolutional layers (C), followed by 3 hidden layers (H). Max-pooling is performed on the first, fifth, and eighth layer, whereas dropout is only performed on the two hidden layers. Each layer uses an exponential linear unit (Elu) activation function (with range $[-1,\infty]$) while the last layer uses a Softmax (SMax) activation function in order to normalize the output values to be between zero and one so as to give a probability value for each class.

by the neural network) as a function of training epoch where the training, validation and testing detection probability are denoted as the purple, blue and orange curves respectively. The goal is to minimize the loss function, which will in turn maximise the detection probability of the classifier. We see in Fig. 4.2 that as the loss decreases, the detection probability increases, indicating the the network is performing more accurately as training progresses. We should note here though that it does appear that the network is slightly overfitting to the training data. This is because the validation/testing detection probability curves (blue, orange) are lower than the training probability curve (purple). The bottom panel shows the learning rate used as a function of training epoch which alternates between 10^{-3} and 5×10^{-3}.

Our learning rate alternates between a minimum and upper bound because we decided to employ a cyclic learning rate scheduler [283]. A cyclic learning rate is commonly used avoid saddle points, or local minima, in the loss function parameter space. Specifically, a learning rate which is too low will only apply small updates to the neural network weights and thus may get stuck in that saddle point. A cyclic learning rate will allow both large and small network weight updates to occur, thus increasing the likelihood of breaking out of saddle points. There is also the issue of choosing a poor initial learning rate at the beginning of training. If our network and/or optimiser is strongly influenced by the initial learning rate, we may never see the loss function minimised. A cyclic learning rate allows us apply a variety of learning rate values over a broad range, thus minimising the chance of choosing an inappropriate learning rate.

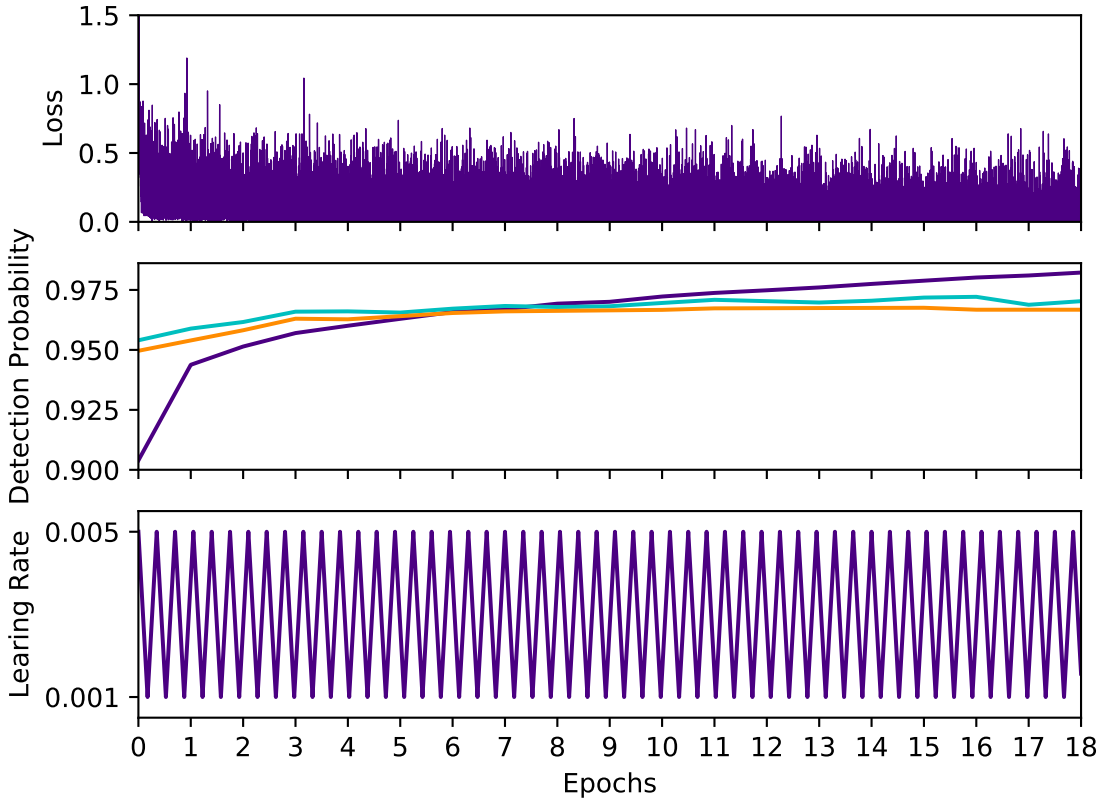


Figure 4.2: The loss, detection probability and learning rate plots (shown above top panel) for a network trained on SNR 8 signals illustrate how the network's performance is defined as a function of the number of training epochs. The first initial epochs see an exponential decrease in the loss function and then a slowly falling values to follow. This indicates that the longer our network is trained, a limit with respect to the accuracy is approached. In our case, we cyclically adjust the learning rate to alternate between 5×10^{-4} and 10^{-3} at a constant frequency. The detection probability on training samples is represented by the purple curve, validation detection probability as the blue, and testing detection probability as the orange curve.

4.4 Applying Matched-Filtering

In order to establish the power of the deep learning approach we must compare our results to the standard matched-filtering process used in the detection of CBC signals [97, 284]. The ranking statistic used in this case is the matched-filter SNR (Eq. 1.35 in Ch. 1) numerically maximized over arrival time and analytically maximised over phase and distance [51]. By first defining the noise weighted inner product as a function of a time shift Δt between a and b given as,

$$(a \mid b)(\Delta t) = 4 \int_{f_{\min}}^{\infty} \frac{\tilde{a}(f)\tilde{b}^*(f)}{S_{\rm n}(f)} e^{2\pi i f \Delta t} \, df, \tag{4.6}$$

where $\Delta t = 1/f_s$, f_s is the sampling frequency, and the exponential term $e^{2\pi i f \Delta t}$ allows us to use the FFT to compute the SNR for every discrete time shift between the data and template. We can construct the squared matched-filter SNR as

$$\rho^2 = \frac{\langle s,h\rangle^2 + \langle s,h\rangle^2}{\langle h,h\rangle}, \tag{4.7}$$

where s is the data containing noise and a potential signal, and h is the noise-free gravitational-wave template [99]. For a given template this quantity is efficiently computed using the FFT, where the FFT allows the SNR to be computed for all possible signal arrival times within the observation window with cost $N \log N$, where N is the number of samples we have[5]. The maximum SNR value from the output SNR timeseries, defined by Eq. 4.7, is also computed. The subsequent step is to further numerically maximize this quantity over a collection of component mass combinations. In this analysis a comprehensive template bank is generated in the m_1, m_2 mass space covering our predefined range of masses. We use a maximum mismatch of 3% and a lower frequency cutoff of 20 Hz using the PyCBC geometric non-spinning template bank generation tool [6, 285]. This template bank contained 8056 individual templates.

When generating an SNR timeseries (Eq. 4.7) for an input dataset we select $f_{\min}$ according to the conservative case in which the signal merger occurs at the 0.95 fraction of the 1 s timeseries. We therefore select only maximised SNR timeseries values recovered from within

[5] If we didn't use the FFT, then the brute force cost would N^2 (correlating the template once costs N operations and then shifting it by one bin and doing it again N times.

the $[0.75, 0.95]$ fractional range since this is the parameter space on which the CNN has been trained. For the practical computation of the matched-filtering analysis we take each of the data samples from the testing dataset to compute the matched-filter ranking statistic.

4.5 Matching Matched Filtering Results

After tuning the multiple hyper-parameters (Table 4.1) and training the neural network (Fig. 4.2), we present the results of our CNN classifier on a noise versus signal+noise sample set. With values of ranking statistics now assigned to each test data sample from both the CNN and matched-filtering approaches, and having knowledge of the true class associated with each sample, we may now construct ROC curves to compare performance.

A standard method for displaying the accuracy of a classifier is through a confusion matrix in which the number of samples of each true class identified as every possible class are listed in a square matrix. A fully diagonal matrix would imply no incorrectly classified samples and a uniform matrix would imply no classification power (with the stipulation that there are equal numbers of testing samples in each class). Classification, within the context of a confusion matrix, is defined as when the neural network predicts a value of greater than $p = 0.5$ for a single class. In Fig. 4.3 we show results for the CNN approach from which we highlight the overall accuracy (the ratio of incorrectly identified samples to the total number of samples) is $[51.88\%, 65.51\%, 85.63\%, 96.66\%, 99.41\%, 99.99\%]$ at $\rho_{\text{opt}} = [2, 4, 6, 8, 10, 12]$, for a fixed detection threshold of $p = 0.5$. We highlight that the measure of CNN accuracy illustrated by Fig. 4.3 is interesting in general, but not useful for us in practice since we want to maximise the TAP (fraction of signal samples correctly identified) at a fixed FAP (fraction of noise samples incorrectly identified as signals) value.

In Fig. 4.4 we compare our CNN results to that of matched-filtering. Given the ranking statistic from a particular analysis and defining a parametric threshold value on that statistic we are able to plot the FAP) versus the TAP. These curves are defined as ROC curves and a ranking statistic is deemed superior to another if at a given FAP it achieves a higher detection probability. Our results show that the CNN approach closely matches the sensitivity of matched-filtering for

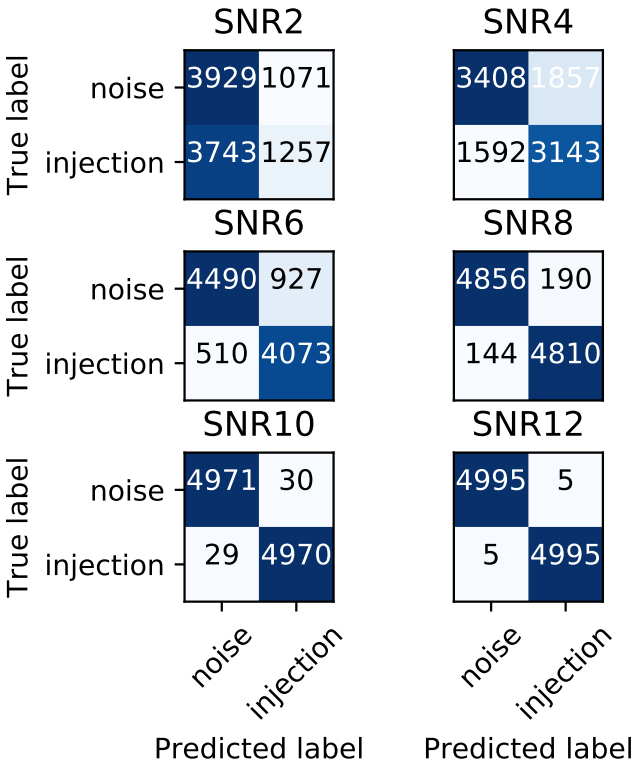


Figure 4.3: Confusion matrices for testing datasets containing signals with optimal SNR (Given in Eq. 4.1) $\rho_{\text{opt}} = 2, 4, 6, 8, 10, 12$. Numerical values superimposed within matrix elements represent the number of samples that were of true class indicated by the y-axis label but identified as the corresponding x-axis label. For our 2 class system these are equivalent to the numbers of true alarm, true dismissal, false dismissal, or false alarm, for a fixed detection threshold of $p = 0.5$. The accuracy percentages for all injection SNR values are listed as follows: 51.86% at $\rho_{\text{opt}} = 2$, 65.51% at $\rho_{\text{opt}} = 4$, 85.63% at $\rho_{\text{opt}} = 6$, 96.66% at $\rho_{\text{opt}} = 8$, 99.41% at $\rho_{\text{opt}} = 10$ and 99.99% at $\rho_{\text{opt}} = 12$.

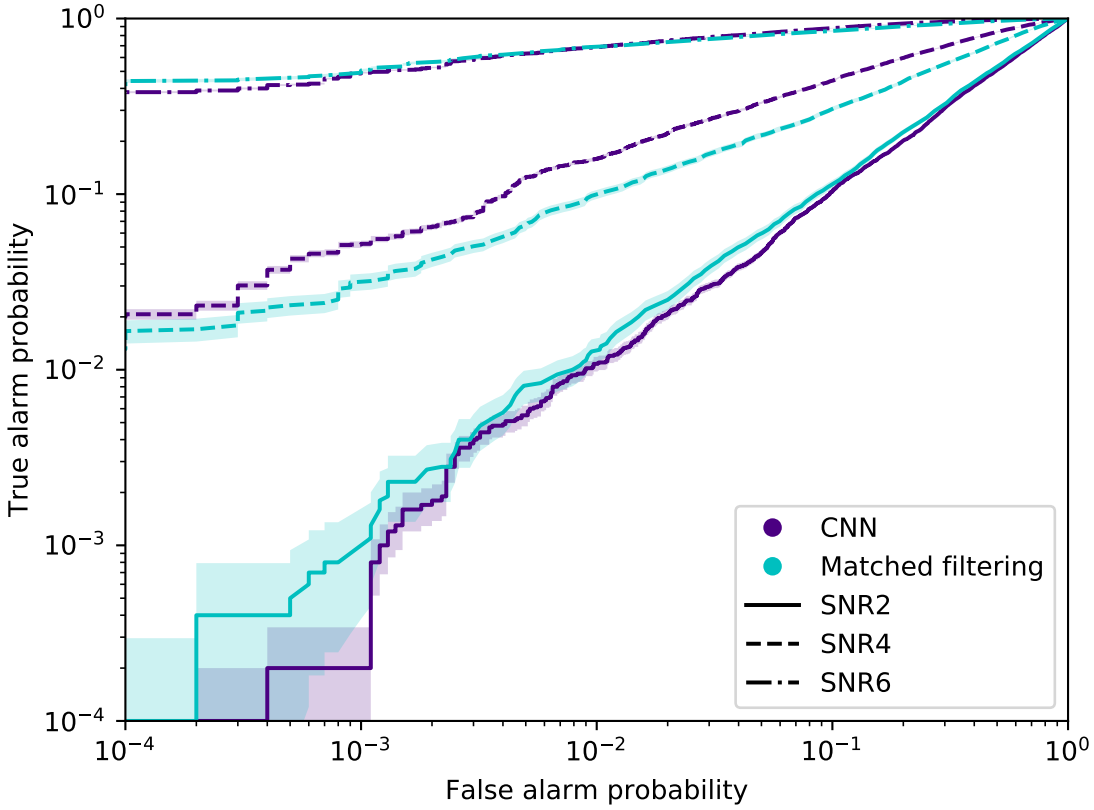


Figure 4.4: The ROC curves for test datasets containing signals with optimal SNR, $\rho_{\text{opt}} = 2, 4, 6$. We plot the true alarm probability versus the false alarm probability estimated from the output of the CNN (purple) and matched-filtering (cyan) approaches. Uncertainties in the true alarm probability correspond to 1-σ bounds assuming a binomial distribution. We note that because the results from the SNR 2 analyses follow a diagonal line from the bottom-left corner of the plot to the upper-right corner, that the predictions from both the machine learning approach and the matched filtering approach are essentially equivalent to random guessing.

all test datasets across the range of FAPs explored in this analysis[6]. It can clearly be seen that our classifier also exceeds the performance of the matched-filtering method at optimal SNR $\rho_{\text{opt}} = 2, 4, 6$. This is an interesting result and is not entirely unexpected given that matched-filtering is not expected to be completely optimal [203, 204].

We can make an additional direct comparison between approaches by fixing a FAP and plotting the corresponding TAP versus the optimal SNR of the signals in each test dataset. We show these efficiency curves in Fig. 4.5 at FAPs $10^{-1}, 10^{-2}, 10^{-3}$ for both the CNN and matched-

[6]We are limited to a minimal FAP of $\sim 10^{-4}$ due to the limited number of testing samples used.

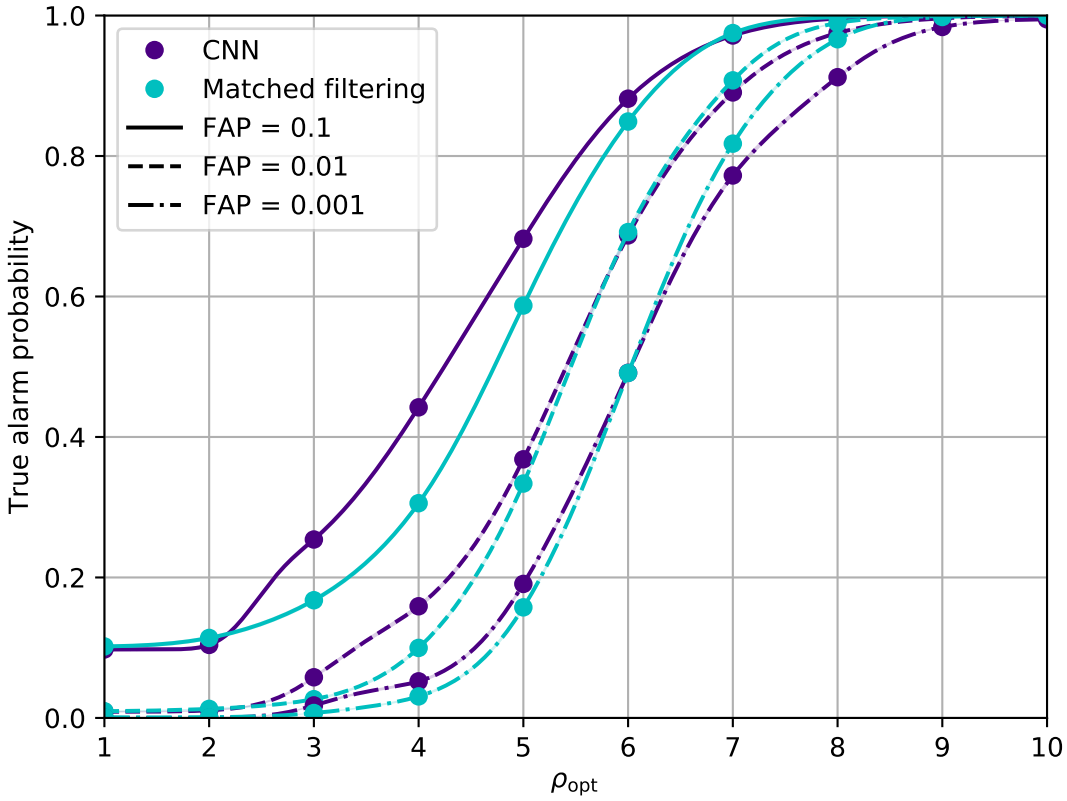


Figure 4.5: Efficiency curves comparing the performance of the CNN and matched-filtering approaches for false alarm probabilities 10^{-1} (solid), 10^{-2} (dashed), and 10^{-3} (dot-dashed). The true alarm probability is plotted as a function of the optimal SNR for the CNN (purple) and the matched-filtering (cyan) analyses. Solid dots indicate at which SNR values analyses were performed and light shaded areas are representative of the statistical uncertainties in the curves (which are all smaller than the line thicknesses).

filtering approaches. We again see very good agreement between the approaches at all FAPs with the CNN sensitivity exceeding that of the matched-filter approach at low SNR and high FAP. Conversely we see the matched-filter sensitivity marginally exceeds the CNN at high SNR and low false alarm probability. This latter discrepancy could be mitigated by increasing the number of training samples in the CNN approach.

We acknowledge that because we do not include a trials factor which takes into account the outputs of all neural networks used for each SNR value the FAP would be underestimated if one were to apply our method to a realistic search scenario. There are multiple methods we could have employed in order to take this into account. First, we could have used the median SNR

network (SNR 6) and then applied it to the whole range of SNR testing sets, thus avoiding the additional trials factors. However, it is unclear how well the network would then be able to generalise to lower and higher SNR values. Secondly, one could exploit techniques such as ensemble learning whereby the outputs from all networks could be given as input to another network in order to produce a final detection statistic. We note that many other practitioners in the ML community commonly employ such methods in order to optimise their approaches [286, 287]. Finally, the most optimal method to use would be to simply use a single neural network which has been trained over a range of SNR values that we would expect to see in the data. This specific method has already been used in a wide variety of GW signal detection applications and has been shown to generalise well to a large range of SNR values [288, 289, 290]. We further note that including a range of SNRs in the training set, rather than just 1 SNR value is essentially equivalent to adding an additional dimension in the parameter space (i.e. luminosity distance). In fact, it has already been shown in this chapter that our multiple SNR neural networks show no signs of bias across the full prior space for a range of source parameters including sky position, inclination angle, etc. This has also been shown to be the case for our own work in Bayesian parameter estimation [2], as well as in ML signal detection for other source types (e.g. CWs [221, 225]).

4.6 Conclusions

We have demonstrated that deep learning, when applied to GW timeseries data, is able to closely reproduce the results of a matched-filtering analysis in Gaussian noise. We employ a deep convolutional neural network with rigorously tuned hyperparameters and produce an output that returns a ranking statistic interpreted as the inferred probability that data contains a signal. Matched-filtering analyses are often described as the optimal approach for signal detection in Gaussian noise, but in reality are only close to optimal [203, 204]. By building a neural network that is capable of matching the efficiency of matched filtering we answer a fundamental question regarding the applicability of neural networks for GW data analysis.

In practice, searches for transient signals in GW data are strongly affected by non-Gaussian

noise artefacts. To account for this, standard matched-filtering approaches are modified to include carefully chosen changes to the ranking statistic [109, 291] together with the excision of poor quality data [292, 293]. Our analysis represents a starting point from which a deep network can be trained on realistic non-Gaussian data. Since the claim of matched-filtering near optimality is applicable only in the Gaussian noise case, there exists the potential for deep networks to exceed the sensitivity of existing matched-filtering approaches in real data.

We should also note that one of the downsides of our approach has been the use of separate neural networks for each SNR value. This could easily be overcome by simply training a single CNN model using a training set which contains GW waveforms with multiple SNR values. In fact, it was shown in [221] that single networks can generalise well to different SNR values.

In this work we have presented results for BBH mergers, however, this method could be applied to other merger types, such as BNS (as was shown in [206, 208]) and NSBH signals. This supervised learning approach can also be extended to other well modelled GW targets such as the continuous emission from rapidly rotating non-axisymmetric NSs (as was done by [221, 225]). Moreover, unsupervised approaches have the potential to be powerful detection tools in searches for unmodelled burst-like GW signals, where it was shown in [201] that GANs could be used for such a task. Finally we mention the possibilities for parameter estimation [275] where in the simplest cases an output regression layer can return point estimates of parameter values. There have also been several other studies done since the publication of this work which have used various ML techniques to perform Bayesian parameter estimation in low-latency [2, 234, 294] As was exemplified in the case of GW170817 [9], rapid detection confidence coupled with robust and equally rapid parameter estimates is critical for GW multi-messenger astronomy.

Since the publication of this paper, a lot has changed in the field of ML for GW signal detection. Many of the outstanding problems we outlined here in the conclusions have now been largely solved, such as applying deep learning towards other signal types [201, 206, 208, 216, 233]. There has also been much work using other unique network architectures such as Bayesian neural networks [295] for CBC detection, as well as the early-warning detection of BNS signals using CNNs [296]. Following this paper, we decided to pivot towards the more challenging unsolved problem of ML for Bayesian parameter estimation. In the following chapter (Ch. 5)

we will show how we have used CVAEs for the purpose of generating rapid Bayesian posterior parameter estimates.

Chapter 5

Variational Inference for GW Parameter Estimation

We note to the reader that this text is a modified version of the paper [2] recently accepted for publication in Nature Physics.

So far, we have introduced fundamental concepts from GW astronomy and ML. We have also provided a broad survey of how ML is being applied across a variety of domains within GW astronomy. In the previous chapter (Ch. 4) we showed one of the first implementations of deep learning for GW signal detection and how our approach was able to match the sensitivities of standard methods, opening the door for a variety of follow-up studies listed in Ch. 3. We now move on to the more challenging task of applying ML methods towards GW Bayesian parameter estimation. We show for the first time that a form of ML, CVAEs, may be used to produce Bayesian posteriors of GW source parameter values given GW time series data in a fraction of the time taken by more traditional samplers.

5.1 Introduction

GW detection is now commonplace [9, 277] and as the sensitivity of the global network of GW detectors improves, we will observe $\mathcal{O}(100)$s of transient GW events per year [7, 18, 297]. The current methods used to estimate their source parameters employ optimally sensitive [298] but

computationally costly Bayesian inference approaches [20] where typical analyses have taken between 6 hours and 38 days [299] to run. We determined these values by compiling tables (Tab. 5.1 and Tab. 5.2) containing of all detected events during the O3 observing run using the GraceDB database. We provide in the tables the length of time to complete parameter estimation analyses using the lalinference pipeline [20], as well as the predicted source class using the `p-astro`[1] computing package.

For BNS and NSBH systems prompt counterpart EM signatures are expected on timescales of 1 s – 1 minute and the current fastest method for alerting EM follow-up observers [22], can provide estimates in $\mathcal{O}(1)$ minute, on a limited range of key source parameters. Here we show that a CVAE [164, 300] pre-trained on BBH signals can return Bayesian posterior probability estimates. The training procedure need only be performed once for a given prior parameter space and detector network configuration and the resulting trained machine can then generate samples describing the posterior distribution $\sim$ 6 orders of magnitude faster than existing techniques.

With the overwhelmingly successful observation runs of O1, O2 and now O3 complete, LIGO and Virgo have produced a large catalogue of GW data covering both BBH and BNS signals [5]. Over the next five years we expect the number of detections to increase to be upwards of $\sim$ 180 BNS and $\sim$ 400 BBH events per year [7, 18, 297]. This large influx in the number of detections will put an increased amount of pressure on the current computationally costly GW inference methods used for parameter estimation.

The problem of detecting GWs has largely been solved through the use of template based matched-filtering, a process recently replicated using machine learning techniques [1, 204, 212, 275]. Once a GW has been identified through this process, Bayesian inference, known to be the optimal approach [298], is used to extract information about the source parameters of the detected GW signal.

In the standard Bayesian GW inference approach (See Sec. 1.7 of Ch. 1), we assume a signal and noise model and both may have unknown parameters that we are either interested in inferring or prefer to marginalise away. Each parameter is given a prior astrophysically motivated probability distribution and in the GW case, we typically assume a Gaussian additive

[1] See https://pypi.org/project/p-astro/.

Table 5.1: O3 events table containing information on detected event parameter estimation runtimes and classification probability (according to GraceDB) from April 8, 2019 - September 10, 2019. The amount of time for a run to produce final parameter estimation results is given by the difference between the event time and the first reported parameter estimation results from lalinference. We do not show here any detection events flagged by GraceDB that were later retracted. Columns with a "-" are values which were either not reported by GraceDB, or could not be found.

Event Name	Classification	Event Time	First LAL Results	Results Delay
S190408an	BBH (> 99%)	April 8, 2019 18:18:02 UTC	2019-04-14 05:37:36 UTC	5 d, 11 hrs, 19 min, 34 s
S190412m	BBH (> 99%)	April 12, 2019 05:30:44 UTC	-	-
S190421ar	BBH (97%), Terrestrial (3%)	April 21, 2019 21:38:56 UTC	May 3, 2019 08:18:56 UTC	11 d, 10 hrs, 40 mins, 0 s
S190425z	BNS (> 99%)	April 25, 2019 08:18:05 UTC	Apr 26, 2019 11:02:22 UTC	1 d, 2 hrs, 44 mins, 17 s
S190426c	BNS (49%), MassGap (24%), Terrestrial (14%), NSBH (13%)	April 26, 2019 15:21:55 UTC	Apr 28, 2019 17:11:32 UTC	2 d, 1 hr, 49 min, 37 s
S190503bf	BBH (96%), MassGap (3%)	May 3, 2019 18:54:04 UTC	Jun 11, 2019 08:18:41 UTC	38 d, 13 hrs, 24 min, 37 s
S190510g	Terrestrial (58%), BNS (42%)	May 10, 2019 02:59:39 UTC	Jun 3, 2019 16:19:23 UTC	24 d, 13 hrs, 19 min, 44 s
S190512at	BBH (99%), Terrestrial (1%)	May 12, 2019 18:07:14 UTC	May 17, 2019 15:27:36 UTC	4 d, 21 hrs, 20 min, 22 s
S190513bm	BBH (94%), MassGap (5%)	May 13, 2019 20:54:28 UTC	May 16, 2019 14:43:37 UTC	2 d, 17 hrs, 49 min, 9 s
S190517h	BBH (98%), MassGap (2%)	May 17, 2019 05:51:01 UTC	May 21, 2019 15:22:31 UTC	4 d, 9 hrs, 31 min, 30 s
S190519bj	BBH (96%), Terrestrial (4%)	May 19, 2019 15:35:44 UTC	May 22, 2019 10:05:27 UTC	2 d, 18 hrs, 29 min, 43 s
S190521g	BBH (97%), Terrestrial (3%)	May 21, 2019 03:02:29 UTC	May 21, 2019 09:11:04 UTC	0 d, 6 hrs, 8 min, 35 s
S190521r	BBH (> 99%)	May 21, 2019 07:43:59 UTC	May 24, 2019 21:22:47 UTC	3 d, 13 hrs, 38 min, 48 s
S190602aq	BBH (99%)	June 2, 2019 17:59:27 UTC	Jun 7, 2019 14:21:51 UTC	4 d, 20 hrs, 22 min, 24 s
S190630ag	BBH (94%), MassGap (5%)	June 30, 2019 18:52:05 UTC	Jul 1, 2019 18:09:39 UTC	0 d, 23 hrs, 17 min, 34 s
S190701ah	BBH (93%), Terrestrial (7%)	July 1, 2019 20:33:06 UTC	Jul 3, 2019 02:08:14 UTC	1 d, 5 hrs, 35 min, 8 s
S190706ai	BBH (99%), Terrestrial (1%)	July 6, 2019 22:26:41 UTC	Jul 8, 2019 05:17:16 UTC	1 d, 6 hrs, 50 min, 35 s
S190707q	BBH (> 99%)	July 7, 2019 09:33:26 UTC	Jul 9, 2019 17:48:32 UTC	2 d, 8 hrs, 15 min, 6 s
S190718y	Terrestrial (98%), BNS (2%)	July 18, 2019 14:35:12 UTC	-	-
S190720a	BBH (99%), Terrestrial (1%)	July 20, 2019 00:08:36 UTC	Jul 21, 2019 13:54:03 UTC	1 d, 13 hrs, 45 min, 27 s
S190727h	BBH (92%), Terrestrial (5%), MassGap (3%)	July 27, 2019 06:03:33 UTC	Jul 31, 2019 20:08:10 UTC	4 d, 14 hrs, 4 min, 37 s
S190728q	BBH (95%), MassGap (5%)	July 28, 2019 06:45:10 UTC	Jul 30, 2019 10:32:36 UTC	2 d, 3 hrs, 47 min, 26 s
S190814bv	NSBH (> 99%)	Aug. 14, 2019 21:10:39 UTC	Aug 15, 2019 09:02:34 UTC	11 hrs, 51 min and 55 s
S190828j	BBH (> 99%)	Aug. 28, 2019 06:34:05 UTC	Aug 30, 2019 07:58:27 UTC	2 d, 1 hr, 24 min and 22 s
S190828l	BBH (> 99%)	Aug. 28, 2019 06:55:09 UTC	Aug 29, 2019 15:43:57 UTC	1 d, 8 hrs, 48 min and 48 s
S190901ap	BNS (86%), Terrestrial (14%)	Sept. 1, 2019 23:31:01 UTC	Sep 2, 2019 11:21:59 UTC	11 hrs, 50 min, 58 s
S190910d	NSBH (98%), Terrestrial (2%)	Sept. 10, 2019 01:26:19 UTC	Sep 10, 2019 22:33:57 UTC	21 hrs, 7 min, 38 s

Table 5.2: O3 events table containing information on detected event parameter estimation runtimes and classification probability (according to GraceDB) from September 10, 2019 - March 16, 2020. The amount of time for a run to produce final parameter estimation results is given by the difference between the event time and the first reported parameter estimation results from lalinference. We do not show here any detection events flagged by GraceDB that were later retracted. Columns with a "-" are values which were either not reported by GraceDB, or could not be found.

Event Name	Classification	Event Time	First LAL Results	Results Delay
S190910h	BNS (61%), Terrestrial (39%)	Sept. 10, 2019 08:29:58 UTC	Sep 11, 2019 17:11:34 UTC	1 d, 8 hrs, 41 min, 36 s
S190915ak	BBH (99%)	Sept. 15, 2019 23:57:02 UTC	2019-09-17 13:42:31 UTC	1 d, 13 hrs, 45 min, 29 s
S190923y	NSBH (68%), Terrestrial (32%)	Sept. 23, 2019 12:55:59 UTC	-	-
S190924h	MassGap (> 99%)	Sept. 24, 2019 02:18:46 UTC	2019-09-27 19:17:41 UTC	3 d, 16 hrs, 58 min, 55 s
S190930s	MassGap (95%), Terrestrial (5%)	Sept. 30, 2019 13:35:41 UTC	2019-10-04 19:25:05 UTC	4 d, 5 hrs 49 min, 24 s
S190930t	NSBH (74%), Terrestrial (26%)	Sept. 30, 2019 14:34:07 UTC	-	-
S191105e	BBH (95%), Terrestrial (5%)	Nov. 5, 2019 14:35:21 UTC	2019-11-12 12:51:56 UTC	6 d, 22 hrs, 16 min, 35 s
S191109d	BBH (> 99%)	Nov. 9, 2019 01:07:17 UTC	2019-11-10 14:31:40 UTC	1 d, 13 hrs, 24 min, 23 s
S191129u	BBH (> 99%)	Nov. 29, 2019 13:54:17 UTC	2019-12-05 13:54:17 UTC	6 d, 13 min, 48 s
S191204r	BBH (> 99%)	Dec. 4, 2019 17:15:26 UTC	-	-
S191205ah	NSBH (93%), Terrestrial (7%)	Dec. 5, 2019 21:52:08 UTC	-	-
S191213g	BNS (77%), Terrestrial (23%)	Dec. 13, 2019 04:34:08 UTC	-	-
S191215w	BBH (> 99%)	Dec. 15, 2019 22:30:52 UTC	2019-12-20 09:18:36 UTC	4 d, 10 hrs, 47 min, 44 s
S191216ap	BBH (99%)	Dec. 16, 2019 21:33:38 UTC	-	-
S191222n	BBH (> 99%)	Dec. 22, 2019 03:35:37 UTC	2019-12-22 22:06:36 UTC	18 hrs, 30 min, 59 s
S200105ae	Terrestrial (97%), NSBH (3%)	Jan. 5, 2020 16:24:26 UTC	2020-01-09 16:56:28 UTC	4 d, 32 min, 2 s
S200112r	BBH (> 99%)	Jan. 12, 2020 15:58:38 UTC	2020-01-14 15:54:35 UTC	1 d, 23 hrs, 55 min, 57 s
S200114f	-	Jan. 14, 2020 02:08:18 UTC	-	-
S200115j	MassGap (> 99%)	Jan. 15, 2020 04:23:09 UTC	2020-01-20 04:51:25 UTC	5 d, 28 min, 16 s
S200128d	BBH (97%), Terrestrial (3%)	Jan. 28, 2020 02:20:11 UTC	2020-01-30 09:35:52 UTC	2 d, 7 hrs, 15 min, 41 s
S200129m	BBH (> 99%)	Jan. 29, 2020 06:54:58 UTC	2020-02-03 00:08:54 UTC	4 d, 17 hrs, 13 min, 56 s
S200208q	BBH (99%)	Feb. 8, 2020 13:01:17 UTC	2020-02-10 21:01:12 UTC	2 d, 7 hrs, 59 min, 55 s
S200213t	BNS (63%), Terrestrial (37%)	Feb. 13, 2020 04:10:40 UTC	-	-
S200219ac	BBH (96%), Terrestrial (4%)	Feb. 19, 2020 09:44:15 UTC	-	-
S200224ca	BBH (> 99%)	Feb. 24, 2020 22:22:34 UTC	2020-02-26 13:49:38 UTC	1 d, 15 hrs, 27 min, 4 s
S200225q	BBH (96%), Terrestrial (4%)	Feb. 25, 2020 06:04:21 UTC	2020-02-26 07:42:24 UTC	1 d, 1 hrs, 38 min, 3 s
S200302c	BBH (89%), Terrestrial (11%)	March 2, 2020 01:58:11 UTC	2020-03-02 19:06:13 UTC	17 hrs, 8 min, 2 s
S200311bg	BBH (> 99%)	March 11, 2020 11:58:53 UTC	2020-03-13 04:55:55 UTC	1 d, 16 hrs, 57 min, 2 s
S200316bj	MassGap (> 99%)	March 16, 2020 21:57:56 UTC	2020-03-21 02:46:41 UTC	4 d, 4 hrs, 48 min, 45 s

noise model (in reality, the data is not truly Gaussian). Given a noisy GW waveform, we would like to find an optimal procedure for inferring some set of the unknown GW parameters. Such a procedure should be able to give us an accurate estimate of the parameters of our observed signal, whilst accounting for the uncertainty arising from the noise in the data.

According to Bayes' Theorem, a posterior probability distribution on a set of parameters, conditional on the measured data, can be represented as

$$p(x|y) \propto p(y|x)p(x), \tag{5.1}$$

where x are the parameters, y is the observed data, $p(x|y)$ is the posterior, $p(y|x)$ is the likelihood, and $p(x)$ is the prior on the parameters. The constant of proportionality, which we omit here, is $p(y)$, the probability of our data, known as the Bayesian evidence or the marginal likelihood. We typically ignore $p(y)$ since it is a constant and for parameter estimation purposes we are only interested in the shape of the posterior (See Sec. 1.7 of Ch. 1 for further details).

Due to the size, dimensionality and volume of the parameter space typically encountered in GW parameter estimation and the volume of data analysed, we must stochastically sample the parameter space in order to estimate the posterior. Sampling is done using a variety of techniques including Nested Sampling [157, 160, 161] and Markov chain Monte Carlo methods [153, 154]. The primary software tools used by the LIGO parameter estimation analysis are `LALInference` and `Bilby` [19, 20], which offer multiple sampling methods.

Machine learning has featured prominently in many areas of GW research over the last few years. These techniques have shown to be particularly promising in signal detection [1, 212, 275], glitch classification [86], earthquake prediction [301], and to augment existing Bayesian sampling methods [159]. We also highlight recent developments in GW parameter estimation (independent to this work) where one- and two-dimensional marginalised Bayesian posteriors are produced rapidly using neural networks [234], and where normalised flows in conjunction with CVAEs can reproduce Bayesian posteriors for a single GW detector case [236, 239]. These methods, including the one presented in this paper, are known as "likelihood-free" approaches in which there is no requirement for explicit likelihood evaluation [302], only the need to sample

from the likelihood. Nor is it the case that pre-computed posterior distributions are required in the training procedure.

Recently, a type of neural network known as CVAE was shown to perform exceptionally well when applied towards computational imaging inference [300, 303], text to image inference [304], high-resolution synthetic image generation [305], end-to-end text-to-speech synthesis [306], and the fitting of incomplete heterogeneous data [307]. CVAEs, as part of the variational family of inference techniques are ideally suited to the problem of function approximation and have the potential to be significantly faster than existing approaches. It is therefore this type of ML network that we apply in the GW case to accurately approximate the Bayesian posterior $p(x|y)$, where x represents the physical parameters that govern the GW signal, and are the quantities we are interested in inferring. The data y represents the noisy measurement containing the GW signal and obtained from a network of GW detectors.

The construction of a CVAE begins with the definition of a quantity to be minimised (referred to as a cost, or loss function). In our case we take the expectation over the cross entropy

$$H(p,r) = -\left\langle \int dx\, p(x|y) \log r_\theta(x|y) \right\rangle \tag{5.2}$$

between the true posterior $p(x|y)$ and $r_\theta(x|y)$, the parametric distribution that we will use neural networks to model and which we aim to be equal to the true posterior. The expectation value is taken over different realisations of signal and noise, y. The parametric model is constructed from a combination of 2 (encoder and decoder) neural networks $r_{\theta_1}(z|y)$ and $r_{\theta_2}(x|y,z)$ where

$$r_\theta(x|y) = \int dz\, r_{\theta_1}(z|y) r_{\theta_2}(x|y,z). \tag{5.3}$$

In this case the θ subscripts represent sets of trainable neural network parameters and the variable z represents locations within a *latent space*. This latter object is typically a lower dimensional space within which an encoder can represent the input data, and via marginalisation over z allows the construction of a rich family of possible probability densities of x.

Starting from Eq. 5.2 it is possible to derive a computable bound for the cross-entropy that is reliant on the r_{θ_1} and r_{θ_2} networks and a third "recognition" encoder network $q_\phi(z|x,y)$ governed

by the trainable parameter-set ϕ. The details of the derivation are described in the cost function derivation section (Sec. 5.2) and in [300] but equate to an optimisation of the ELBO (Sec. 2.6.2). The final form of the cross-entropy cost function is given by the bound

$$H \lesssim \frac{1}{N}\sum_{n=1}^{N_b}\left[\overbrace{-\log r_{\theta_2}(x_n|z_n,y_n)}^{L} + \overbrace{\mathrm{KL}\left[q_\phi(z|x_n,y_n)||r_{\theta_1}(z|y_n)\right]}^{\mathrm{KL}}\right], \tag{5.4}$$

which is also represented graphically in Fig. 5.1. The cost function is composed of 2 terms, the "reconstruction" cost L which is a measure of how well the decoder network r_{θ_2} predicts the true signal parameters x, and the KL-divergence cost that measures the similarity between the latent space distributions modelled by the r_{θ_1} and q_ϕ encoder networks. In practice, for each iteration of the training procedure, the integrations over x, y and z are approximated by a sum over a batch of N_b draws from the user defined prior $p(x)$, the known likelihood $p(y|x)$, and the recognition function $q_\phi(z|,x,y)$. Details of the training procedure are given in Sec. 5.4.

The implementation of the CVAE that we employ in this chapter has a number of specific features that were included in order to tailor the analysis to GW signals. The details of these enhancements are described in the network design (Sec. 5.3), training procedure (Sec. 5.4), data augmentation (Sec. 6.4), and phase/polarisation angle reparameterisation (Sec. 6.5) sections but in summary, the primary modifications are as follows, 1) Physically appropriate output decoder distributions are used for each output parameter: von Mises-Fisher distribution on the sky location parameters, von Mises distributions on all parameters with cyclic prior bounds, and truncated Gaussians for parameters with defined prior bounds. 2) Each of the functions $r_{\theta_1}, r_{\theta_2}$, and q_ϕ are modelled using deep convolutional neural networks with multi-detector timeseries represented as independent input channels. 3) The r_{θ_1} encoder models an $M = 32$ component Gaussian mixture model within the $n_z = 15$ dimensional latent space in order to capture the corresponding typical multi-modal nature of GW posterior distributions. 4.) All cyclic parameters are represented as points in an abstract 2D plane. In the next section, we will now derive the cost function used to train the entire CVAE outlined above.

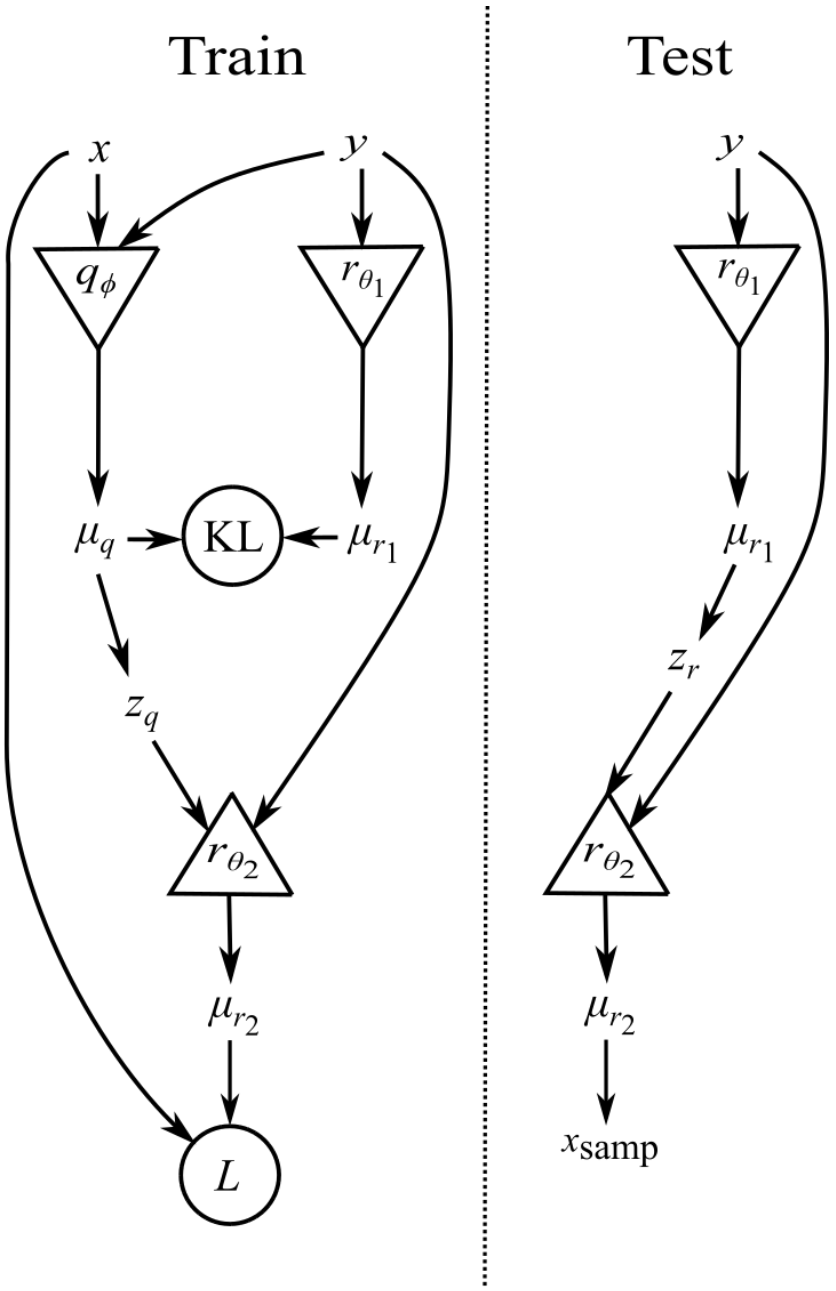


Figure 5.1: The configuration of the CVAE neural network. During training (left-hand side), a training set of noisy GW signals (y) and their corresponding true parameters (x) are given as input to encoder network q_ϕ, while only y is given to encoder network r_{θ_1}. The KL-divergence (Eq. 5.9) is computed between the encoder output latent space representations (μ_q and μ_r) forming one component of the total cost function. Samples (z_q) from the q_ϕ latent space representation are generated and passed to the decoder network r_{θ_2} together with the original input data y. The output of the decoder (μ_x) describes a distribution in the physical parameter space and the cost component L is computed by evaluating that distribution at the location of the original input x. When performed in batches this scheme allows the computation of the total cost function Eq. 5.4. After having trained the network and therefore having minimised the cross-entropy H, the testing stage (right-hand side) is performed using only the r_{θ_1} encoder and the r_{θ_2} decoder to produce samples (x_{samp}). These samples are drawn from the distribution $r_\theta(x|y)$ (Eq. 5.3) and accurately model the true posterior $p(x|y)$.

5.2 VItamin Cost Function Derivation

A CVAE is a form of variational autoencoder that is conditioned on an observation, where in our case the observation is a one-dimensional GW timeseries signal y, over a multi-detector network. The autoencoders from which variational autoencoders are derived are typically used for problems involving image reconstruction and/or dimensionality reduction. They perform a regression task whereby the autoencoder attempts to predict its own given input (model the identity function) through a "bottleneck layer" — a limited and therefore distilled representation of the input parameter space. An autoencoder is composed of two neural networks, an encoder and a decoder [308]. The encoder network takes as input a vector, where the number of dimensions is a fixed number predefined by the user. The encoder converts the input vector into a (typically) lower dimensional space, referred to as the *latent space*. A representation of the data in the latent space is passed to the decoder network which generates a reconstruction of the original input data to the encoder network. Through training, the two sub-networks learn how to efficiently represent a dataset within a lower dimensional latent space which will take on the most important properties of the input training data. In this way, the data can be compressed with little loss of fidelity. Additionally, the decoder simultaneously learns to decode the latent space representation and reconstruct that data back to its original form (the input data).

The primary difference between a variational autoencoder [164] and an autoencoder concerns the method by which locations within the latent space are produced. In our variant of the variational autoencoder, the output of the encoder is interpreted as a set of parameters governing statistical distributions in the latent space. In proceeding to the decoder network, samples from the latent space (z) are randomly drawn from these distributions and fed into the decoder, therefore adding an element of variation into the process. A particular input can then have a range of possible outputs. Any trainable network architectures can be used in both the decoder and the encoder networks and within `VItamin` we use deep convolutional neural networks in all cases.

We will now derive the cost function and the corresponding network structure and we begin with the statement defining the aim of the analysis. We wish to obtain a function that reproduces the posterior distribution (the probability of our physical parameters x given some measured

data y). The cross-entropy between 2 distributions is defined in Eq. 5.2 where we have made the distributions explicitly conditional on y (our measurement). In this case $p(x|y)$ is the target distribution (the true posterior) and $r_\theta(x|y)$ is the parametric distribution that we will use neural networks to construct. The variable θ represents the trainable neural network parameters.

The cross-entropy is minimised when $p(x|y) = r_\theta(x|y)$ and so by minimising

$$H = -\mathrm{E}_{p(y)}\left[\int dx\, p(x|y) \log r_\theta(x|y)\right], \tag{5.5}$$

where $\mathrm{E}_{p(y)}[\cdot]$ indicates the expectation value over the distribution of measurements y, we therefore make the parametric distribution as similar as possible to the target for all possible measurements y.

Converting the expectation value into an integral over y weighted by $p(y)$ we get

$$H = -\int dy\, p(y) \int dx\, p(x|y) \log r_\theta(x|y). \tag{5.6}$$

We then apply Bayes' theorem to obtain

$$H = -\int dy\, p(y) \int dx \frac{p(y|x)p(x)}{p(y)} \log r_\theta(x|y) \tag{5.7}$$

where $p(x)$ is the prior distribution on the physical parameters x, and $p(y|x)$ is the likelihood of x (the probability of measuring the data y given the parameters x). Cancelling out the $p(y)$ terms we arrive at

$$H = -\int dx\, p(x) \int dy\, p(y|x) \log r_\theta(x|y). \tag{5.8}$$

The CVAE network outlined in Fig. 5.1 makes use of a conditional latent variable model and our parametric model is constructed from the product of 2 separate distributions marginalised over the latent space as defined in Eq. 5.3. We have used θ_1 and θ_2 to indicate that the 2 separate networks modelling these distributions will be trained on these parameter sets respectively. The encoder $r_{\theta_1}(z|y)$ takes as input the data y and outputs parameters that describe a probability dis-

tribution within the latent space. The decoder $r_{\theta_2}(x|z,y)$ takes as input a single location z within the latent space together with the data y and outputs sets of parameters describing a probability distribution in the physical parameter space. The explicit mathematical form $r_{\theta_2}(x|z,y)$ takes is that of multiple multivariate Gaussian distributions whose moments, μ_{r_2}, are predicted by the parametric model given latent space samples z and input data y. $r_{\theta_1}(z|y)$ takes the form of a Gaussian mixture model with a whose moments and mixture component weights (collectively labeled as μ_{r_2}) are also inferred by the parametric model given only observed data y.

One could be forgiven for thinking that by setting up networks that simply aim to minimise H over the θ_1 and θ_2 would be enough to solve this problem. However, as shown in [303], this is an intractable problem and a network cannot be trained directly to do this. Instead, we have to also train an additional network to approximate the theoretical joint probability distribution $r_\theta(z|x,y)$, which is essentially already defined by the existing $r_\theta(x|y)$, $r_{\theta_1}(z|y)$ and $r_{\theta_2}(x|z,y)$ joint distributions. We call the neural network which approximates the theoretical distribution, $r_\theta(z|x,y)$, the recognition function , $q_\phi(z|x,y)$, which is governed by the trainable network parameters, ϕ, that will be used to derive an ELBO. Furthermore, $q_\phi(z|x,y)$ takes the form of multiple multivariate Gaussian distributions whose moments, μ_{q}, are predicted by the parametric model given parameters x and observed data y. It will become more clear in the derivation below that both defining and approximating this extra joint probability distribution is necessary because it allows us to define a computable form for $\log r_\theta(x|y)$.

We first define the KL-divergence between the recognition function and the distribution $r_\theta(z|x,y)$ as

$$\mathrm{KL}\left[q_\phi(z|x,y)||r_\theta(z|x,y)\right] = \int dz\, q_\phi(z|x,y) \log\left(\frac{q_\phi(z|x,y)}{r_\theta(z|x,y)}\right). \tag{5.9}$$

This is done because we want to minimise the difference between the theoretical joint distribution $r_\theta(z|x,y)$ and the approximate version, $q_\phi(z|x,y)$. Using Bayes theorem we can write $r_\theta(z|x,y)$ as

$$r_\theta(z|x,y) = \frac{r_{\theta_2}(x|z,y)\, r_{\theta_1}(z|y)}{r_\theta(x|y)}. \tag{5.10}$$

Plugging this into Eq. 5.9 we get

$$\mathrm{KL}\left[q_\phi(z|x,y)||r_\theta(z|x,y)\right] = \int dz\, q_\phi(z|x,y) \log\left(\frac{q_\phi(z|x,y) r_\theta(x|y)}{r_{\theta_2}(x|z,y) r_{\theta_1}(z|y)}\right). \tag{5.11}$$

Using the logarithm multiplication rule we arrive at

$$\begin{aligned}\mathrm{KL}\left[q_\phi(z|x,y)||r_\theta(z|x,y)\right] = &\int dz\, q_\phi(z|x,y) \log\left(\frac{q_\phi(z|x,y)}{r_{\theta_2}(x|z,y) r_{\theta_1}(z|y)}\right) + \\ &\int dz\, q_\phi(z|x,y) \log r_\theta(x|y).\end{aligned} \tag{5.12}$$

Realising then that the $\log r_\theta(x|y)$ may be taken out of the integral since it is not a function of z and that the integral of a probability distribution, $q_\phi(z|x,y)$ in this case, is simply equivalent to 1 we can write

$$\mathrm{KL}\left[q_\phi(z|x,y)||r_\theta(z|x,y)\right] = \int dz\, q_\phi(z|x,y) \log\left(\frac{q_\phi(z|x,y)}{r_{\theta_2}(x|z,y) r_{\theta_1}(z|y)}\right) + \log r_\theta(x|y). \tag{5.13}$$

Moving $\log r_\theta(x|y)$ to the left-hand side of the equation and moving the KL term to the right-hand side we get

$$\begin{aligned}\log r_\theta(x|y) = &\mathrm{KL}\left[q_\phi(z|x,y)||r_\theta(z|x,y)\right] + \\ &\int dz\, q_\phi(z|x,y) \log\left(\frac{q_\phi(z|x,y)}{r_{\theta_2}(x|z,y) r_{\theta_1}(z|y)}\right),\end{aligned} \tag{5.14}$$

where we realise that the right-hand integral term is simply a KL-divergence which we define as the ELBO given by

$$\mathrm{ELBO} = \int dz\, q_\phi(z|x,y) \log\left(\frac{r_{\theta_2}(x|y,z) r_{\theta_1}(z|y)}{q_\phi(z|x,y)}\right). \tag{5.15}$$

It is so-named since the KL-divergence has a minimum of zero and cannot be negative. Plugging Eq. 5.15 into Eq. 5.14 we arrive at

$$\log r_\theta(x|y) = \mathrm{ELBO} + \mathrm{KL}\left[q_\phi(z|x,y)||r_\theta(z|x,y)\right]. \tag{5.16}$$

where we now have $\log r_\theta(x|y)$ which we need for Eq. 5.8. If we were to find a $q_\phi(z|x,y)$ function (optimised on ϕ) that minimised the KL-divergence defined in Eq. 5.9 then we can state that

$$\log r_\theta(x|y) \geq \text{ELBO}. \tag{5.17}$$

Substituting Eq. 5.15 into Eq. 5.17 we get

$$\log r_\theta(x|y) \geq \int dz\, q_\phi(z|x,y) \log\left(\frac{r_{\theta_2}(x|y,z) r_{\theta_1}(z|y)}{q_\phi(z|x,y)}\right). \tag{5.18}$$

Using the logarithm division property we find

$$\log r_\theta(x|y) \geq \int dz\, q_\phi(z|x,y) \left[\log(r_{\theta_2}(x|y,z) r_{\theta_1}(z|y)) - \log q_\phi(z|x,y)\right]. \tag{5.19}$$

Distributing $q_\phi(z|x,y)$ to the log terms and using the logarithm multiplicative property it can be shown that

$$\begin{aligned}\log r_\theta(x|y) \geq &\int dz\, q_\phi(z|x,y) \log r_{\theta_2}(x|y,z) + \int dz\, q_\phi(z|x,y) \log r_{\theta_1}(z|y) \\ &- \int dz\, q_\phi(z|x,y) \log q_\phi(z|x,y).\end{aligned} \tag{5.20}$$

Making the realisation that $\int dz\, q_\phi(z|x,y) r_{\theta_2}(x|y,z)$ is simply an expectation value and pulling out $q_\phi(z|x,y)$ from the other two integrals we get

$$\log r_\theta(x|y) \geq \mathrm{E}_{q_\phi(z|x,y)}\left[\log r_{\theta_2}(x|z,y)\right] + \int dz q_\phi(z|x,y)(\log r_{\theta_1}(z|y) - \log q_\phi(z|x,y)).$$

and using the logarithm division property we get

$$\log r_\theta(x|y) \geq \mathrm{E}_{q_\phi(z|x,y)}\left[\log r_{\theta_2}(x|z,y)\right] + \int dz q_\phi(z|x,y) \frac{\log r_{\theta_1}(z|y)}{\log q_\phi(z|x,y)}$$

Finally, we make the realisation that the integral term is the negative KL-divergence of $q_\phi(z|x,y)$

and $r_{\theta_1}(z|y)$ and find that

$$\begin{aligned}\log r_\theta(x|y) \geq & \mathrm{E}_{q_\phi(z|x,y)}\left[\log r_{\theta_2}(x|z,y)\right] \\ & - \mathrm{KL}\left[q_\phi(z|x,y)||r_{\theta_1}(z|y)\right].\end{aligned} \tag{5.21}$$

We can now substitute this inequality into our cost function as defined by Eq. 5.8 to obtain

$$\begin{aligned}H \leq -\int dx\, p(x) \int dy\, p(y|x) \Big[& \mathrm{E}_{q_\phi(z|x,y)}\left[\log r_{\theta_2}(x|z,y)\right] \\ & - \mathrm{KL}\left[q_\phi(z|x,y)||r_{\theta_1}(z|y)\right]\Big],\end{aligned} \tag{5.22}$$

which can in practice be approximated as a stochastic integral over draws of x from the prior, y from the likelihood function $p(y|x)$, and from the recognition function, giving us Eq. 5.4, the actual function evaluated within the training procedure. In standard sampling algorithms it is required that the likelihood is calculated explicitly during the exploration of the parameter space and hence an analytic noise and signal model must be assumed. For a CVAE implementation we are required only to sample from the likelihood distribution, i.e., generate simulated noisy measurements given a set of signal parameters. This gives us the option of avoiding the assumption of detector noise Gaussianity in the future by training the CVAE using "real" non-Gaussian detector noise. We note that while the mathematics which describe our CVAE method outlined above are similar to those of the VAE and CVAE described in Ch. 2, Sec. 2.6, they are not the same (i.e. all methods are constructed to accomplish different objectives). In the next section, we will discuss in detail the CVAE network architecture, as well as specific design choices meant to tailor the model to our GW-specific problem domain.

5.3 VItamin Network Design

The CVAE network outlined in Fig. 5.1 is constructed from the 3 separate neural networks modelling the encoder and decoder distributions r_{θ_1} and r_{θ_2} as well as the recognition function q_ϕ. Each of these components is a deep convolutional network consisting of a series of one-

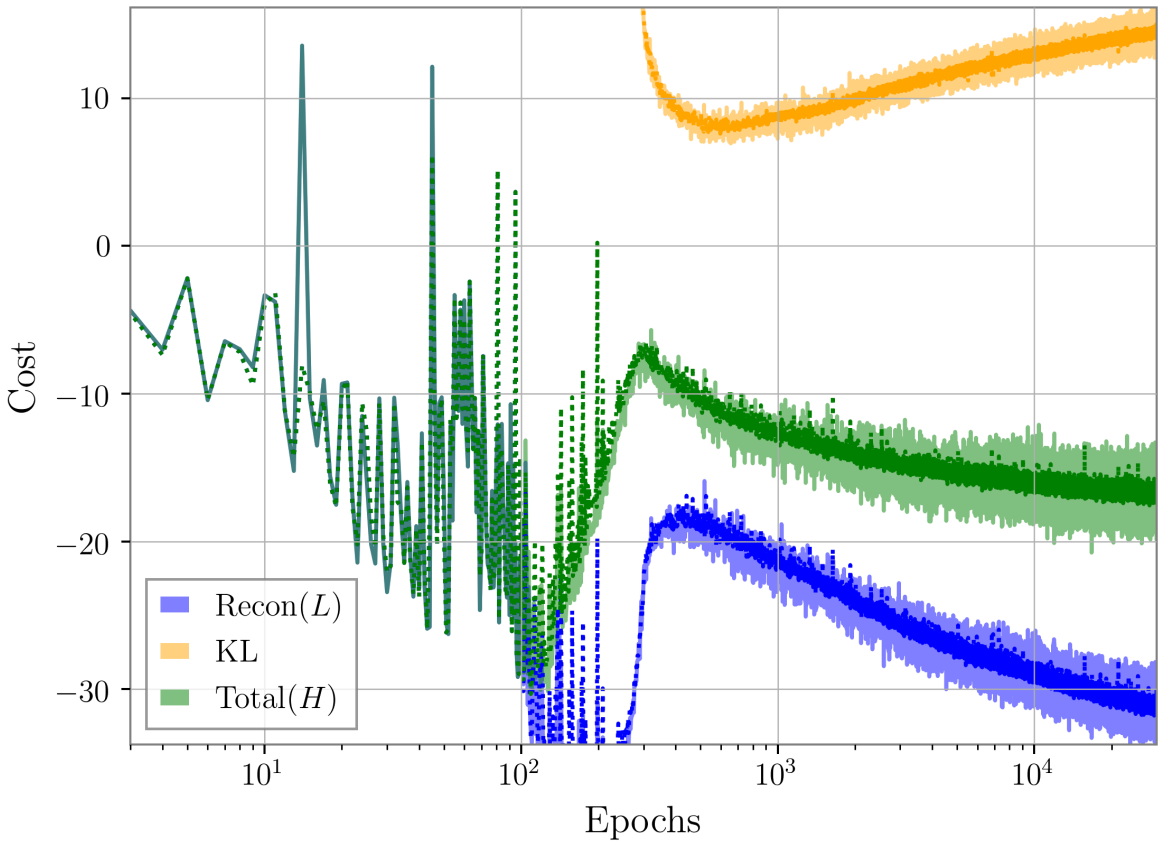


Figure 5.2: The cost as a function of training epoch. We show the total cost function (green) together with its component parts: the KL-divergence component (orange) and the reconstruction component (blue) which are simply summed to obtain the total. The dark curves correspond to the cost computed per epoch, defined as the network seeing 2×10^4 data samples, of training data and the lighter curves represent the cost when computed on independent validation data. The close agreement between training and validation cost values indicates that the network is not overfitting to the training data. The change in behavior of the cost between 10^2 and 3×10^2 epochs is a consequence of gradually introducing the KL cost term contribution via an annealing process, described in Sec. 5.4.

dimensional convolutional layers followed by a series of fully-connected layers. The details of each network structure are given in Table 5.3 where we indicate the activations used. We arrived at this network design through trial and error.

The r_{θ_1} network takes the input timeseries data y in the form of multi-channel 1-dimensional vectors where channels represent different GW detectors. After passing through a series of 1-dimensional convolutional and fully connected layers, the output then defines the parameters of a n_z-dimensional (diagonal covariance matrices) Gaussian mixture model in the latent space. We label these parameters as μ_{r_1} containing $n_z \times M$ means and log-covariances, where $M = 32$ mixture component weights and $n_z = 15$. The motivation for using this mixture model representation comes from the multi-modal nature of GW posterior distributions. The encoder network can use this flexibility to represent the y timeseries data as belonging to multiple possible latent space regions.

The recognition function network q_ϕ is very similar to the r_{θ_1} network with only 2 differences. The network takes as input the y timeseries and the true signal parameters x, however, only the y data is passed through the 1-dimensional convolutional layers. Only after the final convolutional layer where the output is flattened is the x data appended. It is then this compound timeseries data "feature-space" and true signal parameters that are processed using the remaining fully-connected layers. The second difference is that the output of the network defines a *single-modal* (diagonal) n_z-dimensional Gaussian. We label these parameters as μ_q containing $n_z = 15$ means and log-covariances. The rationale behind this choice is that since the q_ϕ distribution is conditional on the true signal parameters, there should be no ambiguity as to which mode in the latent space that a particular timeseries belongs to.

The decoder network r_{θ_2} is identical in structure to the q_ϕ network but with a difference in the form of their outputs and inputs. The r_{θ_2} network takes as input both latent space samples z and timeseries y. The r_{θ_2} output represents the parameters (μ_{r_2}) that govern an n_x-dimensional distribution in the physical parameter space where we have carefully chosen appropriate distributions for each of the physical parameters. For the luminosity distance, the binary inclination, the time of coalescence, and spin parameters $a_1, a_2, \Theta_1, \Theta_2$ we have adopted truncated Gaussian distributions where the truncation occurs at the predefined prior boundaries of the respective

Table 5.3: The `VItamin` network hyper-parameters. Dashed lines "—" indicate that convolutional layers are shared between all 3 networks.

Layer \ Network	$r_{\theta_1}(z\|y)$	$r_{\theta_2}(x\|y,z)$	$q_\phi(z\|x,y)$
Input y	[1024,3][a]	[1024,3]	[1024,3]
Layer 1	conv(64,3,96)[b] L2Reg(0.001)[c] act[d]=LeakyReLU	— — —	— — —
Layer 2	conv(32,96,96) stride(4)[e] L2Reg(0.001) act=LeakyReLU	— — — —	— — — —
Layer 3	conv(32,96,96) L2Reg(0.001) act=LeakyReLU	— — —	— — —
Layer 4	conv(16,96,96) stride(2) L2Reg(0.001) act=LeakyReLU	— — — —	— — — —
Layer 5	conv(16,96,96) L2Reg(0.001) act=LeakyReLU	— — —	— — —
Layer 6	conv(16,96,96) stride(2) L2Reg(0.001) act=LeakyReLU	— — — —	— — — —
Input z,x	flatten[f]→[6144]	flatten→[6144] append[g](z)→[6159]	flatten→[6144] append(x)→[6159]
Layer 7	FC(6159,4096)[h] act=LeakyReLU	FC(6159,4096) act=LeakyReLU	FC(6159,4096) act=LeakyReLU
Layer 8	FC(4096,2048) act=LeakyReLU	FC(4096,2048) act=LeakyReLU	FC(4096,2048) act=LeakyReLU
Layer 9	FC(2048,1024) act=LeakyReLU	FC(2048,1024) act=LeakyReLU	FC(2048,1024) act=LeakyReLU
Layer 10	FC(1024,960) act=None output=μ_{r_1} →[15,32,2][i]	FC(1024,30) act=(Sigmoid,-ReLU)[j] output=μ_{r_2} →[19,2][k]	FC(1024,30) act=None output=μ_q →[15,2][l]

[a] The shape of the data [one-dimensional dataset length, No. channels].
[b] one-dimensional convolutional filter with arguments (filter size, No. channels, No. filters).
[c] L2 regularization funciton applied to the kernel weights matrix.
[d] The activation function used.
[e] Striding layer with arguments (stride length).
[f] Take the multi-channel output of the previous layer and reshape it into a one-dimensional vector.
[g] Append the argument to the current dataset.
[h] Fully connected layer with arguments (input size, output size).
[i] The r_{θ_1} output has size [latent space dimension, No. modes, No. parameters defining each component per dimension].
[j] Different activations are used for different parameters. For the scaled parameter means we use sigmoids and for log-variances we use negative ReLU functions.
[k] The r_{θ_2} output has size [physical space dimension+additional cyclic dimensions, No. parameters defining the distribution per dimension]. The additional cyclic dimensions account for the 2 parameters each cyclic parameter is represented by in the abstract 2D plane.
[l] The q_ϕ output has size [latent space dimension, No. parameters defining the distribution per dimension].

parameter space dimensions. For the component masses we had initially adopted conditional truncated Gaussians where the conditional aspect was to ensure that $m_1 \geq m_2$[2], but found that training duration increased significantly because of this choice. We also found that the network typically learned this conditional boundary based on the training data alone anyways, so have now instead opted for using truncated Gaussian distributions alone for m_1 and m_2. Independent von Mises distributions are applied to phase, polarisation angle and spin parameters ϕ_{12}, ϕ_{jl} in order to capture the periodic nature of these parameters. We model all cyclic parameters as locations in an abstract 2D plane (Sec. 6.4) and apply an additional reparameterisation on ϕ_0 and ψ (Sec. 6.5). Finally, we use the von Mises-Fisher distribution to model the right ascension and declination (sky) parameters.

5.4 Training Procedure

Our cost function is composed of 3 probability distributions modelled by neural networks with well defined inputs and outputs where the mapping of those inputs to outputs is governed by the parameter sets θ_1, θ_2 and ϕ. These parameters are the weights and biases of 3 neural networks acting as (variational) encoder, decoder, and encoder respectively, as well as the trainable parameters of the optimiser. To train such a network one must connect the inputs and outputs appropriately to compute the cost function H (Eq. 5.4) and back-propagate cost function derivatives to update the network parameters.

Training is performed via a series of steps illustrated schematically in Fig. 5.1. A batch of data composed of pairs of timeseries y and their corresponding true GW signal parameters x are passed as input and the following steps are applied to each element of the batch.

1. The encoder q_ϕ takes both the timeseries y and the true parameters x defining the GW signal. It then encodes these instances into parameters μ_q defining an uncorrelated (diagonal covariance matrix) n_z-dimensional Gaussian distribution in the latent space.

2. The encoder r_{θ_1} is given only the timeseries data y and encodes it into a set of variables

[2]We note that this additional complication of requiring conditional decoder output distributions could have been avoided if a different mass parameterisation were chosen, e.g., total mass and symmetric mass ratio.

μ_{r_1} defining a multi-component multivariate Gaussian mixture distribution in the latent space.

3. We then draw a sample from the distribution described by μ_q giving us a location z_q within the latent space.

4. This sample, along with its corresponding y data, are then passed as input to the decoder r_{θ_2}. This decoder outputs μ_{θ_2} comprising a set of parameters that define a distribution in the physical x space.

5. The first term of the loss function, the reconstruction loss (defined as L in Eq. 5.4), is then computed by evaluating the probability density defined by μ_{θ_2} at the true x training value (the average is then taken over the batch of input data).

6. The second loss component, the KL-divergence between the distributions $q_\phi(z|x,y)$ and $r_{\theta_1}(z|y)$ (described by the parameter sets μ_q and μ_{r_1}), is given as

$$\mathrm{KL}\left[q_\phi(z|x_n,y_n)||r_{\theta_1}(z|y_n)\right] = q_\phi(z|x_n,y_n)\log\left(\frac{q_\phi(z|x_n,y_n)}{r_{\theta_1}(z|y_n)}\right)$$

where z_q is the sample drawn from $q_\phi(z|x_n,y_n)$ in the first training stage. We use this single-sample Monte-Carlo integration approximation since the KL-divergence between a single-component and a multi-component multivariate Gaussian distribution has no analytic solution (the average is then taken over the batch of input data, hence the index n).

7. The 2 loss components are then summed according to Eq. 5.4 and all trainable network parameters (defined by θ_1, θ_2, ϕ) are updated based on the derivative of the cost function with respect to these parameters.

A problematic aspect of training relates to the behaviour of the network during the initial stages of training. The network has a strong tendency to become trapped in local minima resulting in a decreasing cost component L (the reconstruction cost) but a non-evolving KL-divergence term that remains close to zero. To avoid this state we apply an annealing process in which the

KL-divergence term is initially ignored but its contribution is then increased logarithmically from 0 to 1 between the epoch indices 10^2–3×10^2. This allows the q_ϕ encoder to learn the latent space representation of the data via the reconstruction cost before being required to simultaneously try to best match its distribution to that modelled by the r_{θ_1} encoder. In parallel with the gradual introduction of the KL cost term, we also find that the stability of training is negatively affected by the complexity of our tailored output decoder likelihood functions. To resolve this we apply the same annealing procedure over the same epoch range in transitioning between unbound Gaussian likelihoods on all physical parameters to the tailored likelihoods, where the boundaries of the Gaussian likelihoods are brought in from -10 to 0 on the lower bound and 11 to 1 on the upper bound.

As is standard practice in ML applications, the cost is computed over a batch of training samples and repeated for a pre-defined number of epochs. An epoch traditionally is defined as the point at which the network has been trained on a number of samples equivalent to the size of the entire training set. However, in this study we define an epoch as the network having been trained on a number of samples equivalent to 2×10^4. For our purposes, we found that $\sim 3 \times 10^4$ training epochs, a batch size of 1500 training samples and a learning rate of 10^{-4} was sufficient. We used a total of 10^7 training samples in order to adequately cover the BBH parameter space. We additionally ensure that an (effectively) infinite number of noise realizations are employed by making sure that every time a training sample is used it is given a unique noise realisation despite only having a finite number of waveforms. Every epoch we also randomly shuffle the phase, time of coalescence and distance parameters for all training samples loaded from disk. See Sec. 6.4 for further details on data augmentation techniques used in this chapter.

Completion of training is determined by comparing output posteriors on test samples with those of `Bilby` iteratively during training. This comparison is done using standard figures of merit such as the p-p-plot and JS-divergence (see Figs. 5.5 and 5.10). We also assess training completion based on whether the evolution of the cost function and its component parts (Fig. 5.2) have converged to a reasonable degree. We use a single Nvidia Tesla V100 GPUs with 16/32 Gb of RAM although consumer grade “gaming" GPU cards are equally fast for this application. We also state that the onboard RAM memory of the machine/cluster itself has implications for

the batch size and consequently the speed and optimal learning rate to use.

5.5 Testing Procedure

After training has completed and we wish to use the network for inference we follow the procedure described in the right hand panel of Fig. 5.1. Given a new y data sample (not taken from the training set) we simply input this into the trained encoder r_{θ_1} from which we obtain a single value of μ_{r_1} describing a distribution (conditional on the data y) in the latent space. We then repeat the following steps:

1. We randomly draw a latent space sample z_{r_1} from the latent space distribution defined by μ_{r_1}.

2. The z_{r_1} sample and the corresponding original y data are fed as input to our pre-trained decoder network r_{θ_2}. The decoder network returns a set of parameters μ_{r_2} which describe a multivariate distribution in the physical parameter space.

3. We then draw a random x realisation from that distribution.

A comprehensive representation in the form of samples drawn from the entire joint posterior distribution can then be obtained by simply repeating this procedure and hence sampling from our latent model $r_\theta(x|y)$ (see Eq. 5.3). We also note that some physical parameters are in-fact reparameterised in the neural network model (i.e. all cyclic parameters, ϕ_0, ψ, and α) and must then be converted back to their original parameterisation immediately following step 3 above. See Sec. 6.4 for further details regarding how this is done.

5.6 Primary `VItamin` Results

We present results on 250 multi-detector GW test BBH waveforms in simulated advanced detector noise [309] from the LIGO Hanford, Livingston and Virgo detectors. We compare between variants of the existing Bayesian approaches and our CVAE implementation which we

call `VItamin`. Posteriors produced by the `Bilby` inference library [19] are used as a benchmark in order to assess the efficiency and quality of our machine learning approach with the existing methods for posterior sampling.

For the benchmark analysis we assume that 15 parameters are unknown: the component masses m_1, m_2, the luminosity distance $d_{\rm L}$, the sky position α, δ, the binary inclination Θ_{jn}, the GW polarisation angle ψ, the time of coalescence t_0, and the spin parameters $a_1, a_2, \Theta_1, \Theta_2, \phi_{12}, \phi_{jl}$. We do not include phase ϕ_0 in our results because we apply phase marginalisation to all Bayesian samplers since this improves overall stability and runtime [19]. For each parameter we use a uniform prior with the exception of the declination, inclination, and tilt angle parameters for which we use priors uniform in $\cos\delta$, $\sin\Theta_{jn}$, $\sin\Theta_1$, and $\sin\Theta_2$ respectively. We also use a conditional mass prior, such that m_2 is constrained to be $m_2 < m_1$. The corresponding prior ranges are defined in Table 5.5 and result in a training set SNR distribution that has a median value of SNR ≈ 9 and ranging between 0 and 85 (see Fig. 5.3). We use a sampling frequency of 1024 Hz, a timeseries duration of 1 s, and the waveform model used is `IMRPhenomPv2` [310] with a minimum cutoff frequency of 20Hz. For each input test waveform we run the benchmark analysis using multiple sampling algorithms (`ptemcee`,`Dynesty`,`emcee`, `CPnest`) available within `Bilby`. For each run and sampler we extract $\mathcal{O}(8000)$ samples from the posterior on the 14 physical parameters.

With regards to the parameters choices in Table 5.4, after having discussed with experts in the Bayesian sampler community, it is evident that Bayesian samplers are certainly not guaranteed to converge to the same results. Full convergence in many cases may require much fine tuning over many iterations for each individual run. Although we do not fine tune Bayesian benchmark samplers for each sampler and each individual test case, we do use settings which have been recommended to us by `bilby` developers and outside experts for each respective Bayesian sampler. Both the `Dynesty` and `CPNest` samplers have a tolerance threshold (change in the log evidence from one proposal to the next) which guarantees a certain level of convergence. We use the recommended tolerance level of 0.1 for both nested samplers. For the MCMC samplers, `emcee` performs poorly, but is known to have difficulties with convergence within the community. There are a handful of `ptemcee` test cases (5 of the 250) which show some

Table 5.4: Benchmark sampler configuration parameters. Values were chosen based on a combination of their recommended default parameters [19] and private communication with the `Bilby` development team.

sampler	parameters
`Dynesty` [160]	live-points = 1000, dlogz = 0.1, nact = 50, npool = 8, bound = None, sample = uniform
`ptemcee` [154]	walkers = 200, temperatures = 20, burn_in_nact = 50, thin_by_nact = 0.5, nsamples = 10000, threads = 10, autocorr_tol = 50, autocorr_csafety = 1, autocorr_tau = 1, gradient_tau = 0.1, gradient_mean_log_posterior = 0.1, Q_tol = 1.01, min_tau = 1, threads = 1,
`CPNest` [161]	live-points = 2048, maxmcmc = 1000, nthreads = 1, seed = 1994, dlogz = 0.1
`emcee` [153]	nwalkers = 250, nsteps = 14000, nburn = 4000, a = 1.4, burn_in_fraction = 0.25, burn_in_act_ = 3

minor indication of incomplete convergence, but after careful review we have determined that a lengthier burn-in period does not significantly improve the resulting posteriors.

The `VItamin` training process uses as input 10^7 whitened waveforms corresponding to parameters drawn from the same priors as assumed for the benchmark analysis. The waveforms are also of identical duration, sampling frequency, and use the same waveform model as in the benchmark analysis. The signals are whitened[3] using the same advanced detector PSDs [309] as assumed in the benchmark analysis. When each whitened waveform is placed within a training batch it is given a unique detector Gaussian noise realisation (after signal whitening this is simply zero mean, unit variance Gaussian noise). See Sec. 6.4 for further data augmentation details. The `VItamin` posterior results are produced by passing each of our 250 whitened noisy testing set of GW waveforms as input into the testing path of the pre-trained CVAE (Fig. 5.1). For each input waveform we sample until we have generated 8000 posterior samples on 15 physical parameters $x = (m_1, m_2, d_{\rm L}, t_0, \Theta_{jn}, a_1, a_2, \Theta_1, \Theta_2, \psi, \phi_0, \phi_{12}, \phi_{jl}, \alpha, \delta)$. We also note that any parameters (such as ϕ_0) can (if desired) be marginalised out within the CVAE procedure itself, rather than after training by choosing only to output a subset of the source parameter space in the final layer of the decoder network. When performing comparisons between the `VItamin` ap-

[3]The whitening is used primarily to scale the input to a magnitude range more suitable to neural networks. The *true* PSD does not have to be used for whitening, but training data and test data must be contain signals that have been whitened by the same PSD.

Table 5.5: The prior boundaries used on the BBH signal parameters for the benchmark and the CVAE analyses. We note that the polarisation angle and phase are represented in the 2D plane through a reparameterisation given in (Sec. 6.5).

Parameter name	symbol	min	max	units	prior function
mass 1	m_1	35	80	solar masses	Uniform
mass 2	m_2[a]	35	80	solar masses	Uniform
luminosity distance	d_{L}	1	3	Gpc	Uniform
time of coalescence	t_0	0.65	0.85	s	Uniform
phase at coalescence	ϕ_0	0	2π	radians	Uniform[b]
right ascension	α	0	2π	radians	Uniform[c]
declination	δ	$-\pi/2$	$\pi/2$	radians	Cosine
inclination	ι	0	π	radians	Sine
polarisation	ψ	0	π	radians	Uniform[d]
spin magnitude 1	a_1	0	0.8	-	Uniform
spin magnitude 2	a_2	0	0.8	-	Uniform
tilt angle 1	Θ_1	0	π	radians	Sine
tilt angle 2	Θ_2	0	π	radians	Sine
azimuthal angle	ϕ_{12}	0	2π	radians	Uniform[e]
azimuthal position	ϕ_{jl}	0	2π	radians	Uniform[f]
epoch	1126259642			GPS time	-
detector network	LIGO H1,L1, & Virgo V1			-	-

[a] Additionally m_2 is constrained such that $m_2 < m_1$.
[b] Phase has a periodic boundary condition.
[c] Right ascension has a periodic boundary condition.
[d] Polarisation has a periodic boundary condition.
[e] Azimuthal angle has a periodic boundary condition.
[f] Azimuthal position has a periodic boundary condition.

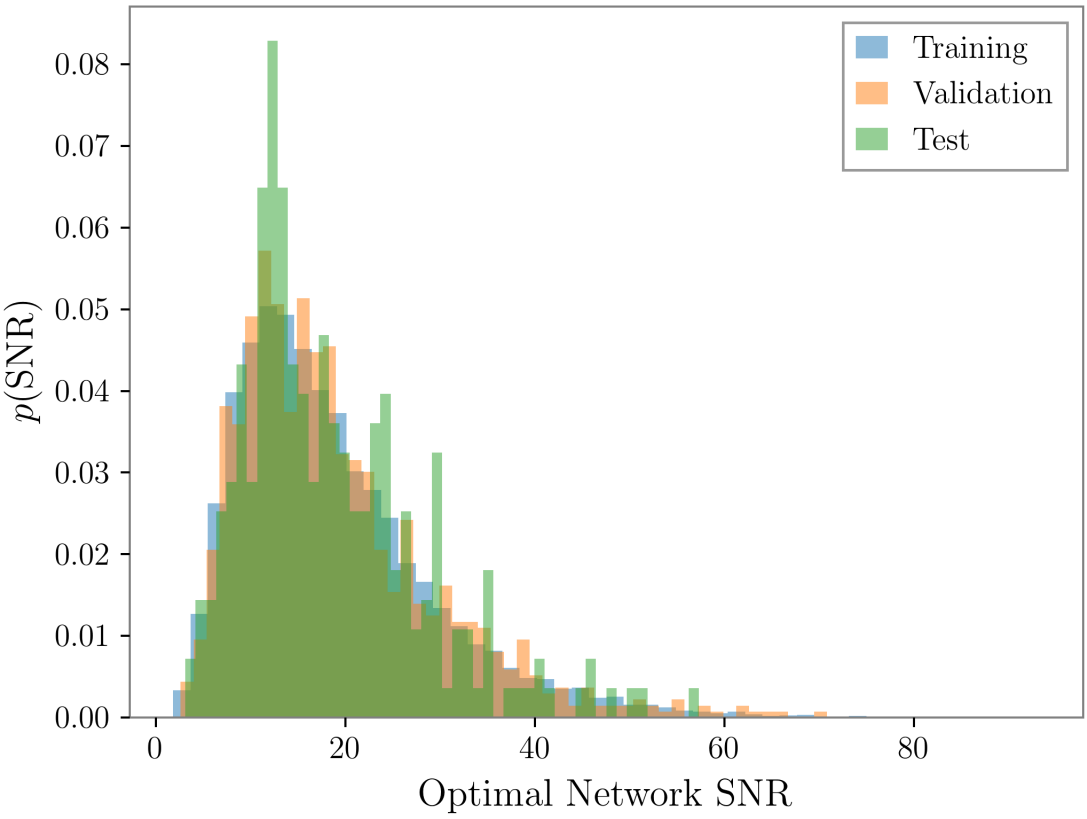


Figure 5.3: We show here a histogram of the optimal network SNR values of the `VItamin` training, validation and testing sets. The mode for all plotted distributions occurs at an SNR value of ~ 8 which drops off quickly to zero on the left-hand side. There is a tail on the right-hand side which drops off more gradually up to a maximum SNR value of ~ 85 for the training set, ~ 70 for the validation set and ~ 57 for the testing set. The peak location and general distribution of the SNR values is heavily dependent on both the chosen source parameter priors (Tab. 5.5) and the PSD.

proach and other bilby samplers, we only compare using 14 parameters (i.e. excluding ϕ_0) since we apply phase marginalisation to the bilby samplers when generating benchmark Bayesian test case posteriors.

We can immediately illustrate the accuracy of our machine learning predictions by directly plotting 2- and 1-dimensional marginalised posteriors generated using the output samples from our `VItamin` and `Bilby` approaches superimposed on each other. We show this for one example test dataset in Fig. 5.4 where strong agreement between the `Bilby` sampler `Dynesty` in blue, and the CVAE (red) is clear. It is also evident that whilst we refer to the `Bilby` sampler results as benchmark cases, different existing samplers do not perfectly agree with each other (i.e. `ptemcee` in green) despite using expert recommended sampler settings shown in Tab. 5.4. For each of our 250 test cases we see reasonable levels of agreement between pairs of benchmark samplers *and* between any benchmark sampler and our CVAE results.

Figures 5.5 and 5.10 show the results of multiple statistical tests (the p-p plot test and JS-divergence tests) performed on the entire test dataset and between all samplers (`Dynesty`, `ptemcee`, `CPNest`, `emcee`, and `VItamin`). In both tests the quality of the `VItamin` results are reasonably consistent with the benchmark samplers . A standard test used within the GW parameter estimation community is the production of so-called p-p plots which we show for our analysis and the benchmark comparisons in Fig. 5.5. The plot is constructed by computing a cumulative probability for each 1-dimensional marginalised test posterior evaluated at the true simulation parameter value (the fraction of posterior samples $\leq$ the simulation value). We then plot the cumulative distribution of these values [20]. Curves consistent with the black dashed diagonal line indicate that the 1-dimensional Bayesian probability distributions are consistent with the frequentist interpretation - that the truth will lie within an interval containing $X\%$ of the posterior probability with a frequency of $X\%$ of the time. It is clear to see that results obtained using `VItamin` show deviations from the diagonal that are entirely consistent with those observed in all benchmark samplers. The p-value has also been calculated for each sampler and each parameter under the null-hypothesis that they are consistent with the diagonal. These results show that for at least 1 parameter, emcee shows inconsistency with the modal at the 0.4% level. `Dynesty` has a worst case that is consistent only at the 0.7% level. All other

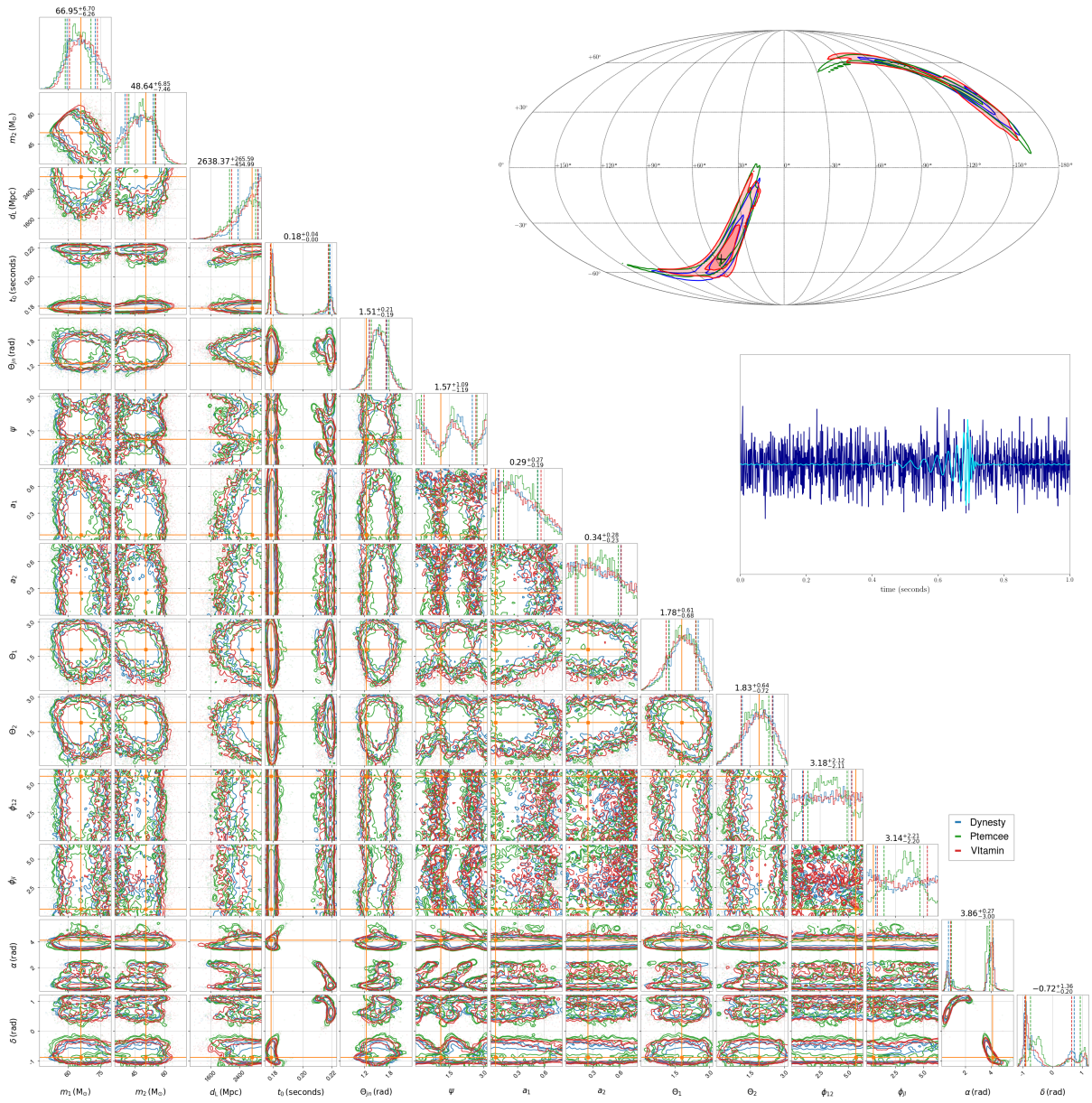


Figure 5.4: Corner plot showing 1 and 2-dimensional marginalised posterior distributions on the GW parameters for one example test dataset. Red contours represent the two-dimensional joint posteriors obtained from `VItamin` and blue and green contours are the corresponding posteriors output from our benchmark analyses (using the `Dynesty` and `ptemcee` samplers within `Bilby`). In each case, the contour boundaries enclose 68, 90 and 95% probability. 1 dimensional histograms of the posterior distribution for each parameter from both methods are plotted along the diagonal. Orange vertical and horizontal lines denote the true parameter values of the simulated signal. Vertical dashed lines in the 1 dimensional plots are representative of the 5% and 95% symmetric confidence bounds of the 3 sampler 1 dimensional posteriors. At the top right of the figure we include a Mollweide projection of the sky location posteriors from all three analyses. All results presented in this chapter correspond to a three-detector configuration but for clarity we only plot the H1 whitened noisy timeseries y and the noise-free whitened signal (in blue and cyan respectively) to the right of the figure. The test signal was simulated with an optimal multi-detector signal-to-noise ratio of 14.3.

samplers (including `VItamin`) show consistency at $> 0.4\%$ in the worst case. What these p-p plot results show is that the posteriors produced by `VItamin`, whilst perhaps not optimal, are still trustworthy and are unbiased in their estimation.

The JS-divergence is generally used as measure of the similarity between distributions defined as

$$\mathrm{JS}(\mathrm{P}||\mathrm{Q}) = \frac{1}{2}\mathrm{KL}[\mathrm{P}||\mathrm{Q}] + \frac{1}{2}\mathrm{KL}[\mathrm{Q}||\mathrm{P}] \tag{5.23}$$

where KL is the KL-divergence as defined in Eq. 5.9, and (P, Q) are two distributions we would like to measure the similarity between. In Fig. 5.6 we use this quantity to compare the output posterior estimates between samplers for the same input test data. To do this we run each independent sampler (including `VItamin`) on the same test data to produce samples from the corresponding posterior. We then compute the 1-dimensional JS-divergence between the output single parameter distributions from each sampler with every other sampler [311]. For distributions that are identical, the JS-divergence *should* equal zero but since we are representing our posterior distributions using finite numbers of samples, identical distributions result in finite JS-divergence values [312]. In Fig. 5.6, it can be seen that `Dynesty` vs. `VItamin` JS values closely match results from `Dynesty` vs. `Ptemcee` for nearly all parameters, with the exception of ϕ_{jl}, and ϕ_{12}. `VItamin` predictions have slightly higher JS values across all source parameters except for the spin parameters. `Dynesty` vs. CPNest seems to generally have similar JS values to `Dynesty` vs. `Ptemcee` with the exception of having broader credible intervals on t_0, θ_{jn}, ϕ_{jl}, α and δ. We also note in the 1-dimensional case that while JS-divergence values are reliable, they do not directly test the multi-dimensional correlations between source parameters.

We also provide additional JS-divergence figures of merit on 1-dimensional source parameter posteriors in Figs. 5.7, 5.8, and 5.9, where in each we highlight the comparison results of one Bayesian sampler (i.e. Fig. 5.7 highlights `CPNest`). In all 3 figures it can be seen that `VItamin` is generally consistent with the Bayesian sampler highlighted. In particular, we point out that `VItamin` appears to most closely agree with `CPNest`, as shown in the maroon colored bars of Fig. 5.7. Across all 3 figures there is generally more strong agreement between samplers on the spin parameters, and the least amount of agreement on the sky location pa-

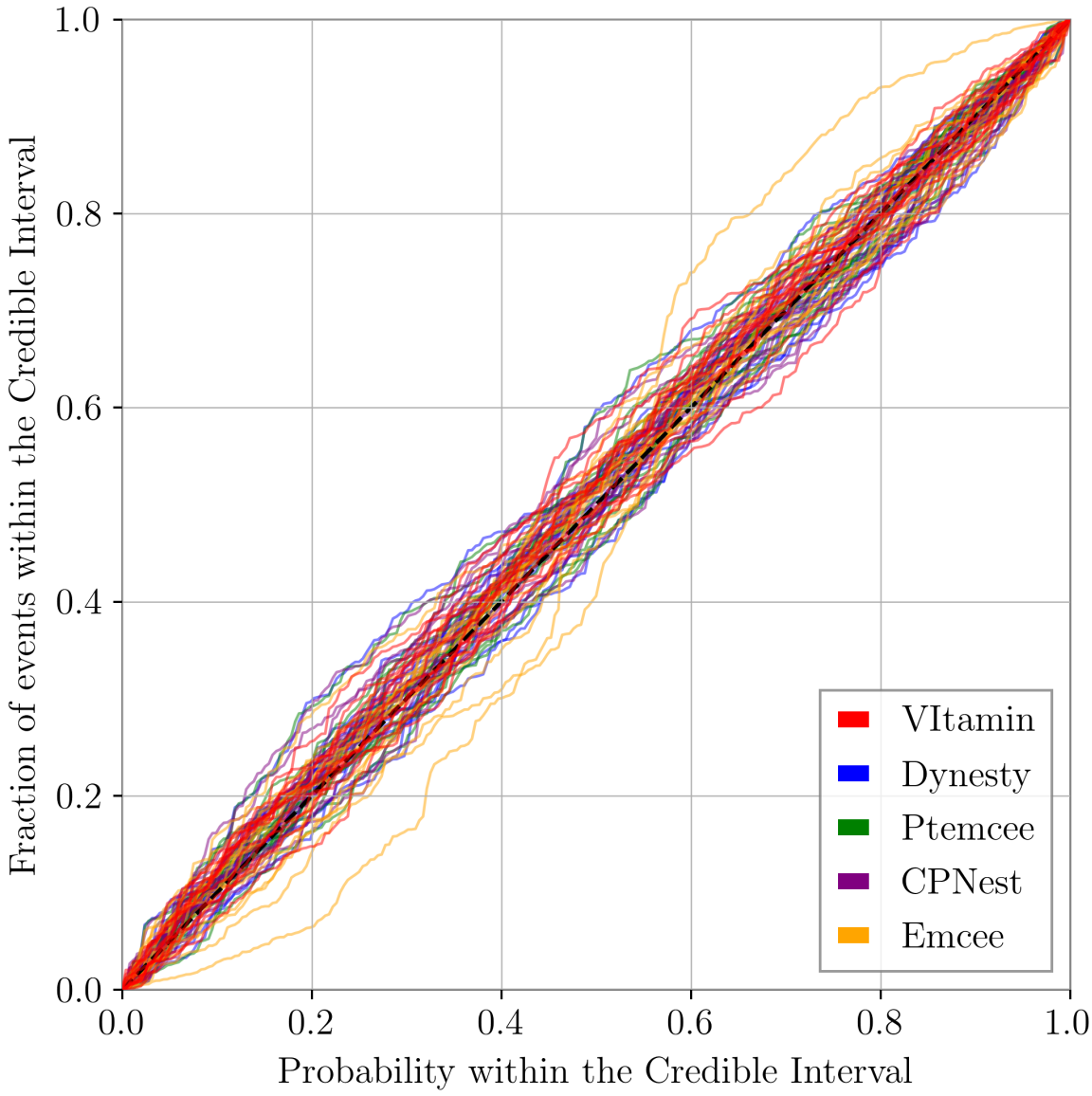


Figure 5.5: One-dimensional p-p plots for each parameter and for each benchmark sampler and `VItamin`. The curves were constructed using the 250 test datasets and the dashed black diagonal line indicates the ideal result. The best and worst-case p-values associated with each sampling method are (0.918, 0.047 `VItamin`), (0.912, 0.007 `Dynesty`), (0.931,0.007 `ptemcee`), (0.706,0.007 `CPNest`), (0.667,0.004 `emcee`).

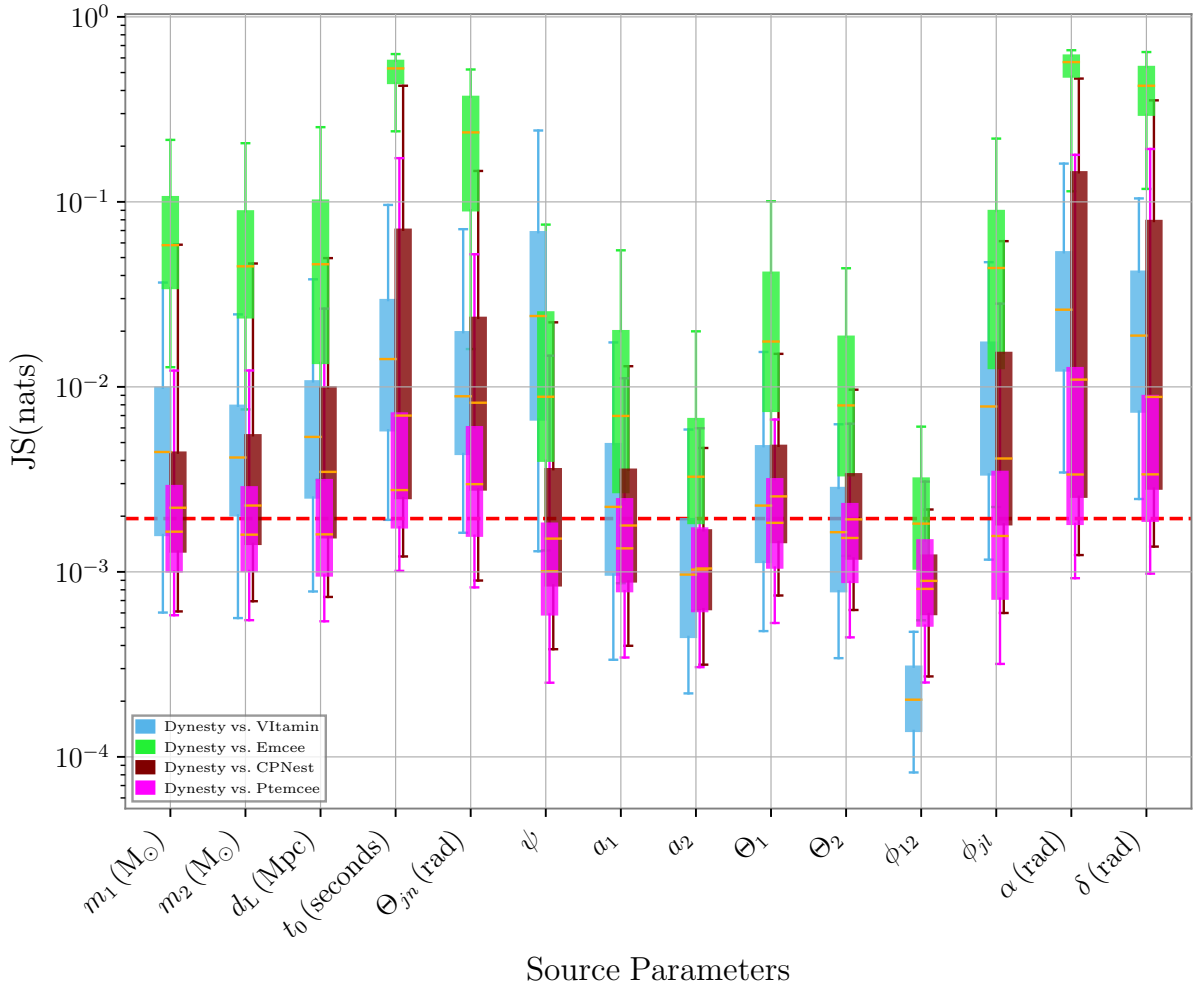


Figure 5.6: We show JS divergence values for all 250 test samples as a function of test sample source parameter for `Dynesty` against every other sampling approach. Each sampler method vs. another sampler method are denoted as different colors. The lower and upper end of boxes represent the 25th and 75th percentile credible regions respectively. The lower and upper end of the whiskers represent the 5th and 95th percentile credible regions. The orange lines are representative of the median JS values for each pair of compared samplers. The red horizontal line is representative of a JS baseline which is computed by taking the mean of all mean JS values computed over the 14 histograms in Fig. 6.9.

rameters. VItamin also consistently performs most poorly with respect to each sampler on the polarisation angle. What is also interesting to note is the level of disagreement of other Bayesian samplers with themselves across all 3 figures. This disagreement is especially prominent with regards to source parameters t_0, θ_{jn}, ϕ_{jl}, α, and δ. emcee vs. all other methods generally has higher JS-divergence values than all other comparison results. This is expected given the difficulty of obtaining emcee convergence. Finally, we note that although emcee results are fairly poor in comparison with other approaches, they are a useful benchmark for indicating underperformance.

In Fig. 5.10 we show the distributions of 14 dimensional JS-divergences for the 250 test GW samples. In each panel we plot the distribution of JS-divergences obtained when comparing one of the 4 benchmark samplers with all other benchmark samplers (excluding VItamin). We also plot the distribution of JS-divergences obtained when comparing the same sampler with VItamin alone. In all 4 cases the VItamin results show distributions completely consistent with the deviations observed between benchmark samplers. It is evident from the plot that Dynesty, CPNest, and ptemcee comparison results between themselves reach JS values far lower than any comparison result with VItamin. On the flip side, they also have tails at high JS values in their distributions which are consistent with those of VItamin comparison results. We see in all 4 subplots of Fig. 5.10 (and most clearly in the lower right subplot) that emcee continues to have the highest JS values of all comparison results. We also state here that the 14-dimensional JS-divergence distributions were estimated using an approximation technique[4] and a finite number of samples such that there was a fundamental noise of $\sim +/-0.15$ on the output values - hence even samples from 2 different sampler runs on the same test data would have JS-divergence scatter of this magnitude around 0. We have also carried out an additional study in Sec. 6.3 where we provide an approximated limit through two independent Dynesty runs over the entire test set.

The dominating computational cost of running VItamin lies in the training time, which can take $\mathcal{O}(7)$ days to complete. Completion is determined by comparing posteriors produced by the machine learning model and those of Bilby iteratively during training. We additionally assess

[4] (https://pypi.org/project/universal-divergence/)

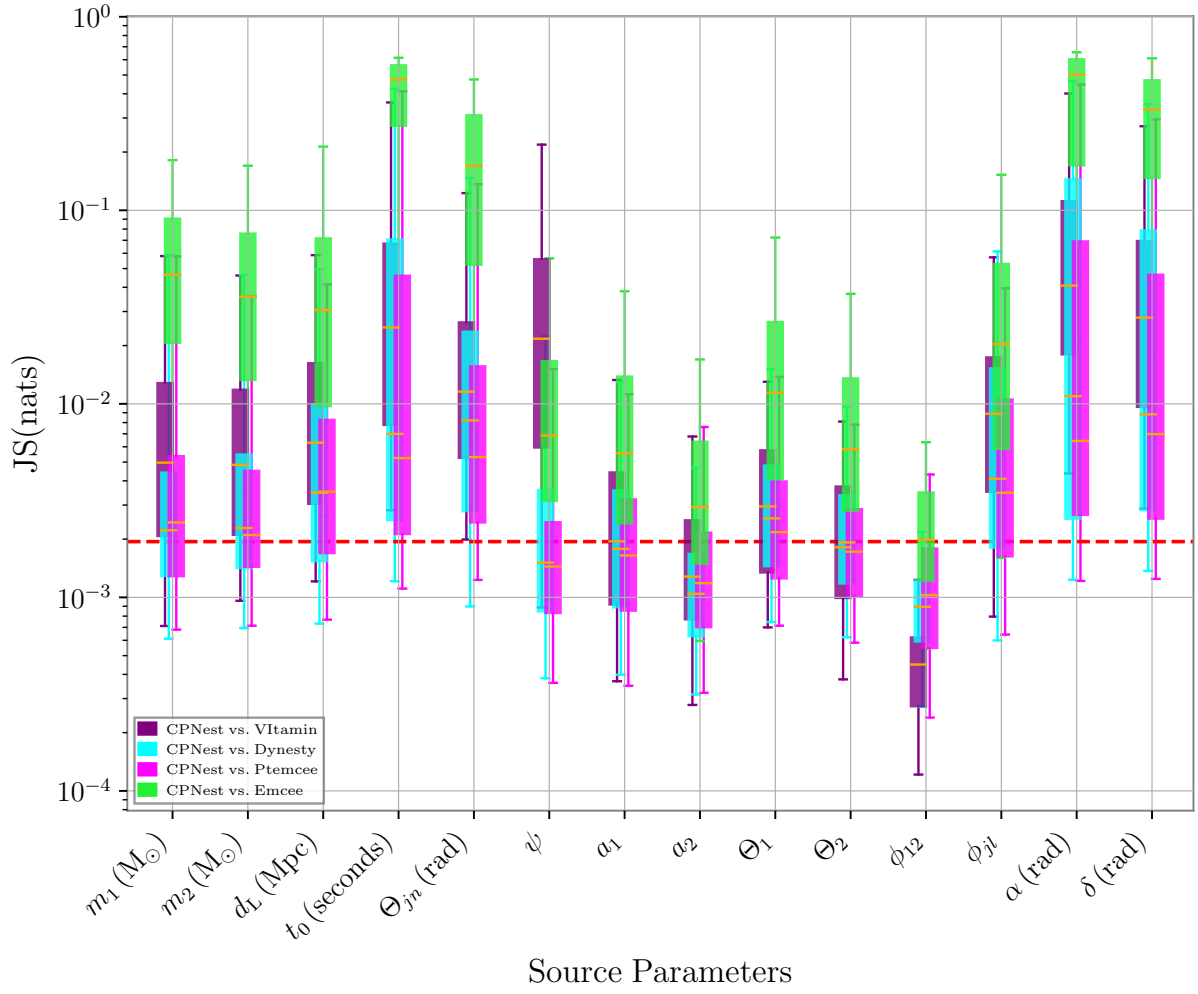


Figure 5.7: We show JS divergence values for all 250 test samples as a function of test sample source parameter for `CPnest` against every other sampling approach. Each sampler method vs. another sampler method are denoted as different colors. The lower and upper end of boxes represent the 25th and 75th percentile credible regions respectively. The lower and upper end of the whiskers represent the 5th and 95th percentile credible regions. The orange lines are representative of the median JS values for each pair of compared samplers. The red horizontal line is representative of a JS baseline which is computed by taking the mean of all mean JS values computed over the 14 histograms in Fig. 6.9.

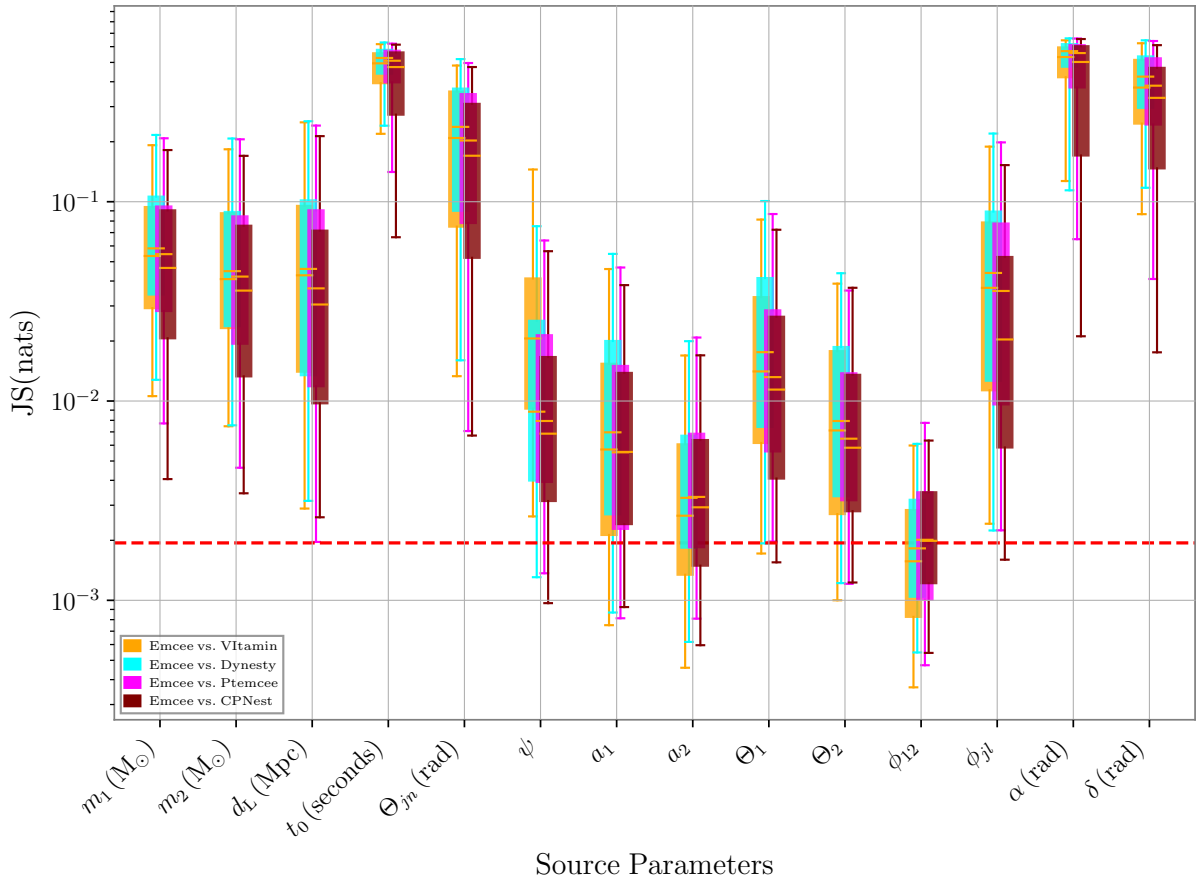


Figure 5.8: We show JS divergence values for all 250 test samples as a function of test sample source parameter for `emcee` against every other sampling approach. Each sampler method vs. another sampler method are denoted as different colors. The lower and upper end of boxes represent the 25th and 75th percentile credible regions respectively. The lower and upper end of the whiskers represent the 5th and 95th percentile credible regions. The orange lines are representative of the median JS values for each pair of compared samplers. We see here that `VItamin` performs to within a degree of accuracy which is consistent with other Bayesian samplers when looking at predictions on an individual source parameter basis. The red horizontal line is representative of a JS baseline which is computed by taking the mean of all mean JS values computed over the 14 histograms in Fig. 6.9.

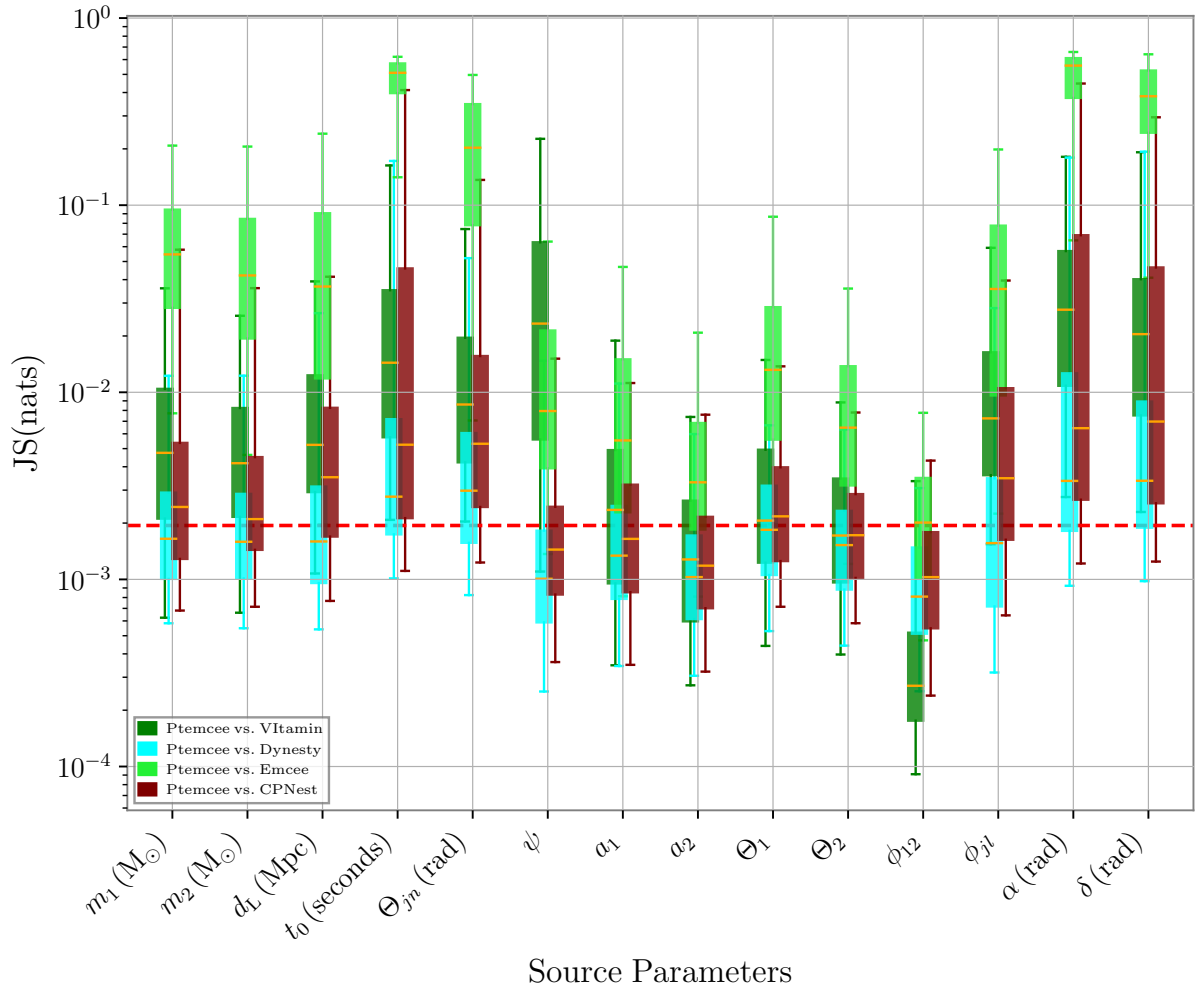


Figure 5.9: We show JS divergence values for all 250 test samples as a function of test sample source parameter for `Ptemcee` against every other sampling approach. Each sampler method vs. another sampler method are denoted as different colors. The lower and upper end of boxes represent the 25th and 75th percentile credible regions respectively. The lower and upper end of the whiskers represent the 5th and 95th percentile credible regions. The orange lines are representative of the median JS values for each pair of compared samplers. We see here that `VItamin` performs to within a degree of accuracy which is consistent with other Bayesian samplers when looking at predictions on an individual source parameter basis. The red horizontal line is representative of a JS baseline which is computed by taking the mean of all mean JS values computed over the 14 histograms in Fig. 6.9.

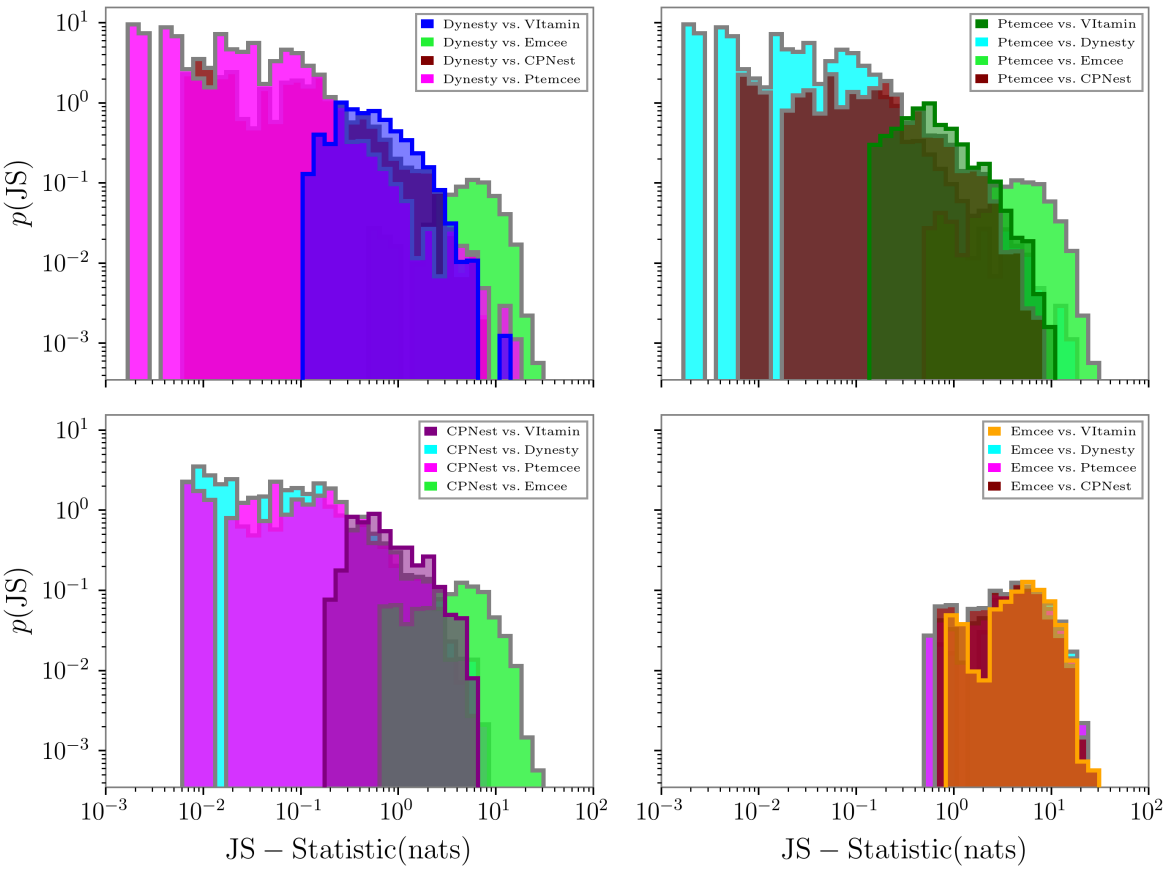


Figure 5.10: Distributions of JS-divergence values between posteriors produced by different samplers. In each panel we show the distribution of JS-divergences computed between a single benchmark sampler and every other benchmark sampler over all 250 GW test cases (grey histogram outlines). JS-divergence is in terms of nats where nat stands for "natural unit of measurement" and is a unit of information. Also plotted in each panel are the corresponding JS-divergence distributions between the single benchmark sampler and the `VItamin` outputs (colored histogram outlines).

whether the cost curves (Fig. 5.2) have converged, such that their slope is starting to approach zero. In Fig. 5.11 we plot the validation set moving average cost as a function of training epoch, where each color is a different run of `VItamin` using different hyperparameters/run settings carried out over the course of this thesis. We see that, generally speaking, the cost for all runs increases and then subsequently decreases over the course of training. The reason for the initial increase is because we slowly anneal in the KL component of the cost and thus it is high initially since it is not being used to update the parameters of the network. There are also other runs which are flat in the initial epochs and this is because those runs are in-fact runs which have been resumed from previously trained networks, so thus do not have KL annealing applied and are initially much lower in cost value. Overall, what we see for all `VItamin` runs near the end of training is that the cost values all start to approach a reasonable equilibrium. One could argue that if we were to allow all networks to continue training, that it would possible that the loss *may* decrease a significant amount. Whilst this could happen, it can take about a week to reach 5×10^4 epochs. One could then infer that given the general trends in Fig. 5.11 that we would receive diminishing returns in terms of improved cost values with further training time. We stress that once trained, there is no need to retrain the network unless the user wishes to use different priors $p(x)$ or assume different noise characteristics. The speed at which posterior samples are generated for all samplers used, including `VItamin`, is shown in Table 5.6. Run-time for the benchmark samplers is defined as the time to complete their analyses when configured using the parameter choices defined in Table 5.4. For `VItamin`, the run-time is defined as the total time to produce 8000 samples. To be clear, this does not include the $\mathcal{O}(7)$ days required to train the network. For our test case of BBH signals `VItamin` produces samples from the posterior at a rate which is ~ 6 orders of magnitude faster than our benchmark analysis using current inference techniques, representing a dramatic speed-up in performance.

5.7 Summary

In this chapter we have demonstrated that we are able to reproduce, to a high degree of accuracy, Bayesian posterior probability distributions generated through ML. This is accomplished using

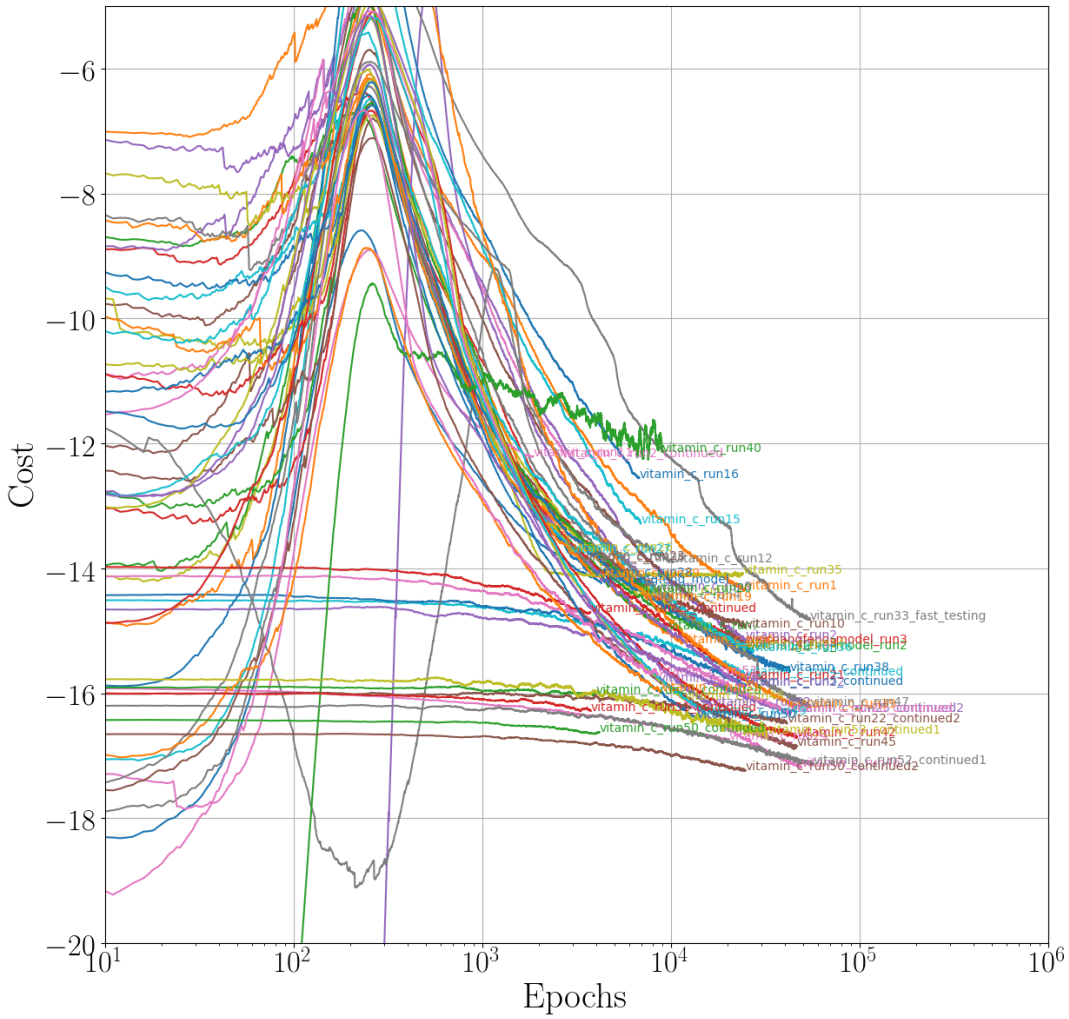


Figure 5.11: We plot multiple independent runs of `VItamin` validation set cost curves as a function of training epoch. All validation curve values are computed using a moving average. Epoch values are plotted in log-scale on the x-axis.

Table 5.6: Durations required to produce samples from each of the different posterior sampling approaches.

sampler	run time (s)			ratio $\frac{\tau_{\texttt{VItamin}}}{\tau_X}$
	min	max	median	
`Dynesty`[a] [160]	21564	261268	45607 [b]	2.2×10^{-6}
`emcee` [153]	16712	39930	19821	5.1×10^{-6}
`ptemcee` [154]	2392	501632	41151.0	2.4×10^{-6}
`CPNest` [161]	10309	437008	83807	1.2×10^{-6}
`VItamin`[c]		$\mathbf{1 \times 10^{-1}}$		1

[a] The benchmark samplers all produced $\mathcal{O}(8000)$ samples dependent on the default sampling parameters used.

[b] We note that there are a growing number of specialised techniques [313, 314, 315, 316] designed to speed up traditional sampling algorithms that could be used to reduce the runtimes quoted here by $\mathcal{O}(1-2)$ orders of magnitude.

[c] For the `VItamin` sampler 8000 samples are produced as representative of a typical posterior. The run time is independent of the signal content in the data and is therefore constant for all test cases.

a CVAE trained on simulated GW signals and does not require the input of pre-computed posterior estimates. We have demonstrated that our neural network model, which when trained, can produce complete and accurate posterior estimates in a fraction of a second, achieves the same quality of results as the trusted benchmark analyses used within the LVK.

The significance of our results is most evident in the orders of magnitude increase in speed over existing algorithms. We have demonstrated the approach using BBH signals but with additional work to increase sample rate and signal duration, the method can also be extended for application to signals from BNS mergers (e.g., GW170817 [9], and GW190425 [317]) and NSBH [15] systems where improved low-latency alerts will be especially pertinent. By using our approach, parameter estimation speed will no longer be limiting factor[5] in observing the prompt EM emission expected on shorter time scales than is achievable with existing LVK analysis tools such as Bayestar [22].

The predicted number of future detections of BNS mergers (~ 180 [18]) will severely strain the GW community's current computational resources using existing Bayesian methods (Tab. 5.1 and Tab. 5.2). We anticipate that future iterations of our approach will provide full-parameter estimation on all classes of CBC signals in $\mathcal{O}(1)$ s on single GPUs. Our trained network is

[5] A complete low-latency pipeline includes a number of steps. The process of GW data acquisition is followed by the transfer of data. There is then the corresponding candidate event identification, parameter estimation analysis, and the subsequent communication of results to the EM astronomy community after which there are physical aspects such as slewing observing instruments to the correct pointing.

also modular, and can be shared and used easily by any user to produce results. The specific analysis described in this chapter assumes a uniform prior on the signal parameters. However, this is a choice and the network can be trained with any prior the user demands, or users can cheaply resample accordingly from the output of the network trained on the uniform prior. We also note that our method will be invaluable for population studies since populations may now be generated and analysed in a fully-Bayesian manner on a vastly reduced time scale.

For BBH signals, GW data is usually sampled at 1—4 kHz dependent upon the mass of binary. We have chosen to use the noticeably low sampling rate of 1024Hz in order to decrease the computational time required to develop our approach and the computational burden of computing our 250 benchmark analyses for each of 4 benchmark samplers. We have found that increasing the sampling frequency of our input comes at the cost of a small increase in training time and a similar increase on the GPU memory requirement. We note that with the exception of requiring 1-dimensional convolutional layers and an increase in the amount of training data to efficiently deal with a multi-detector analysis, the network complexity has not increased with the dimensionality of the physical parameter space nor with the sampling rate of the input data. Given this, it is possible that extending the parameter space to lower masses may not be problematic.

In reality, GW detectors are affected by non-Gaussian noise artefacts and time-dependent variation in the detector noise PSD. Existing methods incorporate a parameterised PSD estimation into their inference [318]. To account for these and to exploit the "likelihood-free" nature of the CVAE approach, we could re-train our network at regular intervals using samples of real detector noise (preferably recent examples to best reflect the state of the detectors). In this case we could also apply transfer learning to speed up each training instance based on the previously trained network state. Alternatively, since the PSD is an estimated quantity, we could marginalise over its uncertainty by providing training data whitened by samples drawn from a distribution of possible PSDs. Furthermore, one could also provide the PSD estimates from a distribution of PSDs as an additional conditional input to the CVAE. Our work can naturally be extended to include the full range of CBC signal types but also to any and all other parameterised GW signals and to analyses of GW data beyond that of ground based experiments. Given the

abundant benefits of this method, we hope that a variant of this of approach will form the basis for future GW parameter estimation.

Chapter 6

Supplemental `VItamin` Results and Analysis

In the following sections we will discuss additional analysis which supplement the main results of Ch. 5 including the training set SNR distribution and it's relationship to the performance of `VItamin`, as well as the structure and behavior of the CVAE latent space and it's relationship with the predicted posterior distributions.

6.1 Jensen-Shannon Divergence as a Function of Signal-to-Noise Ratio

Over the course of the work carried out in this chapter it was thought that there was the possibility that high SNR signals could be a limiting factor with regards to the performance of the neural network model. It was originally hypothesised that due to the low number of high SNR signals in our training set (as seen in Fig. 5.3), that the network would perform worse on high SNR signals. This hypothesis is supported by the well known understanding in ML literature that less available data in specific regions of the parameter space can cause a neural network model to underfit to those regions [162]. In Fig. 6.1 we plot the 14-dimensional JS-divergence for all test sample cases of `VItamin` vs. `Dynesty` as a function of SNR. Each triangle sign is representative of individual test cases and different colors correspond to each interferometer.

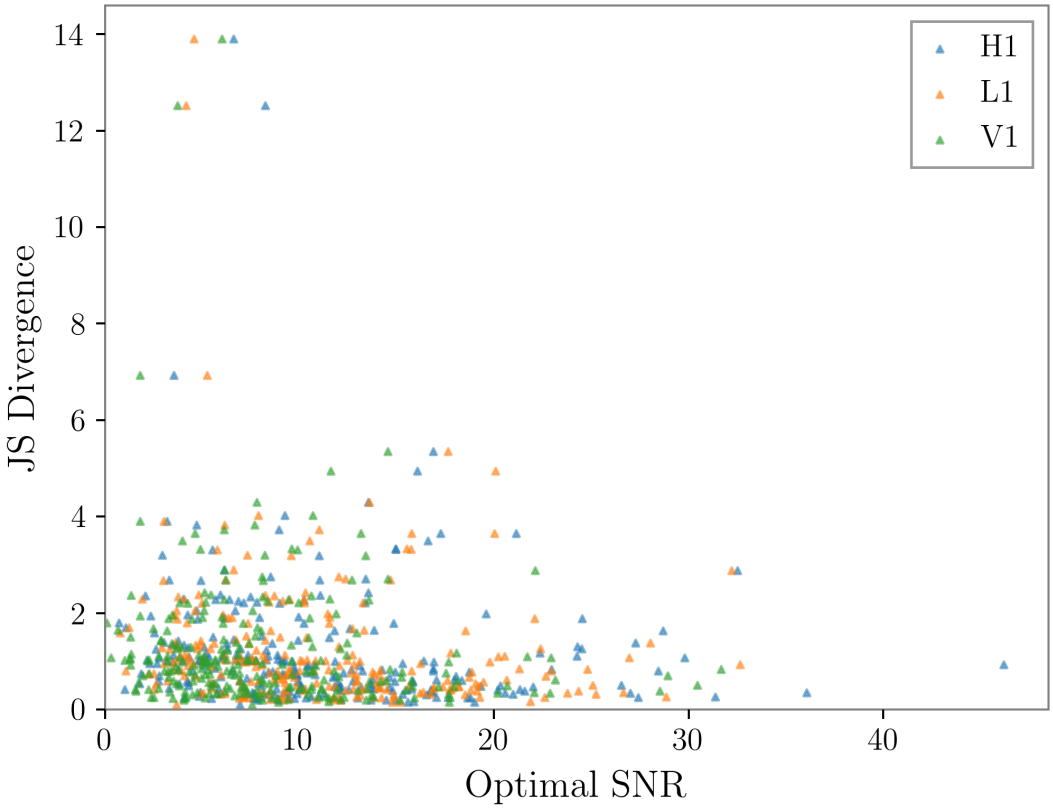


Figure 6.1: Illustrated in the plot are the 14-dimensional JS-divergence values of `Dynesty` vs. `VItamin` as a function of optimal SNR (Ch. 1, Sec. 1.6.2). Different colours are representative of each of the 3 detectors (H1, L1, V1) used in the analysis. Each triangle symbol represents a different test GW case. As can be seen, there does not appear to be a strong positive correlation with SNR.

We see in Fig. 6.1 that there is little to no positive correlation between SNR and the 14-dimensional JS- divergence; instead, it appears that there is a slight negative correlation. This plot shows that there would be marginal benefit gained from augmenting the training set to more strongly emphasise high SNR signals, and that the neural network may in fact struggle the most with low SNR events. Given that the highest JS-divergence values appears to be loosely correlated with low SNR values, one might assume that this could partially be solved by over-sampling our training set in the low SNR regime. This is not necessarily practical given that the prior would then need to be changed and thus the output posterior would then need to be resampled to compensate through techniques such as importance sampling [319] or rejection sampling [320].

One other interesting relationship to analyse is that between the 14-dimensional JS-divergence and the optimal SNR spread across all 3 detectors for a given GW test signal. The SNR spread is calculated by taking the difference between the maximum and minimum optimal SNR values from each detector for each test sample. We plot the JS-divergence as a function of SNR spread in Fig. 6.2 in order to gauge whether our neural network model performs worse when 1 or more detectors sees the test signal at a lower SNR than the other detectors. We see in Fig. 6.2 that there appears to be no positive correlation and possibly a very weak negative correlation between the two quantities. This could indicate that our model has some difficulties with signals which have an equal amount of optimal SNR across all detectors.

We also plot in Fig. 6.3 the JS-divergence as a function of the second highest SNR for each test signal. We plot as a function of the second highest SNR so as to highlight the cases where only one detector sees the GW signal clearly. This is of interest because cases such as these are more likely to be an issue for reliable parameter estimation. We see in Fig. 6.3 that there is little positive correlation and perhaps some negative correlation.

6.2 VItamin Latent Space Analysis

In this section we will introduce diagnostic plots we have used in order to gauge the behaviour of our neural network model with respect to the latent space. For context, we will first examine

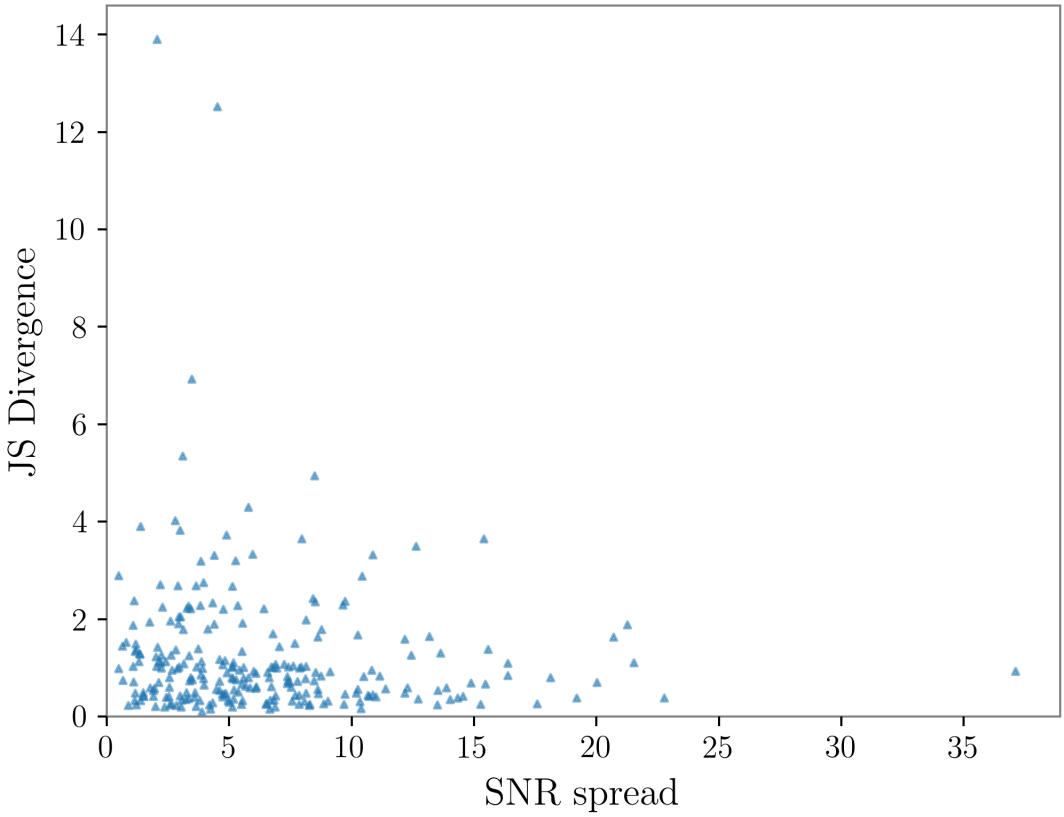


Figure 6.2: Illustrated in this plot are the 14-dimensional JS-divergence values of Dynesty vs. VItamin as a function of optimal SNR (Ch. 1, Sec. 1.6.2) spread between the 3 detectors for each test sample. Each triangle symbol represents a different test GW case.

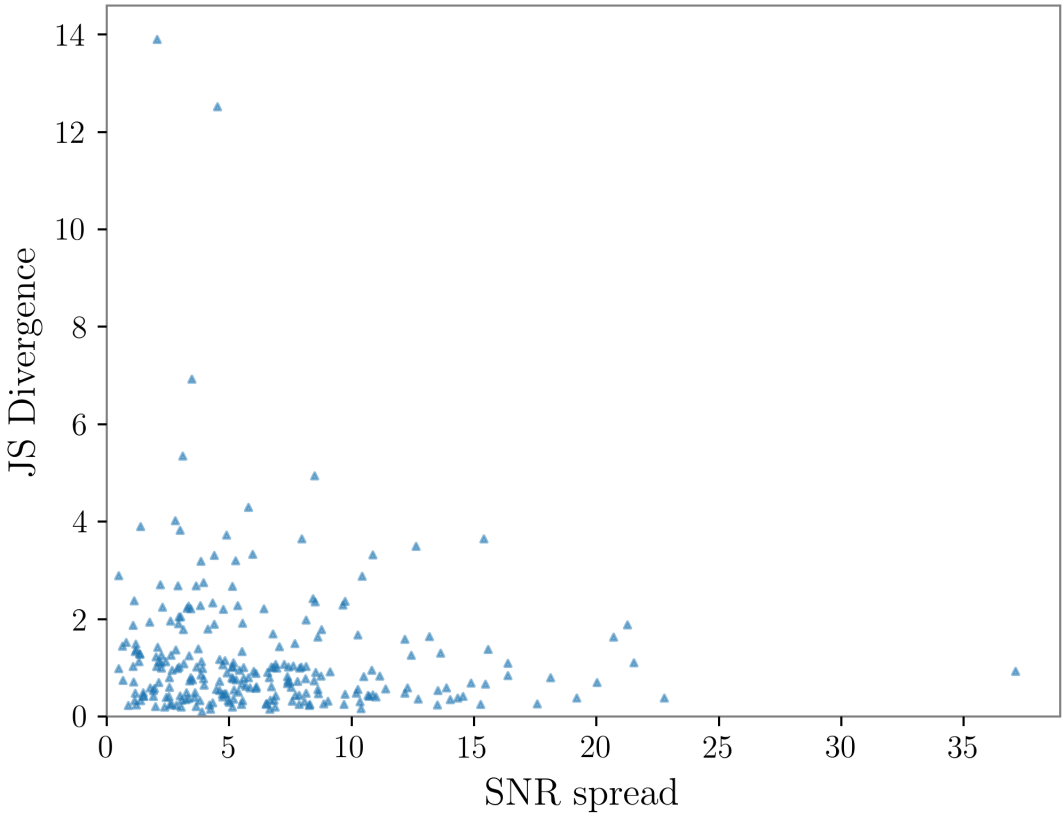


Figure 6.3: Illustrated in this plot are the 14-dimensional JS-divergence values of Dynesty vs. VItamin as a function of the second highest SNR between the 3 detectors for each test sample. Each triangle symbol represents a different test GW case.

a test case at an approximately median network SNR value of 10.89. As seen in Fig. 6.4, this median SNR event exhibits complex multi-modal behavior on multiple parameters including: the polarisation angle, time of coalescence, right ascension and declination. We will discuss in the following paragraphs the latent space structure associated with this test sample.

There is no hard and fast rule for determining the number of latent space dimensions to use when deciding on the architecture for a CVAE. That being said, our primary logical motivation for choosing a latent space size of 15 is directly related to the total number of source parameters we are trying to get the neural network model to learn. We want to ensure that the information in each source parameter dimension can be encoded in a latent space which is of sufficient size to do so. As can be seen in Fig. 6.5, there are some indications that this hypothesis may indeed be holding true.

In Fig. 6.5 we plot latent space samples drawn from latent space predictions made by both the q (blue) and r_1 (red) encoder networks after training. The probability distributions modeled by the encoder networks are multivariate Gaussians, whose means and standard deviations are inferred by the encoders. If a dimension of the latent space is informing posterior predictions, we would expect that the corresponding 1-dimensional marginalised posterior for that latent space dimension along the diagonal of the corner plot to show some level of disagreement between the predictions from both encoder networks. The reasoning behind this statement is that the q network is given a different level of information with respect to the r_1 network. Specifically, the q network is given not only the GW timeseries, but also the true values of the source parameters themselves, whereas the r_1 network is given the GW time series by itself. If a latent space dimension is not being used it means that there is no additional amount of information that can be gleaned from that dimension, so both encoder networks will return a simple Gaussian distributions, which contribute little to the loss function. If little is being contributed to the loss function, then little is being learned from that latent space dimension. We can say that there is a near null contribution to the loss function because the KL divergence between two Gaussian distributions is known to be close zero. Since the KL component of the loss is near zero for this dimension, weights will not be updated during the backpropagation process which encourage this dimension to be used.

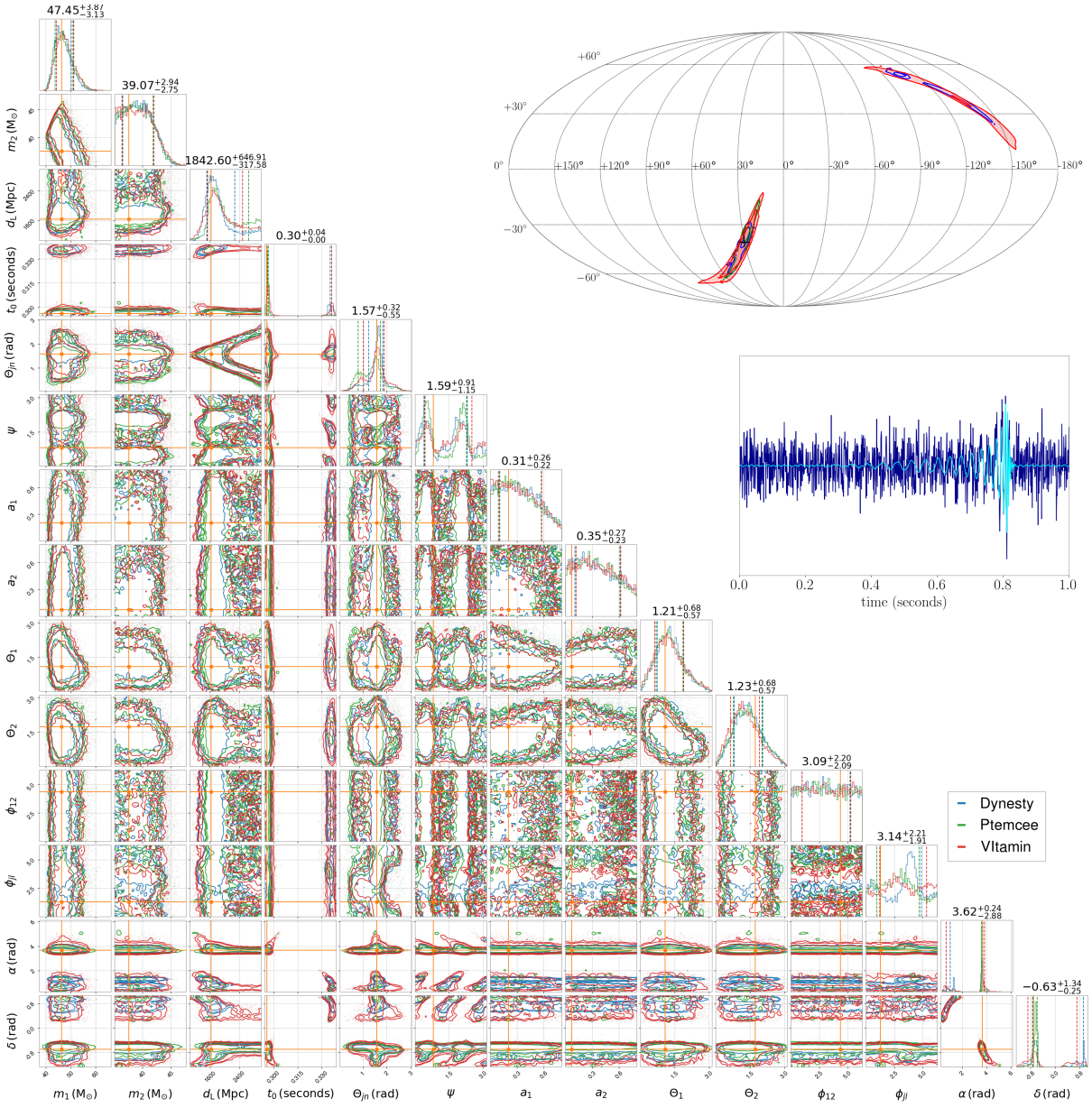


Figure 6.4: Corner plot showing 2 and 1-dimensional marginalised posterior distributions for the median SNR test dataset sample. Filled (red) contours represent the posteriors obtained from the CVAE approach and solid (blue) contours are the posteriors output from our baseline analysis (`Bilby` using the `Dynesty` sampler). In each case, the contour boundaries enclose 68, 90 and 95% probability. One dimensional marginalised posteriors for each parameter from both methods are plotted along the diagonal. Blue and red vertical lines represent the 5—95% symmetric confidence bounds for `Bilby` and variational inference respectively. Orange crosses and vertical orange lines denote the true parameter values of the simulated signal. The original whitened noisy timeseries y and the noise-free signal are plotted in blue and cyan respectively in the upper right hand panel. The test signal was simulated with network optimal signal-to-noise ratio of 10.89.

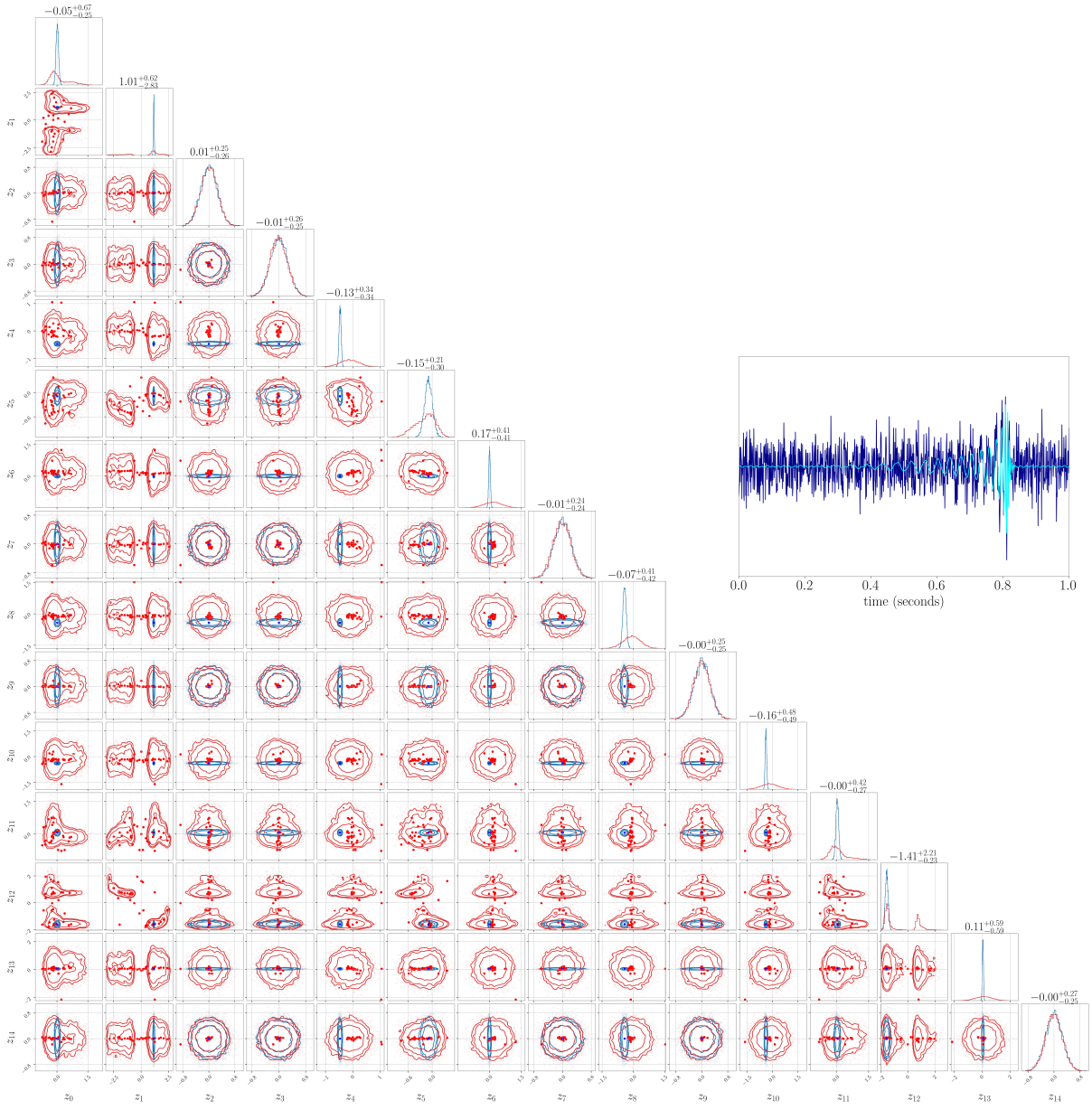


Figure 6.5: We show in this figure latent space samples from all latent space dimensions of both the q encoder network (blue) and the r_1 encoder network (red). Each point is representative of the predicted mean values for each latent space dimension (15). Each dimension on the horizontal and vertical axis represents a different latent space dimension. 1-dimensional histograms of latent space samples for each dimension are plotted along the diagonal. Contours represent the $68, 90, 95\%$ credibility intervals.

Mathematically, we can describe why latent space dimensions aren't being used when predicted distributions from r_1 and q are nearly identical by first expressing the theoretical probability distribution of $p(z|x,y)$ as

$$p(z|x,y) = \frac{p(x|z,y)p(z|y)}{p(x|y)}, \tag{6.1}$$

where $p(z|x,y) \approx q(z|x,y)$, $p(z|y) \approx r_1(z|y)$ and $p(x|z,y) \approx r_2(x|z,y)$. Here, we are assuming that we have reached a point where our neural network model has been successfully trained and we are under the scenario where the true theoretical distributions ($p(z|x,y)$, $p(z|y)$,$p(x|z,y)$) and those predicted by our network ($q(z|x,y)$, $r_1(z|y)$,$r_2(x|z,y)$) are modelling their distributions correctly. If we were to consider this equation given a fixed y and x (i.e. one test sample as in Fig. 6.5), then $p(x|y)$ becomes a constant as a function of the latent space location z and we can say

$$q(z) \propto r_1(z)r_2(z), \tag{6.2}$$

where $q(z)$, $r_1(z)$ and $r_2(z)$ are functions of z. Furthermore, if it turned out that for all dimensions of z, $q(z) \propto r_1(z)$ (i.e. if all blue and red distributions matched in Fig. 6.5) then therefore $r_2(z)$ would not be a function of z, but rather a constant. Since r_2 would not be a function of z in this case, posterior samples predicted by the r_2 decoder are also not a function of z, thus z is not being used at all to make those predictions. In other words, the connection between the encoders in Fig. 5.1 is broken and r_2 only gets access to the timeseries data.

We would also like to know if this holds when only individual dimensions of z are equivalent between q and r_1. We may determine this by first defining $z_{\text{tot}} = \{z_1, z'\}$, where z_1 is a single dimension in the latent space, z' is all other dimensions in the latent space and z_{tot} is the total set of latent dimensions. If we then marginalise the distribution $q(z_{\text{tot}})$ with respect to z' for the

left-hand side of Eq. 6.2

$$\begin{aligned}\int q(z_{\text{tot}})dz' &= \prod_{b=1}^{M} q^b(z_b) \\ &= \int q(z_1)q(z_2)q(z_3)\dots q(z_n)dz_2dz_3\dots dz_n, \\ &= q(z_1)\int\dots\int q(z_2)q(z_3)\dots dz_n\dots, \\ &= q(z_1), \end{aligned} \tag{6.3}$$

where there are M dimensions and we see that have been able to marginalise over all but one latent space dimension z_1. This is possible because the q encoder network models a diagonal covariance, so there are no correlations between dimensions. We now define the decoder network as a function of z as the product of r_2 over multiple dimensions, M, of z given as

$$r_2(z) = \prod_{a=1}^{M} r_2^a(z_a), \tag{6.4}$$

and also define the encoder network r_1 as a product over multiple latent dimensions and a summation over modes N of the latent space as

$$r_1(z) = \sum_{j=1}^{N} \alpha_j \prod_{k=1}^{M} r_1^{jk}(z_k), \tag{6.5}$$

where α is the mode weight. If we then marginalise $r_1(z_{\text{tot}})$ over z' for the right-hand side of

Eq. 6.2 and plug in Eq. 6.4 and Eq. 6.5 we get

$$\begin{aligned}
\int r_2(z_{\text{tot}})r_1(z_{\text{tot}})dz' &= \int \prod_{a=1}^{M} r_2^a(z_a) \sum_{j=1}^{N} \alpha_i \prod_{k=1}^{M} r_1^{jk}(z_k)dz', \\
&= \int r_2^1(z_1)[\alpha_1(r_1^{11}(z_1)r_1^{12}(z_2)\dots r_1^{1M}(z_M)) + \alpha_2(r_1^{21}(z_1)r_1^{22}(z_2)\dots r_1^{2M}(z_M)) \\
&\quad + \dots + \alpha_N(\dots r_1^{NM}(z_M))]r_2^2(z_2)dz_2 r_2^3(z_3)dz_3 \dots r_2^M(z_M)dz_M, \\
&= r_2^1(z_1)\alpha_1 r_1^{11}(z_1) \int r_1^{12}(z_2)dz_2 r_1^{13}(z_3)dz_3 \dots r_1^{1M}(z_M)dz_M \\
&\quad + r_2^1(z_1)\alpha_2 r_1^{21}(z_1) \int r_1^{22}(z_2)dz_2 r_1^{23}(z_3)dz_3 \dots r_1^{2M}(z_M)dz_M \\
&\quad + \dots + r_2^1(z_1)\alpha_2 r_1^{21}(z_1) \int \dots r_1^{NM}(z_M)dz_M, \\
&= r_2^1(z_1)[\alpha_1 r_1^{11}(z_1) + \alpha_2 r_1^{21}(z_1) + \dots + \alpha_N r_1^{N1}(z_1)], \\
&= r_2^1(z_1) \sum_{j=1}^{N} \alpha_i r_1^{j1}(z_1). \qquad (6.6)
\end{aligned}$$

Therefore, given Eq. 6.3 and Eq. 6.6, if one of our latent space dimensions predictions, z_1, from both the q encoder and the r_1 encoder are proportional to each other then

$$q(z_1) \propto \sum_{j=1}^{N} r_1^{j1}(z_1), \qquad (6.7)$$

therefore predictions from the decoder network r_2 are not a function of latent space dimension z_1 (i.e. $r_2(z_1) \neq f(z_1)$). This also means that the latent space dimension z_1 is not being used to pass information to the decoder network.

From Fig. 6.5 we see that 5 of the 15 latent space dimensions are not being used for this test case, where the 5 not being used from left to right on the horizontal axis are dimensions z_2, z_3, z_7, z_9, and z_{14}. This could possibly indicate that we do not need to use as many latent space dimensions as are being employed to accurately model the posterior. It also appears that the r_1 encoder network (red) is in-fact choosing to produce latent space samples which are representative of multimodal distributions on some dimensions (z_2 and z_{14}). This should not be surprising given that the r_1 encoder network's final output latent space distributions are generated using a Gaussian Mixture model, where the motivation for using the Gaussian mixture model was to encourage the r_1 network to encode different modes from the posterior directly in

the latent space. It is also evident from the 2D posterior panels (i.e. z_1 vs. z_4) that the mixture model in the r_1 encoder network is allowing for far more expressive non-Gaussian shaped uni-modal distributions.

We see show in Fig. 6.6 predicted weights from the r_1 encoder network for the median SNR test sample. It can be seen that the r_1 encoder network has assigned a measurable amount of probabilistic weight to 16 out of the 32 available latent space modes, with all other modes having essentially zero weight. The fact that there are several modes which are not assigned any weight at all may indicate that the Gaussian mixture model in the r_1 encoder network could have a higher level of capacity than what is be needed.

The effect that the integrated Gaussian mixture model in the r_1 network has on final posterior samples, is perhaps most clearly illustrated in Fig. 6.7. Here, we sample from the 4 most likely latent space modes predicted by the r_1 encoder (colored from highest to lowest likelihood as blue, orange, green and red), where "most likely" is quantified through the Gaussian mixture model weights. Posteriors for each mode are produced by using latent samples from each mode and passing them through the r_2 decoder network. What we are left with are posteriors which represent the degree to which individual modes in the `VItamin` latent space contribute to the final posterior product. We note that the weight associated with the probability of drawing a sample from each mode is $[2.785, 2.043, 1.768, 1.601]$. We see in Fig. 6.7, that the 2 most likely modes (blue, orange) agree strongly with each other across most dimensions (Θ_{jn} and ϕ_{jl} being possible exceptions), while the least likely 2 (green, red) also show strong disagreement on some parameters (t_0, ψ, α, δ). One might point out that the r_1 network has not chosen to represent each mode in the posterior on ψ using distinct modes in the latent space. We note that this is in-fact not surprising given that we apply a reparameterisation to the ψ and ϕ_0 parameters (Sec. 6.5) such that the number of modes the network actually "sees" is effectively reduced.

What the results from Fig. 6.7 may imply is that the multi-modality clearly seen in the final posteriors of Fig. 6.4 is being encoded across several modes in the latent space produced by r_1 encoder network. This could indicate that the r_1 encoder network is partially determining what level of probabilistic weight to assign to each mode in the posterior space. More work could be done in future studies to more rigorously quantify this.

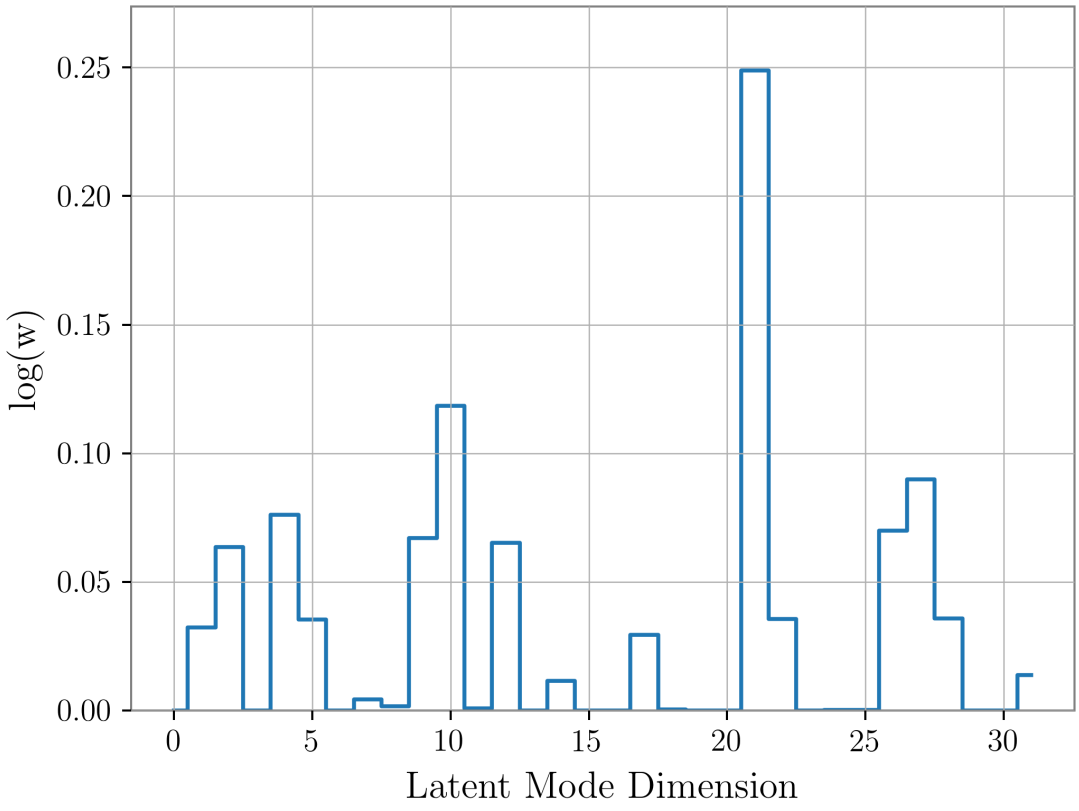


Figure 6.6: Plotted are the predicted weight values from the r_1 encoder network for a given posterior sample of the median SNR test case as a function of latent space mode dimension number. Each value is normalised such that the sum of the weights is 1, where 1 is representative of the network model assigning a high likelihood of sampling from that particular mode.

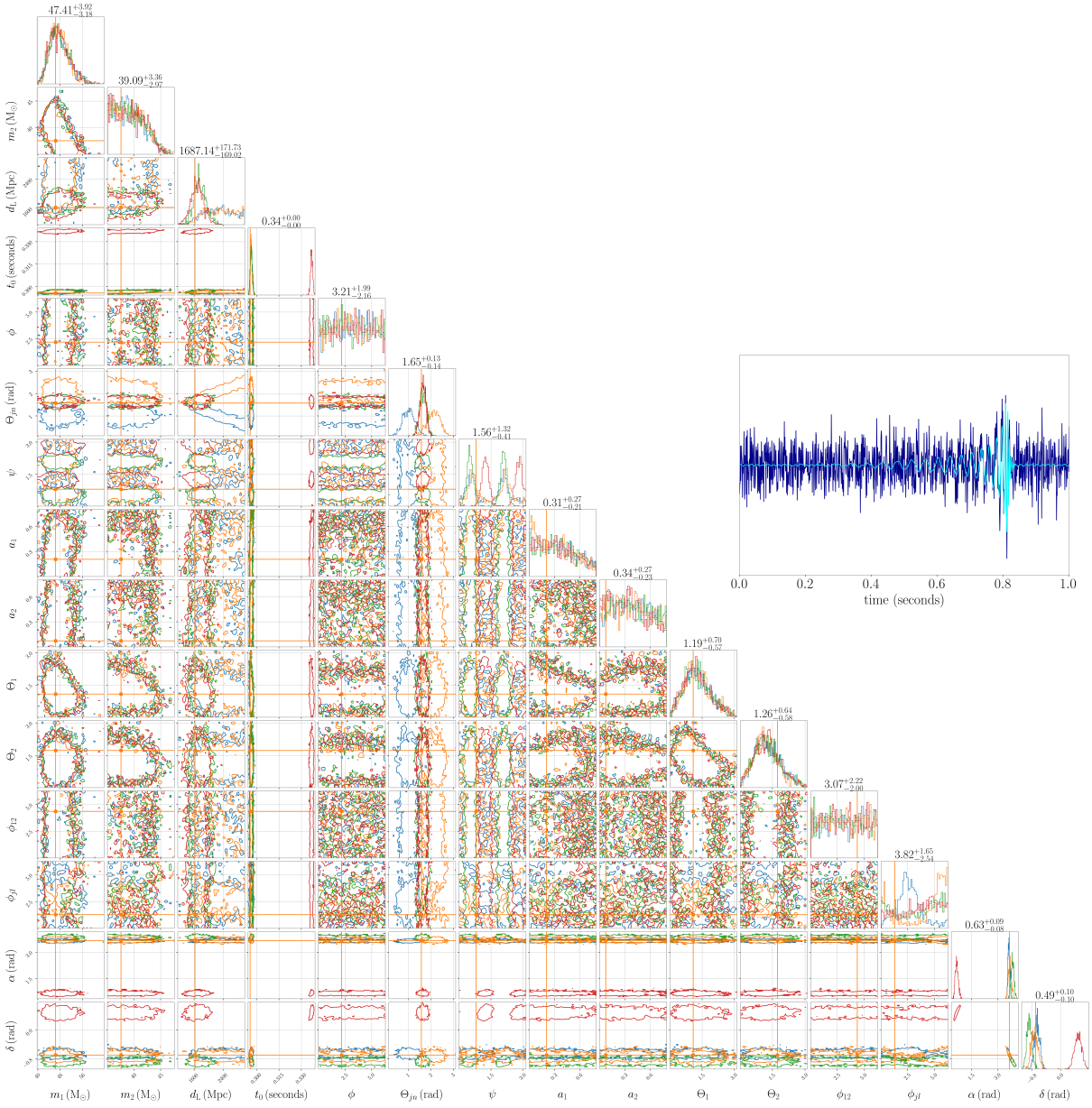


Figure 6.7: Shown are posterior samples drawn from the 4 most likely latent space modes predicted by the r_1 encoder network. Colors denote each mode from highest to lowest mean mode weight as blue, orange, green and red respectively. The log weight associated with each mode from most likely to least likely is given as $[2.785, 2.043, 1.768, 1.601]$. Each dimension on the x and y axis represents 15 source parameter posteriors predicted. 1-dimensional marginalised posteriors of posterior samples for each source parameter are plotted along the diagonal. Contours represent the $68, 90, 95\%$ credibility intervals. The orange vertical and and cross hairs represent the true source parameter values.

6.3 Dynesty vs. Dynesty Jensen–Shannon Divergence

Here we provide some discussion regarding the range of JS values we might expect from VItamin. We approximate a rough limit by comparing two independent Dynesty runs on all 250 test sample cases for both the full 14-dimensional JS-divergence and the 1-dimensional JS-divergence. We use Dynesty since the sampler is known within the GW parameter estimation community to be one of the most trusted and reviewed samplers [5].

In Fig. 6.8, we compute the JS-divergence between two independent runs of Dynesty for each of the 250 test cases across all 14 dimensions. The mean JS value across all test cases is 0.05 with tails extending to an upper bound of ~ 1, hence 0.05 is a measurement of our uncertainty in JS value error. Given that Dynesty is generally consistent between runs of the same data, we would expect that two independent runs of the Dynesty sampler should provide us with a reliable estimate on our fundamental JS results uncertainty. Observing the range of JS values seen in Fig. 6.8 and comparing those values to those of Fig. 5.10, we see that VItamin results plotted against other sampler results do generally seem to have larger JS values than the mean of 0.05 seen in Fig. 6.8. We also see that a sizeable fraction of the VItamin JS distributions do exceed the upper bound of the Dynesty vs. Dynesty JS results, though this trend is also seen in comparison results of other Bayesian samples vs. other Bayesian samplers. In all, this may indicate that there is room for improvement in VItamin results given that the VItamin JS values in Fig. 5.10 generally tend to exceed the fundamental mean JS uncertainty of 0.05.

In Fig. 6.9 we plot the JS-divergence between independent Dynesty runs for using 1-dimensional marginalised source parameter posterior results. Instead of using the universal-divergence code, we calculate the JS-divergence using the analytic expression defined in Eq. 5.23 on the 1-dimensional marginalised posteriors between the two Dynesty runs. In the figure, different colors represent different source parameter JS distributions over all 250 test cases. We see that JS-diverence values across all source parameter distributions range from $\sim 1 \times 10^{-4} - 10^{-1}$. The mean 1-dimensional JS value across all source parameter distributions is approximately $\sim 10^{-3}$. This value indicates a lower limit on the JS-divergence values we

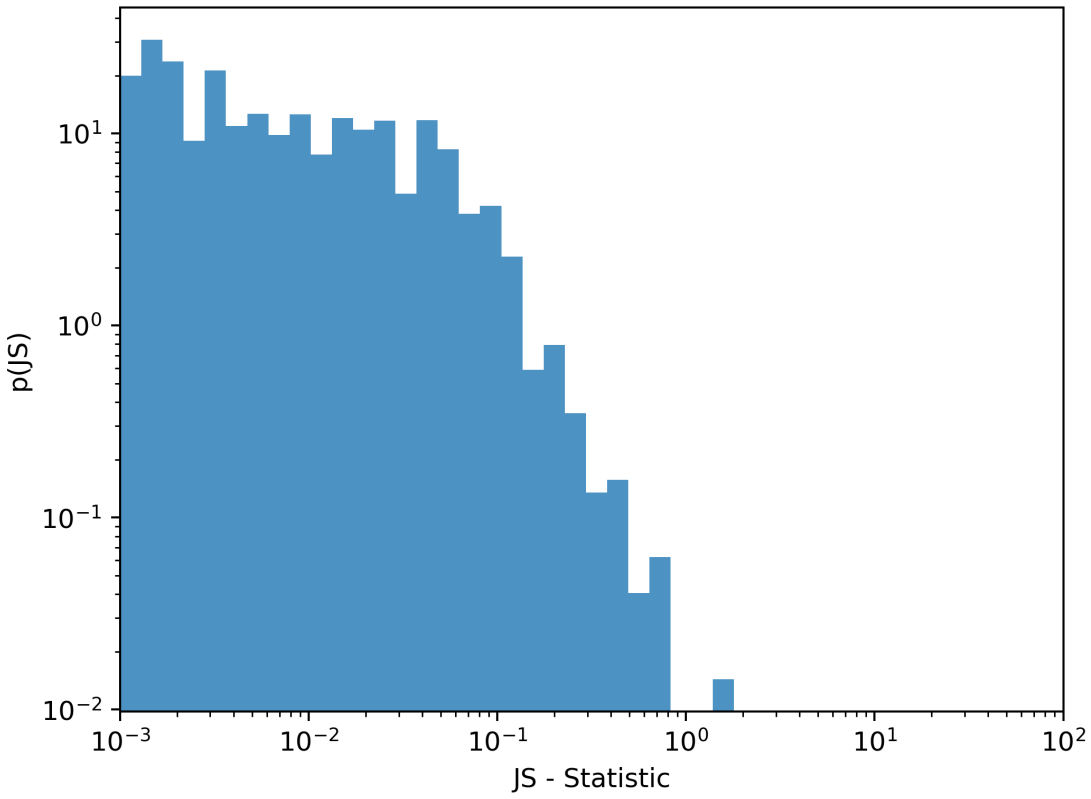


Figure 6.8: A probability distribution of JS-divergence values for `Dynesty` vs. another independent run of `Dynesty` using 14-dimensional posteriors on the same test data. The mean JS divergence has a value of 0.05 . We note that since since the JS-statistic is calculated using the `universal-divergence` [311] code-base, that there are some negative values which are not shown here.

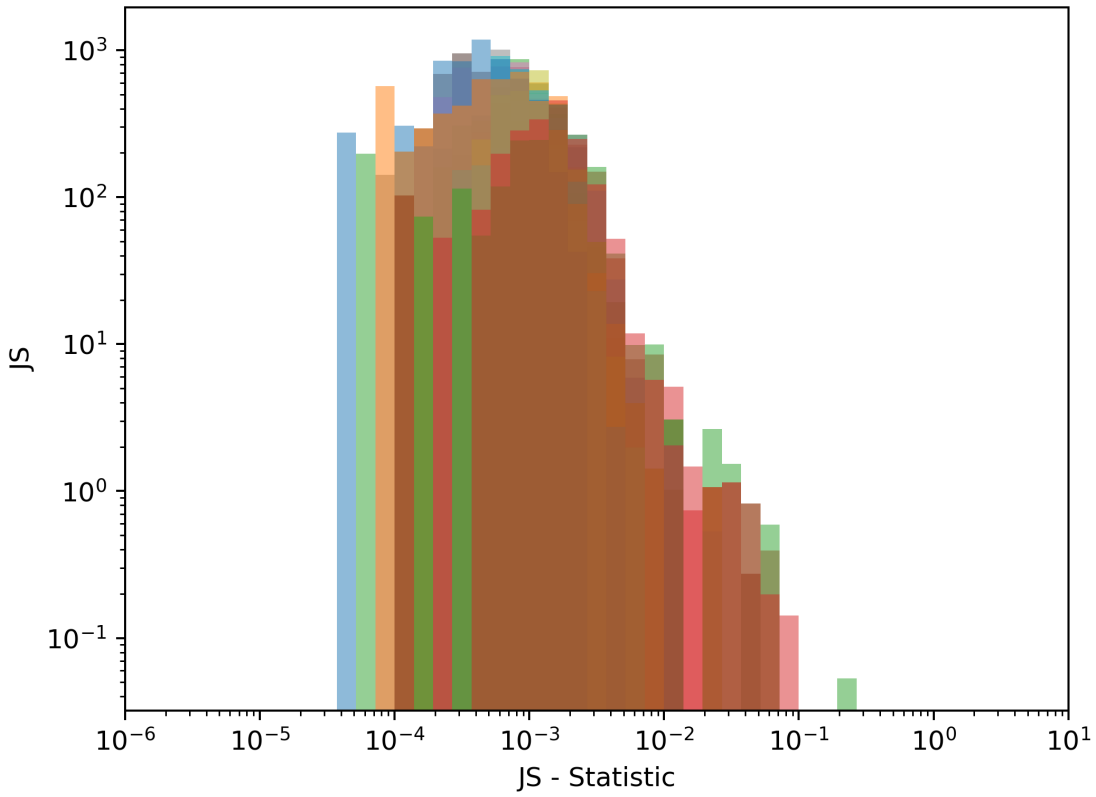


Figure 6.9: Shown are histograms of 1-dimensional JS divergence values for Dynesty vs. another independent run of Dynesty. Different colors represent JS values with respect to individual source parameter posterior Dynesty predictions. Mean JS values for each source parameter are given as: $m_1 \sim 0.00139$, $m_2 \sim 0.00143$, $d_l \sim 0.00142$, $t_0 \sim 0.00358$, $\theta_{jn} \sim 0.00192$, $\psi \sim 0.00134$, $a_1 \sim 0.00101$, $a_2 \sim 0.00088$, $\Theta_1 \sim 0.00143$, $\Theta_2 \sim 0.00123$, $\phi_{12} \sim 0.00080$, $\phi_{jl} \sim 0.00136$, $\alpha \sim 0.00538$, $\delta \sim 0.00401$. JS values are calculated using the scipy JS divergence code-base.

would expect to see with respect to comparison results using 1-dimensional marginalised posteriors. If we compare this mean value with Fig. 5.6 of Ch. 5 we see that the tails of VItamin vs. Dynesty 5th and 95th credibility regions lie either within or near the mean lower limit calculated above of 10^{-3}. Some exceptions to this include $t_0, \Theta_{jn}, \alpha, \delta$ where the tails are higher than this mean, and ϕ_{12} where the tails are lower.

6.4 Data Augmentation and Normalisation

As discussed previously in in Ch. 2, it is often advantageous to augment the training set during training of a neural network. This is done to decrease the complexity of the search space [162], as well as to provide a greater variety of signals to the network such that it is better able to generalise to new signals when testing the model.

A form of pre-processing that we employ is that of normalisation. We normalise each source parameter value (x) for all training samples such that they lie between the values of zero and one (i.e. on the unit-hypercube). We also normalise all timeseries (y) in the training set using a normalisation factor such that all timeseries values are safely contained within the range of $(-1,1)$. We note that this timeseries normalisation factor must also then be applied during testing when using a pre-trained neural network. Both of these normalisations are performed in order to reduce the search space complexity and to avoid exploding gradients (Ch. 2, Sec. 2.2).

We also convert the right ascension source parameter values to the hour angle coordinate system. This is done because the right ascension of a source is measured with respect to it's position on the celestial sphere. For a given GW signal, if we were to fix all source parameters except for the sky location and change it's GPS time, we would be left with a different form of the signal in the data since the Earth rotates, and thus so will the relative position between the source and the detectors. We can avoid this issue by reparameterising the signal such that its right ascension is now an hour angle given by

$$\alpha_{\mathrm{HA}} = (t_{\mathrm{GMT}} - \alpha) \bmod 2\pi, \tag{6.8}$$

where α_{HA} is the hour angle, t_{GMT} is the Greenwich mean sidereal time of the GW signal arrival time, "mod" is the modulus, and α is the right ascension. This reparameterisation makes it such that different GPS times for a fixed GW signal does not change the form of the signal in the data when also changing the GPS time.

In order to make it easier for the network to predict cyclic parameter values, we reparameterise all cyclic parameter values to be represented on the abstract 2D plane. The motivation here is to help the network realise that the bounds which define the cyclic parameters are actually at

equivalent locations. The conversion is done by enforcing the decoder network to produce 2 means and 1 standard deviation characterising multivariate Gaussians for each cyclic parameter (as opposed to the 1 mean for all other source parameters). [1] The angle is then computed between the 2 predicted cyclic parameter means through the inverse tangent function. The inverse tangent converts the 2-dimensional representation back to the original parameter space for all cyclic parameters.

Another augmentation technique that we apply is to allow the network to see multiple noise realizations of the same signal multiple times. This is done by first enforcing that the network be run over a subset of the entire training set, 2×10^4 unique training sample waveforms, 4 times. The cost function of the network is calculated by drawing a random batch of signals from the current training subset of 2×10^4 signals, whereby each signal in the batch is assigned a new white Gaussian noise realisation. This means that we effectively train over an infinite number of Gaussian noise realisations. After the network has seen 2×10^4 training signals 4 times we then load in a new subset of 2×10^4 training signals. By giving the network multiple noise realisations for the same signals, we are hopefully encouraging the network to generalise to new noise realisations during testing.

Every time we give the network a new chunk of 2×10^4 signals, we also randomize the phase, time of arrival and distance of the new loaded in training samples. We note that the new chunk of signals is read in already having existing values which have been drawn from the prior. This process is to relabel the source parameter values with new draws from the prior and to also modify the associated noise-free timeseries of the training samples accordingly. For distance, we first choose values uniformly at random from 0 to 1 for each distance training sample parameter. These values are then converted to units of Mpc by

$$d_{\rm new} = d_{\rm min} + u_d(d_{\rm max} - d_{\rm min}), \tag{6.9}$$

where u_d is a uniform set of numbers between 0 and 1, $d_{\rm max}$ represents the maximum allowed distance according to the prior and $d_{\rm min}$ is the minimum allowed distance according to the prior.

[1] The predicted standard deviation in this case is the concentration parameter which partially characterises the decoder output von Mises distributions used to represent the cyclic parameter posterior distributions.

We sample d_{uni} from a uniform distribution simply because that is the prior we use for the luminosity distance. We then determine the scale factor by which the distance has changed from its old value for each training sample by dividing the old distance by the new distance value

$$S = \frac{d_{\text{old}}}{d_{\text{new}}}, \tag{6.10}$$

where d_{old} is the original distance value for the sample and d_{new} is the new value. In summary, all we're doing here is drawing a new distance from the prior, and then computing a ratio between the old and new distance.

In order to get the phase augmentation correction term we do a similar process as the distance augmentation above. We begin by drawing a new phase value from the prior. A phase correction term is then calculated

$$\Gamma_{\phi_0}^{\text{corr}} = -\exp\left(i(\phi_0^{\text{new}} - \phi_0^{\text{old}})\right), \tag{6.11}$$

where $\Gamma_{\phi_0}^{\text{corr}}$ is the phase correction factor we will use to randomize the phase, ϕ_0^{new} is the new randomized phase value and ϕ_0^{old} is the original training sample phase value.

The time of coalescence correction term is then computed by first randomly drawing a new time of coalescence from the prior. We then find the difference between the new and old times and compute a correction factor applied in the frequency domain given by the expression

$$\mathbf{\Gamma}_{t_0}^{\text{corr}} = -\exp\left(2\pi i \boldsymbol{f}_t (t_0^{\text{new}} - t_0^{\text{old}})\right) \tag{6.12}$$

where t_0^{new} is the new time of coalescence, t_0^{old} is the old time of coalescence, $\boldsymbol{f}_t$ is a frequency vector with values from 0 to the Nyquist frequency, and $\mathbf{\Gamma}_{t_0}^{\text{corr}}$ is the time of coalescence correction factor. Finally, given all the correction factors for time of coalescence $\mathbf{\Gamma}_{t_0}^{\text{corr}}$, distance scale factor S and phase at coalescence $\Gamma_{\phi_0}^{\text{corr}}$ have been calculated, we need only simply multiply the phase correction term and the time correction term by the real FFT [98] of the training sample timeseries, y, where we note the time correction term is frequency dependent and the phase correction term is a constant. The application of the time and phase correction terms may be

expressed as

$$\tilde{y}_{\Gamma\text{corr}} = \mathscr{F}(y)\,\Gamma^{\text{corr}}_{\phi_0}\,\mathbf{\Gamma}^{\text{corr}}_{t_0}, \tag{6.13}$$

where $\mathscr{F}$ is the FFT. We note here that the time correction term effectively "slides" the signal in time, such that the end of the signal can be slid past the end of the timeseries and end up at the start of the timeseries, and vice versa. This is obviously unphysical, but we emphasise that our signals are constructed so that they do not have significant amplitude at the boundaries of our timeseries. Additionally, the time of coalescence window defined by the prior is relatively narrow, hence we expect no unphysical signal wrapping.

We then compute the inverse real FFT [98], $\mathscr{F}^{-1}$, of $\tilde{y}_{\Gamma\text{corr}}$ in order to get back to the time domain and also multiply by the distance correction scale factor

$$y_{\Gamma\text{corr}} = \mathscr{F}^{-1}(\tilde{y}_{\Gamma\text{corr}})\,S. \tag{6.14}$$

Adding these randomized elements to the existing training workflow may help to ensure that the neural network model does not overfit the training set. We also note that these augmentations were a computationally cheap and simple method for expanding the effective training set size.

6.5 Phase and Polarisation Reparameterisation

One of the biggest issues we have faced while training the neural network has been dealing with the complex multi-modal nature of the phase (ϕ) and polarisation (ψ) parameters. Along with the addition of the Gaussian mixture model component of the network mentioned previously, we have also implemented a reparameterization of both phase and polarisation angle in order to simplify the search space for the neural network partly inspired by the work of Jones in [321].

We are allowed to make the following reparameterisation due to the degeneracies in ψ and ϕ_0 of the signal model we use, where degeneracy refers to different ψ, ϕ_0 combinations which give rise to the same GW waveform. In order to go from ψ and ϕ to a new representation ψ' and X, we first define X as

$$X = (\psi + \phi_0) \bmod \pi. \tag{6.15}$$

We then define $\psi^{'}$ as

$$\psi^{'} = \psi \bmod \frac{\pi}{2}. \tag{6.16}$$

The reparameterisation process is visually illustrated in Fig. 6.10. It can be seen that the 2-dimensional representation $X, \psi^{'}$ in both the upper left and lower right subplots, is vastly simpler than the original 2-dimensional ϕ_0, ψ representation in the lower left. The transformation is also able to maintain the property of being reversible. We note that this process does not guarantee that transforming a sample forward and backward would put it in the same location. A forward transformation maps the sample to the $X, \psi^{'}$ space (i.e. upper left plot of Fig. 6.10), but a backwards transformation will equally randomly place the sample in one of the 4 other tessellations of the $(X - \psi^{'}), \psi^{'}$ shape (upper right plot of Fig. 6.10) in the original ϕ, ψ space (lower left plot of Fig. 6.10).

In order to get back to the original ψ, ϕ_0 representation, we first subtract off $\psi^{'}$ from X (upper left to upper right plot of Fig. 6.10). We then need to convert the $(X - \psi')$ parameter to the ϕ_0 space and can do so by choosing a uniform random variable with equal probability between zero and 2π, denoted as D_1, as well as one between 0 and $\pi/2$, denoted as D_2, and then adding D_1 and D_2 to $(X - \psi')$ (upper right to lower left plot in Fig. 6.10). This process is given by the expression

$$\phi_0 = \left((X - \psi^{'}) + D_1 + D_2\right) \bmod 2\pi \tag{6.17}$$

To get back to ψ we add D_2 to ψ' given as

$$\psi = (\psi^{'} + D_2) \bmod \pi. \tag{6.18}$$

As seen going from the upper right plot to the lower left plot of Fig. 6.10, the distribution on the full physical ψ, ϕ_0 space (lower left plot) is just 4 copies of the $\psi^{'}$ and $(X - \psi^{'})$ shape (upper right plot) tessellated in the ϕ_0,ψ space. We also highlight that if one considers GW template waveforms with higher order modes this degeneracy is broken and the above reparameterisation would not be valid [321].

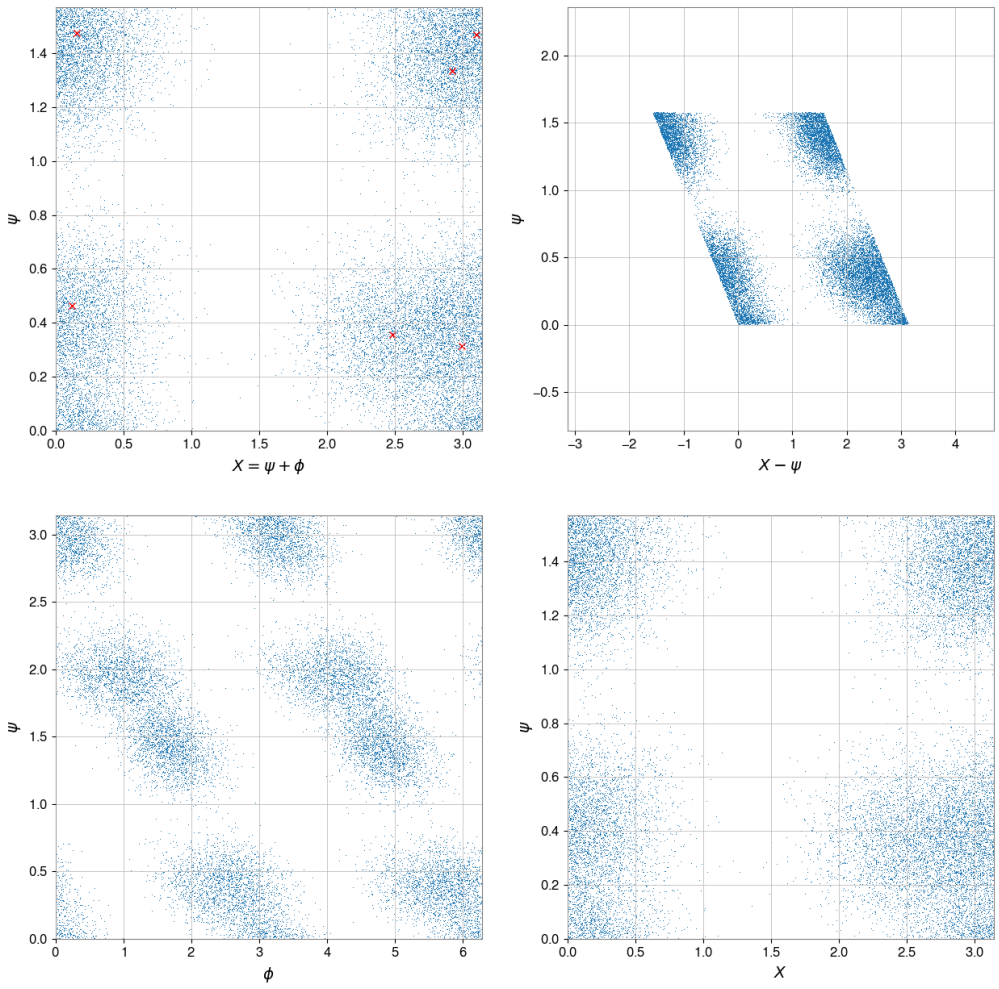


Figure 6.10: An illustration of the polarisation angle and phase reparameterisation process. Upper left plot: representative of samples in the ψ' and X space where distributions are randomly generated according to a 6 component Gaussian mixture model. Red crosses denote each Gaussian's component mean. Upper right: Samples from the $\psi^{'}, X$ space are converted to the $\psi^{'}$, $(X - \psi^{'})$ space. Lower left: We then apply a tessellation of the samples (i.e. $(X - \psi^{'}), \psi^{'}$ shape) in the upper right figure using the D_1 and D_2 randomisation process to get samples in the original ψ, ϕ_0 space. Lower right: We can get back to the ψ' and X space by applying our reparameterisation process again and see that it is identical to the original upper left plot.

6.6 Summary

In this chapter we begin by outlining several studies which investigated the relationship between the JS divergence of VItamin results vs. Dynesty results as a function of SNR. It was found that there was little positive correlation between the two, but instead there was slight negative correlation.

There was also some studies done on the VItamin latent space for a given test sample. It was found that not all latent space dimensions are being used for this test sample and it was shown mathematically that dimensions not used imply not information is being passed to the decoder network to produce posterior samples.

We then end the chapter discussing the expected lower limit on JS values of VItamin vs. other samplers. We also outline several data augmentation and normalisation techniques which we use in order to optimise the training of the VItamin network.

Chapter 7

Conclusions and Future Work

In this thesis it was shown for the first time how machine learning may be used for both GW signal detection and parameter estimation. Whilst current state-of-the-art algorithms used by the LVK are optimal to a certain degree (e.g. matched filtering) or optimal but slow (e.g. Bayesian inference) for both the detection and parameter estimation of GWs, the number of signals expected to be seen in future observation runs have the potential to tax the computational resources needed for continued use of such techniques. The work in this thesis has shown several first of its kind proof-of-principal studies which illustrate that ML techniques are a powerful and accurate tool for signal detection and parameter estimation.

In Ch. 1 the basic concept of GR and how GWs arise from that concept was introduced. The standard techniques for GW detection and parameter estimation (matched template filtering and Bayesian inference respectively) was discussed. It was shown how the LIGO/Virgo detectors detect GWs, as well as account for various noise sources during operation. It was also discussed that while matched filtering and Bayesian parameter estimation are generally efficient and reliable, they can also be computationally expensive. An overview was provided on GW detections made by the LVK up to the present day and how future observation runs will produce an abundance of GW signals. Given the expected number of signals in future observing runs, it was made clear in this introductory chapter that there is a need for efficient low-latency signal detection and parameter estimation methods.

In Ch. 2 the simple concept of a perceptron neural network was introduced. From there, it

was shown how the concept of a perceptron may be expanded to include multiple perceptrons in a layer and multiple layers of perceptrons into a deep neural network. It was also discussed how a deep neural network is trained and how one may then construct other neural network types, such as CNNs, AEs, VAEs and CVAEs.

In Ch. 3 a survey of various ML techniques employed in GW astronomy was performed. It was seen that ML has had a surge in use in only the recent past and has been applied to a variety of tasks including glitch classification, GW population analysis, GW Bayesian parameter estimation and point estimate parameter estimation, signal detection and PSD estimation. Given the wide scope of tasks in which neural networks have been applied across the whole of GW astronomy, it is perhaps an indication that ML will play an increasingly pivotal role in the LVK over the coming years.

In Ch. 4 I presented my study in which I used CNNs to perform GW signal detection. Given that the expected number of detections made by the LVK is expected to increase dramatically in the coming years, there is an urgent need for faster signal detection techniques. A CNN, is one such technique which is able to produce predictions on signal classes in low-latency. This method also has the potential to provide added flexibility to be trained on real noise, thus allowing CNN to directly learn the noise distribution and detector artefacts. In my study I trained a CNN model on BBH signals buried in whitened Gaussian noise. The results from the CNN model were compared to those from the standard signal detection method used in the ground based GW community (matched filtering). Results between both the ML approach and the matched filtering were seen to be in strong agreement with each other. These results showed for the first time that deep learning could match the efficiency of existing signal detection techniques and as such contributed to an increased level of confidence within the GW astronomy community that deep learning could be a powerful tool.

In Ch. 5 and Ch. 6 I presented my study in which I used CVAEs to produce Bayesian posteriors on 15 BBH parameters accepted in Nature Physics. Our CVAE model (`VItamin`) was trained on whitened BBH signals injected in Gaussian noise. I also produced benchmark results using known and trusted Bayesian nested sampling and MCMC samplers (`Dynesty`, `Ptemcee`, `CPNest`, `Emcee`). Results from both `VItamin` and Bayesian samplers were com-

pared against each other using many different figures of merit such as the JS-divergence between posteriors and p-p plots. It was shown that results from the ML approach had similar levels of agreement as those between Bayesian samplers against other Bayesian samplers. It was also shown that the speed of `VItamin` was ~ 6 orders of magnitude greater than those of the Bayesian samplers, producing posteriors in less than 1 s. Given the speed gains from this approach and the accuracy of CVAEs with respect traditional samplers, it is evident that CVAEs may be useful for low-latency EM follow-up analyses. The low-latency data products from CVAEs will allow for the observation of prompt EM emission on timescales which are far shorter than existing Bayesian sampling techniques. It was also stated in this chapter that the ML model used may be extended to a variety of other source types and may also in future work be trained/tested using real non-Gaussian detector noise.

ML has been used in countless applications across a wide variety of fields including: economics [322, 323, 324], biology [325, 326, 327] , chemistry [328, 329, 330] and physics [331, 332, 333]. GW astronomy has only in the past several years seen an increase in the number of studies which apply ML to a wide variety of applications including: GW signal detection, GW parameter estimation, and many other studies (See Ch. 3). Several first proof-of-principal studies were shown which proved that ML can match the accuracy of standard trusted techniques currently used in the collaboration for both signal detection and parameter estimation. Going forward, there are many future directions that could be taken with this work.

At the time of publication for the signal detection paper [1], it was mentioned that one could apply our method (or similar ML models) to sources other than BBH signals including: BNS, NSBH, CW and stochastic GW events. In the subsequent years, there have been many follow-up studies which have demonstrated success using a variety of ML algorithms for BNS signal detection [206, 208, 334], BNS early-warning signal detection [335], CW detection [221, 336], and stochastic background detection [337]. Moreover, it has been shown that unmodelled ML approaches may also be used to detect burst-like signals [218, 338]. Given the success of many of the aforementioned studies, there is still much work left to be done in the regime of testing such algorithms on real data in real noise (including our own work). There is also much interest in the community with regards to the FARs of such algorithms and there should be more studies

done concerning what classes of spurious non-astrophysical signals may "trick" ML models into outputting false detections. I also mentioned the possibilities of using CNNs for point-estimate parameter estimation in my signal detection paper [1], but acknowledge that this may not be the most useful when compared with recent developments in ML for Bayesian parameter estimation which returns the posterior directly.

There have also been many recent developments in the field of ML hyperparameter optimisation [339, 340, 341] (Ch. 2, Sec. 2.4), and such optimisation algorithms have become more widely available and easier to use [342]. It would be interesting to see whether the use of such algorithms can outperform a random hyperparameter search used by most GW astronomy ML practitioners today. It should also be noted that the gap must be bridged between proof-of-principal studies to the development of easy-to-use packaged ML GW detection pipelines in order to make the most use of these approaches.

Following the initial release of my study on Bayesian parameter estimation using CVAEs, there has been a flurry of follow-up studies which have used many alternative ML techniques to produce similar data products. Such work includes those done by [234] using reduced order modeling and perceptron networks and also [236] using normalising flows. There has also been a study which uses hybrid normalising flows - nested sampling algorithms in order to boost sampler convergence times [241], as well as a study using CVAEs which have built upon my own work [243]. It was also noted that there has been exciting recent developments which have shown for the first time that ML models (normalising flows) may be used to accurately estimate Bayesian posteriors for real GW signals in real noise [239]. Applying variational inference algorithms such as CVAEs and normalising flows to BNS signals should also be explored, given the expected increase in detected GW signals by the LVK. Some issues that will need to be overcome will be the large length of BNS signals ($\sim$ 100s or more depending on the lower cut-off frequency) and the large standard sampling rate of preprocessed data used by data analysis pipelines ($\sim$ 8kHz), where the computational power needed to train may be large. Possible solutions could be to use similar techniques as those employed by [236] like principal component analysis to reduce the dimensionality of the GW timeseries, or those employed by [208] who used varying sample rates at different intervals of the input signal. As mentioned earlier in Ch. 6

Sec. 6.4, data augmentation to phase, distance and time of coalescence was also applied. It should be noted that this form of signal augmentation can be extended in future work to include more parameters and may therefore reduce the need for a large training set.

One issue that was not addressed in this thesis is accounting for an unknown (static) PSD, or an estimated PSD as conditional input to the `VItamin` network. It is also known that the PSD of GW detectors varies as a function of time over the course of a given observation run. A changing noise floor can have adverse impact on the performance of neural networks which have been pre-trained on a PSD which does not encompass such variance. One solution may be to provide many different PSD realisations within the bounds of what the variance of the PSD is to be expected over the course of an observing run as input to the neural network during training, thus marginalizing over the PSD. Careful decisions would then have to be made as to how often one should re-train such a model as an observing run progresses. It should also be acknowledged that there is the option to use the estimated PSD for each training/validation/testing sample as a conditional input together with the data, as was done by [294] using normalising flows.

Re-training periodically may also benefit from the use of transfer learning [162] whereby previous iterations of the neural network model are used as the initialised values for the new neural network's weights. It is also important to produce packaged, reviewed pipelines of variational inference GW algorithms, so that their benefits may be more fully utilised by others in the GW astronomical community. Extending the `VItamin` analysis presented in Ch. 5, to larger prior ranges, as well as increasing the sample rate will be necessary in order to confirm the degree to which the computational cost may or may not increase for more complicated signals.

It is also acknowledged that one of the limiting factors of my own work has been the use of Gaussian noise, rather than realistic non-Gaussian noise. If the ML models used by myself for signal detection and parameter estimation were trained on real data, then (in principle) the ML model would be able to learn the true likelihood and not be limited to the analytically imposed version. Assuming this to be the case, it may be possible that ML will be able to exceed the performance of existing methods in non-Gaussian noise both in terms of speed and accuracy [203, 204]. It is my hope that the next generation of Ph.D. students and/or postdocs will apply CVAEs under realistic noise conditions and test on real GW signals in future work.

It is my hope that my own work will continue to be expanded upon and serve as a foundation for future studies which aim to provide rapid ML data products, such as signal detection statistics and Bayesian posteriors, in real time to scientists both inside and outside of the LVK. Given that ML studies for GW astronomy have largely focused on CBC events, possibly due in-part to the fact that they are the only sources detected so far, I believe that ML will also have a crucial role to play regarding sources yet to be detected or remain unmodelled. It is my belief that rapid data products from ML will only enhance current and future analyses, hopefully continuing to advance the field of GW data analysis.

Bibliography

[1] Hunter Gabbard, Michael Williams, Fergus Hayes, and Chris Messenger. Matching matched filtering with deep networks for gravitational-wave astronomy. *Phys. Rev. Lett.*, 120:141103, Apr 2018. doi: 10.1103/PhysRevLett.120.141103. URL https://link.aps.org/doi/10.1103/PhysRevLett.120.141103.

[2] Hunter Gabbard, Chris Messenger, Ik Siong Heng, Francesco Tonolini, and Roderick Murray-Smith. Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy. *arXiv e-prints*, art. arXiv:1909.06296, September 2019.

[3] B. P. Abbott et al. Observation of gravitational waves from a binary black hole merger. *Phys. Rev. Lett.*, 116:061102, Feb 2016. doi: 10.1103/PhysRevLett.116.061102. URL https://link.aps.org/doi/10.1103/PhysRevLett.116.061102.

[4] B. P. Abbott et al. Gw151226: Observation of gravitational waves from a 22-solar-mass binary black hole coalescence. *Phys. Rev. Lett.*, 116:241103, Jun 2016. doi: 10.1103/PhysRevLett.116.241103. URL https://link.aps.org/doi/10.1103/PhysRevLett.116.241103.

[5] R. Abbott et al. Gwtc-2: Compact binary coalescences observed by ligo and virgo during the first half of the third observing run. *Phys. Rev. X*, 11:021053, Jun 2021. doi: 10.1103/PhysRevX.11.021053. URL https://link.aps.org/doi/10.1103/PhysRevX.11.021053.

[6] Samantha A Usman, Alexander H Nitz, Ian W Harry, Christopher M Biwer, Duncan A

Brown, Miriam Cabero, Collin D Capano, Tito Dal Canton, Thomas Dent, Stephen Fairhurst, Marcel S Kehl, Drew Keppel, Badri Krishnan, Amber Lenon, Andrew Lundgren, Alex B Nielsen, Larne P Pekowsky, Harald P Pfeiffer, Peter R Saulson, Matthew West, and Joshua L Willis. The pycbc search for gravitational waves from compact binary coalescence. *Classical and Quantum Gravity*, 33(21):215004, 2016. URL http://stacks.iop.org/0264-9381/33/i=21/a=215004.

[7] B. P. Abbott et al. Gwtc-1: A gravitational-wave transient catalog of compact binary mergers observed by ligo and virgo during the first and second observing runs. *Phys. Rev. X*, 9:031040, Sep 2019. doi: 10.1103/PhysRevX.9.031040. URL https://link.aps.org/doi/10.1103/PhysRevX.9.031040.

[8] A. Buikema et al. Sensitivity and performance of the advanced ligo detectors in the third observing run. *Phys. Rev. D*, 102:062003, Sep 2020. doi: 10.1103/PhysRevD.102.062003. URL https://link.aps.org/doi/10.1103/PhysRevD.102.062003.

[9] B. P. Abbott et al. Gw170817: Observation of gravitational waves from a binary neutron star inspiral. *Phys. Rev. Lett.*, 119:161101, Oct 2017. doi: 10.1103/PhysRevLett.119.161101. URL https://link.aps.org/doi/10.1103/PhysRevLett.119.161101.

[10] B. P. Abbott et al. Gravitational waves and gamma-rays from a binary neutron star merger: GW170817 and GRB 170817a. *The Astrophysical Journal*, 848(2):L13, oct 2017. doi: 10.3847/2041-8213/aa920c. URL https://doi.org/10.3847/2041-8213/aa920c.

[11] B.P. Abbott et al. A gravitational-wave standard siren measurement of the Hubble constant. *Nature*, 551(7678), 2017. ISSN 14764687. doi: 10.1038/nature24471.

[12] B P Abbott et al. A guide to LIGO–virgo detector noise and extraction of transient gravitational-wave signals. *Classical and Quantum Gravity*, 37(5):055002, feb 2020. doi:

10.1088/1361-6382/ab685e. URL https://doi.org/10.1088/1361-6382/ab685e.

[13] Hsin-Yu Chen, Daniel E. Holz, John Miller, Matthew Evans, Salvatore Vitale, and Jolien Creighton. Distance measures in gravitational-wave astrophysics and cosmology. *Classical and Quantum Gravity*, 38(5):055010, March 2021. doi: 10.1088/1361-6382/abd594.

[14] Lee Samuel Finn and David F. Chernoff. Observing binary inspiral in gravitational radiation: One interferometer. *Phys. Rev. D*, 47:2198–2219, Mar 1993. doi: 10.1103/PhysRevD.47.2198. URL https://link.aps.org/doi/10.1103/PhysRevD.47.2198.

[15] R. Abbott et al. Observation of gravitational waves from two neutron star–black hole coalescences. *The Astrophysical Journal Letters*, 915(1):L5, jun 2021. doi: 10.3847/2041-8213/ac082e. URL https://doi.org/10.3847/2041-8213/ac082e.

[16] Clifford E. Rhoades and Remo Ruffini. Maximum Mass of a Neutron Star. , 32(6): 324–327, February 1974. doi: 10.1103/PhysRevLett.32.324.

[17] John Antoniadis, Thomas M. Tauris, Feryal Ozel, Ewan Barr, David J. Champion, and Paulo C. C. Freire. The millisecond pulsar mass distribution: Evidence for bimodality and constraints on the maximum neutron star mass. *arXiv e-prints*, art. arXiv:1605.01665, May 2016.

[18] B. P. Abbott et al. Prospects for observing and localizing gravitational-wave transients with Advanced LIGO, Advanced Virgo and KAGRA. *Living Reviews in Relativity*, 21 (1):3, Apr 2018. doi: 10.1007/s41114-018-0012-9.

[19] Gregory Ashton, Moritz Huebner, Paul D. Lasky, Colm Talbot, Kendall Ackley, Sylvia Biscoveanu, Qi Chu, Atul Divarkala, Paul J. Easter, Boris Goncharov, Francisco Hernandez Vivanco, Jan Harms, Marcus E. Lower, Grant D. Meadors, Denyz Melchor, Ethan Payne, Matthew D. Pitkin, Jade Powell, Nikhil Sarin, Rory J. E. Smith, and Eric Thrane. Bilby: A user-friendly bayesian inference library for gravitational-wave astronomy. *Astrophysical Journal Supplement Series*, 2018. doi: 10.3847/1538-4365/ab06fc.

[20] John Veitch, Vivien Raymond, Benjamin Farr, Will M. Farr, Philip Graff, Salvatore Vitale, Ben Aylott, Kent Blackburn, Nelson Christensen, Michael Coughlin, Walter Del Pozzo, Farhan Feroz, Jonathan Gair, Carl-Johan Haster, Vicky Kalogera, Tyson Littenberg, Ilya Mandel, Richard O'Shaughnessy, Matthew Pitkin, Carl Rodriguez, Christian Röver, Trevor Sidery, Rory Smith, Marc Van Der Sluys, Alberto Vecchio, Will Vousden, and Leslie Wade. Robust parameter estimation for compact binaries with ground-based gravitational-wave observations using the lalinference software library. *Physical Review D*, 2014. doi: 10.1103/PhysRevD.91.042003.

[21] LIGO Scientific Collaboration, Virgo Collaboration, F. GBM, INTEGRAL, IceCube Collaboration, AstroSat Cadmium Zinc Telluride Imager Team, IPN Collaboration, The Insight-Hxmt Collaboration, ANTARES Collaboration, The Swift Collaboration, AGILE Team, The 1M2H Team, The Dark Energy Camera GW-EM Collaboration, the DES Collaboration, The DLT40 Collaboration, GRAWITA, :, GRAvitational Wave Inaf TeAm, The Fermi Large Area Telescope Collaboration, ATCA, :, A. Telescope Compact Array, ASKAP, :, A. SKA Pathfinder, Las Cumbres Observatory Group, OzGrav, DWF, AST3, CAASTRO Collaborations, The VINROUGE Collaboration, MASTER Collaboration, J-GEM, GROWTH, JAGWAR, C. NRAO, TTU-NRAO, NuSTAR Collaborations, Pan-STARRS, The MAXI Team, T. Consortium, KU Collaboration, N. Optical Telescope, ePESSTO, GROND, T. Tech University, SALT Group, TOROS, :, Transient Robotic Observatory of the South Collaboration, The BOOTES Collaboration, MWA, :, M. Widefield Array, The CALET Collaboration, IKI-GW Follow-up Collaboration, H. E. S. S. Collaboration, LOFAR Collaboration, LWA, :, L. Wavelength Array, HAWC Collaboration, The Pierre Auger Collaboration, ALMA Collaboration, Euro VLBI Team, Pi of the Sky Collaboration, The Chandra Team at McGill University, DFN, :, D. Fireball Network, ATLAS, H. Time Resolution Universe Survey, RIMAS, RATIR, and S. South Africa/MeerKAT. Multi-messenger Observations of a Binary Neutron Star Merger. *ArXiv e-prints*, October 2017.

[22] Leo P. Singer and Larry R. Price. Rapid Bayesian position reconstruction for

gravitational-wave transients. , 93(2):024013, Jan 2016. doi: 10.1103/PhysRevD.93.024013.

[23] Alexander Ly, Maarten Marsman, Josine Verhagen, Raoul Grasman, and Eric-Jan Wagenmakers. A Tutorial on Fisher Information. *arXiv e-prints*, art. arXiv:1705.01064, May 2017.

[24] Leo P. Singer, Larry R. Price, Ben Farr, Alex L. Urban, Chris Pankow, Salvatore Vitale, John Veitch, Will M. Farr, Chad Hanna, Kipp Cannon, Tom Downes, Philip Graff, Carl-Johan Haster, Ilya Mandel, Trevor Sidery, and Alberto Vecchio. The First Two Years of Electromagnetic Follow-up with Advanced LIGO and Virgo. , 795(2):105, November 2014. doi: 10.1088/0004-637X/795/2/105.

[25] S. W. Hawking and W. Israel. *Three Hundred Years of Gravitation*. 1989.

[26] Rachel Gray, Ignacio Magaña Hernandez, Hong Qi, Ankan Sur, Patrick R. Brady, Hsin-Yu Chen, Will M. Farr, Maya Fishbach, Jonathan R. Gair, Archisman Ghosh, Daniel E. Holz, Simone Mastrogiovanni, Christopher Messenger, Danièle A. Steer, and John Veitch. Cosmological inference using gravitational wave standard sirens: A mock data analysis. , 101(12):122001, June 2020. doi: 10.1103/PhysRevD.101.122001.

[27] B. P. Abbott et al. A gravitational-wave measurement of the hubble constant following the second observing run of advanced LIGO and virgo. *The Astrophysical Journal*, 909 (2):218, mar 2021. doi: 10.3847/1538-4357/abdcb7. URL https://doi.org/10.3847/1538-4357/abdcb7.

[28] B. P. Abbott, LIGO Scientific Collaboration, and Virgo Collaboration. Tests of general relativity with the binary black hole signals from the LIGO-Virgo catalog GWTC-1. , 100 (10):104036, November 2019. doi: 10.1103/PhysRevD.100.104036.

[29] B. P. Abbott et al. Tests of general relativity with gw150914. *Phys. Rev. Lett.*, 116: 221101, May 2016. doi: 10.1103/PhysRevLett.116.221101. URL https://link.aps.org/doi/10.1103/PhysRevLett.116.221101.

[30] R. Abbott et al. Tests of general relativity with binary black holes from the second ligo-virgo gravitational-wave transient catalog. *Phys. Rev. D*, 103:122002, Jun 2021. doi: 10.1103/PhysRevD.103.122002. URL https://link.aps.org/doi/10.1103/PhysRevD.103.122002.

[31] J. Weber. Gravitational radiation. *Phys. Rev. Lett.*, 18:498–501, Mar 1967. doi: 10.1103/PhysRevLett.18.498. URL https://link.aps.org/doi/10.1103/PhysRevLett.18.498.

[32] J. Weber. Gravitational-wave-detector events. *Phys. Rev. Lett.*, 20:1307–1308, Jun 1968. doi: 10.1103/PhysRevLett.20.1307. URL https://link.aps.org/doi/10.1103/PhysRevLett.20.1307.

[33] Odylio Denys Aguiar. The past, present and future of the resonant-mass gravitational wave detectors. 2010. doi: 10.1088/1674-4527/11/1/001.

[34] R. A. Hulse and J. H. Taylor. Discovery of a pulsar in a binary system. , 195:L51–L53, January 1975. doi: 10.1086/181708.

[35] J. M. Weisberg and J. H. Taylor. The Relativistic Binary Pulsar B1913+16: Thirty Years of Observations and Analysis. In Fred A. Rasio and Ingrid H. Stairs, editors, *Binary Radio Pulsars*, volume 328 of *Astronomical Society of the Pacific Conference Series*, page 25, July 2005.

[36] J Aasi et al. Advanced LIGO. *Classical and Quantum Gravity*, 32(7):074001, mar 2015. doi: 10.1088/0264-9381/32/7/074001. URL https://doi.org/10.1088/0264-9381/32/7/074001.

[37] F Acernese et al. Advanced virgo: a second-generation interferometric gravitational wave detector. *Classical and Quantum Gravity*, 32(2):024001, dec 2014. doi: 10.1088/0264-9381/32/2/024001. URL https://doi.org/10.1088/0264-9381/32/2/024001.

[38] C Affeldt, K Danzmann, K L Dooley, H Grote, M Hewitson, S Hild, J Hough, J Leong, H Lück, M Prijatelj, S Rowan, A Rüdiger, R Schilling, R Schnabel, E Schreiber, B Sorazu, K A Strain, H Vahlbruch, B Willke, W Winkler, and H Wittel. Advanced techniques in GEO 600. *Classical and Quantum Gravity*, 31(22):224002, nov 2014. doi: 10.1088/0264-9381/31/22/224002. URL https://doi.org/10.1088/0264-9381/31/22/224002.

[39] T Akutsu et al. KAGRA: 2.5 generation interferometric gravitational wave detector. *Nature Astronomy*, 3(1):35–40, 2019. ISSN 2397-3366. doi: 10.1038/s41550-018-0658-y. URL https://doi.org/10.1038/s41550-018-0658-y.

[40] Pau Amaro-Seoane, Sofiane Aoudia, Stanislav Babak, Pierre Binétruy, Emanuele Berti, Alejandro Bohé, Chiara Caprini, Monica Colpi, Neil J. Cornish, Karsten Danzmann, Jean-François Dufaux, Jonathan Gair, Ian Hinder, Oliver Jennrich, Philippe Jetzer, Antoine Klein, Ryan N. Lang, Alberto Lobo, Tyson Littenberg, Sean T. McWilliams, Gijs Nelemans, Antoine Petiteau, Edward K. Porter, Bernard F. Schutz, Alberto Sesana, Robin Stebbins, Tim Sumner, Michele Vallisneri, Stefano Vitale, Marta Volonteri, Henry Ward, and Barry Wardell. eLISA: Astrophysics and cosmology in the millihertz regime. *GW Notes*, 6:4–110, May 2013.

[41] Jun Luo, Li-Sheng Chen, Hui-Zong Duan, Yun-Gui Gong, Shoucun Hu, Jianghui Ji, Qi Liu, Jianwei Mei, Vadim Milyukov, Mikhail Sazhin, Cheng-Gang Shao, Viktor T. Toth, Hai-Bo Tu, Yamin Wang, Yan Wang, Hsien-Chi Yeh, Ming-Sheng Zhan, Yonghe Zhang, Vladimir Zharov, and Ze-Bing Zhou. TianQin: a space-borne gravitational wave detector. *Classical and Quantum Gravity*, 33(3):035010, February 2016. doi: 10.1088/0264-9381/33/3/035010.

[42] A. A. Michelson and E. W. Morley. On the relative motion of the earth and the luminiferous ether. *American Journal of Science*, s3-34(203):333–345, 1887. ISSN 0002-9599. doi: 10.2475/ajs.s3-34.203.333. URL https://www.ajsonline.org/content/s3-34/203/333.

[43] A. Perot and Charles Fabry. On the Application of Interference Phenomena to the Solution of Various Problems of Spectroscopy and Metrology. , 9:87, February 1899. doi: 10.1086/140557.

[44] Sergey P. Tarabrin. Interaction of plane gravitational waves with a fabry-perot cavity in the local lorentz frame. *Phys. Rev. D*, 75:102002, May 2007. doi: 10.1103/PhysRevD.75.102002. URL https://link.aps.org/doi/10.1103/PhysRevD.75.102002.

[45] Peter R. Saulson. Gravitational wave detection: Principles and practice. *Comptes Rendus Physique*, 14(4):288–305, 2013. ISSN 1631-0705. doi: https://doi.org/10.1016/j.crhy.2013.01.007. URL https://www.sciencedirect.com/science/article/pii/S163107051300008X. Gravitational waves / Ondes gravitationnelles.

[46] B. P. Abbott et al. Calibration of the advanced ligo detectors for the discovery of the binary black-hole merger gw150914. *Phys. Rev. D*, 95:062003, Mar 2017. doi: 10.1103/PhysRevD.95.062003. URL https://link.aps.org/doi/10.1103/PhysRevD.95.062003.

[47] Warren G. Anderson, Patrick R. Brady, Jolien D. E. Creighton, and Éanna É. Flanagan. Excess power statistic for detection of burst sources of gravitational radiation. *Phys. Rev. D*, 63:042003, Jan 2001. doi: 10.1103/PhysRevD.63.042003. URL https://link.aps.org/doi/10.1103/PhysRevD.63.042003.

[48] Michele Maggiore. *Gravitational Waves. Vol. 1: Theory and Experiments*. Oxford Master Series in Physics. Oxford University Press, 2007. ISBN 978-0-19-857074-5, 978-0-19-852074-0.

[49] Keith Riles. Recent searches for continuous gravitational waves. 2017. doi: 10.1142/S021773231730035X.

[50] B S Sathyaprakash and Bernard F Schutz. Physics, Astrophysics and Cosmology with Gravitational Waves. *Living Reviews in Relativity*, 12(1):2, 2009. ISSN 1433-8351. doi: 10.12942/lrr-2009-2. URL https://doi.org/10.12942/lrr-2009-2.

[51] Warren G Anderson, Jolien D E Creighton, and Jolien Creighton. *Gravitational-Wave Physics and Astronomy : An Introduction to Theory, Experiment and Data Analysis*. John Wiley Sons, Incorporated, Hoboken, GERMANY, 2011. ISBN 9783527636051. URL http://ebookcentral.proquest.com/lib/gla/detail.action?docID=822717.

[52] Christopher Wipf Jameson Rollins and Rana Adhikari. pygwinc. https://git.ligo.org/gwinc/pygwinc.git, July 2019.

[53] D. M. Macleod, S. Fairhurst, B. Hughey, A. P. Lundgren, L. Pekowsky, J. Rollins, and J. R. Smith. Reducing the effect of seismic noise in LIGO searches by targeted veto generation. *Classical and Quantum Gravity*, 29(5):055006, March 2012. doi: 10.1088/0264-9381/29/5/055006.

[54] Gary L Pavlis. "An Introduction to Seismology, Earthquakes, and Earth Structure" by Seth Stein and Michael Wysession. *Seismological Research Letters*, 74(6):824–825, nov 2003. ISSN 0895-0695. doi: 10.1785/gssrl.74.6.824. URL https://doi.org/10.1785/gssrl.74.6.824.

[55] F Matichard, B Lantz, R Mittleman, K Mason, J Kissel, B Abbott, S Biscans, J McIver, R Abbott, S Abbott, E Allwine, S Barnum, J Birch, C Celerier, D Clark, D Coyne, D DeBra, R DeRosa, M Evans, S Foley, P Fritschel, J A Giaime, C Gray, G Grabeel, J Hanson, C Hardham, M Hillard, W Hua, C Kucharczyk, M Landry, A Le Roux, V Lhuillier, D Macleod, M Macinnis, R Mitchell, B O'Reilly, D Ottaway, H Paris, A Pele, M Puma, H Radkins, C Ramet, M Robinson, L Ruet, P Sarin, D Shoemaker, A Stein, J Thomas, M Vargas, K Venkateswara, J Warner, and S Wen. Seismic isolation of advanced LIGO: Review of strategy, instrumentation and performance. *Classical and Quantum Gravity*, 32(18):185003, aug 2015. doi: 10.1088/0264-9381/32/18/185003. URL https://doi.org/10.1088/0264-9381/32/18/185003.

[56] Benjamin P Abbott, R Abbott, TD Abbott, MR Abernathy, F Acernese, K Ackley, M Adamo, C Adams, T Adams, P Addesso, et al. Characterization of transient noise

in advanced ligo relevant to gravitational wave signal gw150914. *Classical and Quantum Gravity*, 33(13):134001, 2016.

[57] A Effler, R M S Schofield, V V Frolov, G González, K Kawabe, J R Smith, J Birch, and R McCarthy. Environmental influences on the LIGO gravitational wave detectors during the 6th science run. *Classical and Quantum Gravity*, 32(3):035017, jan 2015. doi: 10.1088/0264-9381/32/3/035017. URL https://doi.org/10.1088/0264-9381/32/3/035017.

[58] Kazuhiro Yamamoto, Shigemi Otsuka, Masaki Ando, Keita Kawabe, and Kimio Tsubono. Study of the thermal noise caused by inhomogeneously distributed loss. *Classical and Quantum Gravity*, 19(7):1689–1696, mar 2002. doi: 10.1088/0264-9381/19/7/362. URL https://doi.org/10.1088/0264-9381/19/7/362.

[59] D R M Crooks, G Cagnoli, M M Fejer, G Harry, J Hough, B T Khuri-Yakub, S Penn, R Route, S Rowan, P H Sneddon, I O Wygant, and G G Yaralioglu. Experimental measurements of mechanical dissipation associated with dielectric coatings formed using SiO 2 , ta 2 o 5 and al 2 o 3. *Classical and Quantum Gravity*, 23(15):4953–4965, jul 2006. doi: 10.1088/0264-9381/23/15/014. URL https://doi.org/10.1088/0264-9381/23/15/014.

[60] Akira E. Villar, Eric D. Black, Riccardo DeSalvo, Kenneth G. Libbrecht, Christophe Michel, Nazario Morgado, Laurent Pinard, Innocenzo M. Pinto, Vincenzo Pierro, Vincenzo Galdi, Maria Principe, and Ilaria Taurasi. Measurement of thermal noise in multilayer coatings with optimized layer thickness. *Phys. Rev. D*, 81:122001, Jun 2010. doi: 10.1103/PhysRevD.81.122001. URL https://link.aps.org/doi/10.1103/PhysRevD.81.122001.

[61] Kentaro Somiya. Detector configuration of KAGRA-the Japanese cryogenic gravitational-wave detector. *Classical and Quantum Gravity*, 29(12):124007, June 2012. doi: 10.1088/0264-9381/29/12/124007.

[62] Stefan Hild. *A Basic Introduction to Quantum Noise and Quantum-Non-Demolition Techniques*, pages 291–314. Springer International Publishing, Cham, 2014. ISBN 978-3-319-03792-9. doi: 10.1007/978-3-319-03792-9_11. URL https://doi.org/10.1007/978-3-319-03792-9_11.

[63] The LIGO Scientific Collaboration. A gravitational wave observatory operating beyond the quantum shot-noise limit. *Nature Physics*, 7(12):962–965, 2011. ISSN 1745-2481. doi: 10.1038/nphys2083. URL https://doi.org/10.1038/nphys2083.

[64] Scott A. Hughes and Kip S. Thorne. Seismic gravity-gradient noise in interferometric gravitational-wave detectors. *Phys. Rev. D*, 58:122002, Nov 1998. doi: 10.1103/PhysRevD.58.122002. URL https://link.aps.org/doi/10.1103/PhysRevD.58.122002.

[65] A. Einstein. Die grundlage der allgemeinen relativitätstheorie [adp 49, 769 (1916)]. *Annalen der Physik*, 14(S1):517–571, 2005. doi: https://doi.org/10.1002/andp.200590044. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.200590044.

[66] Sean M. Carroll. *Spacetime and Geometry: An Introduction to General Relativity*. Cambridge University Press, 2019. doi: 10.1017/9781108770385.

[67] Curt Cutler and Éanna E. Flanagan. Gravitational waves from merging compact binaries: How accurately can one extract the binary's parameters from the inspiral waveform? *Phys. Rev. D*, 49:2658–2697, Mar 1994. doi: 10.1103/PhysRevD.49.2658. URL https://link.aps.org/doi/10.1103/PhysRevD.49.2658.

[68] Éanna É Flanagan and Scott A Hughes. The basics of gravitational wave theory. *New Journal of Physics*, 7:204–204, sep 2005. doi: 10.1088/1367-2630/7/1/204. URL https://doi.org/10.1088/1367-2630/7/1/204.

[69] K. G. Arun, Alessandra Buonanno, Guillaume Faye, and Evan Ochsner. Erratum: Higher-order spin effects in the amplitude and phase of gravitational waveforms emitted by inspiraling compact binaries: Ready-to-use gravitational waveforms [phys. rev. d 79, 104023

(2009)]. *Phys. Rev. D*, 84:049901, Aug 2011. doi: 10.1103/PhysRevD.84.049901. URL https://link.aps.org/doi/10.1103/PhysRevD.84.049901.

[70] Alessandra Buonanno, Bala R. Iyer, Evan Ochsner, Yi Pan, and B. S. Sathyaprakash. Comparison of post-newtonian templates for compact binary inspiral signals in gravitational-wave detectors. *Phys. Rev. D*, 80:084043, Oct 2009. doi: 10.1103/PhysRevD.80.084043. URL https://link.aps.org/doi/10.1103/PhysRevD.80.084043.

[71] Luc Blanchet. Gravitational radiation from post-newtonian sources and inspiralling compact binaries. *Living Reviews in Relativity*, 17(1):2, Feb 2014. ISSN 1433-8351. doi: 10.12942/lrr-2014-2. URL https://doi.org/10.12942/lrr-2014-2.

[72] Chandra Kant Mishra, Aditya Kela, K. G. Arun, and Guillaume Faye. Ready-to-use post-newtonian gravitational waveforms for binary black holes with nonprecessing spins: An update. *Phys. Rev. D*, 93:084054, Apr 2016. doi: 10.1103/PhysRevD.93.084054. URL https://link.aps.org/doi/10.1103/PhysRevD.93.084054.

[73] A. Buonanno and T. Damour. Effective one-body approach to general relativistic two-body dynamics. *Phys. Rev. D*, 59:084006, Mar 1999. doi: 10.1103/PhysRevD.59.084006. URL https://link.aps.org/doi/10.1103/PhysRevD.59.084006.

[74] Sascha Husa, Sebastian Khan, Mark Hannam, Michael Pürrer, Frank Ohme, Xisco Jiménez Forteza, and Alejandro Bohé. Frequency-domain gravitational waves from nonprecessing black-hole binaries. i. new numerical waveforms and anatomy of the signal. *Phys. Rev. D*, 93:044006, Feb 2016. doi: 10.1103/PhysRevD.93.044006. URL https://link.aps.org/doi/10.1103/PhysRevD.93.044006.

[75] Sebastian Khan, Sascha Husa, Mark Hannam, Frank Ohme, Michael Pürrer, Xisco Jiménez Forteza, and Alejandro Bohé. Frequency-domain gravitational waves from nonprecessing black-hole binaries. ii. a phenomenological model for the advanced detector era. *Phys. Rev. D*, 93:044007, Feb 2016. doi: 10.1103/PhysRevD.93.044007. URL https://link.aps.org/doi/10.1103/PhysRevD.93.044007.

[76] Frans Pretorius. Evolution of binary black-hole spacetimes. *Phys. Rev. Lett.*, 95:121101, Sep 2005. doi: 10.1103/PhysRevLett.95.121101. URL https://link.aps.org/doi/10.1103/PhysRevLett.95.121101.

[77] Isobel Romero-Shaw, Paul D. Lasky, Eric Thrane, and Juan Calderón Bustillo. GW190521: Orbital eccentricity and signatures of dynamical formation in a binary black hole merger signal. *The Astrophysical Journal*, 903(1):L5, oct 2020. doi: 10.3847/2041-8213/abbe26. URL https://doi.org/10.3847/2041-8213/abbe26.

[78] Tanja Hinderer, Benjamin D. Lackey, Ryan N. Lang, and Jocelyn S. Read. Tidal deformability of neutron stars with realistic equations of state and their gravitational wave signatures in binary inspiral. *Phys. Rev. D*, 81:123016, Jun 2010. doi: 10.1103/PhysRevD.81.123016. URL https://link.aps.org/doi/10.1103/PhysRevD.81.123016.

[79] Frans Pretorius. *Binary Black Hole Coalescence*, pages 305–369. Springer Netherlands, Dordrecht, 2009. ISBN 978-1-4020-9264-0. doi: 10.1007/978-1-4020-9264-0_9. URL https://doi.org/10.1007/978-1-4020-9264-0_9.

[80] Michela Mapelli. Binary black hole mergers: Formation and populations. *Frontiers in Astronomy and Space Sciences*, 7:38, 2020. ISSN 2296-987X. doi: 10.3389/fspas.2020.00038. URL https://www.frontiersin.org/article/10.3389/fspas.2020.00038.

[81] S W Hawking. Black holes in general relativity. *Communications in Mathematical Physics*, 25(2):152–166, 1972. ISSN 1432-0916. doi: 10.1007/BF01877517. URL https://doi.org/10.1007/BF01877517.

[82] Werner Israel. Event horizons in static vacuum space-times. *Phys. Rev.*, 164:1776–1779, Dec 1967. doi: 10.1103/PhysRev.164.1776. URL https://link.aps.org/doi/10.1103/PhysRev.164.1776.

[83] B. Carter. Axisymmetric black hole has only two degrees of freedom. *Phys. Rev. Lett.*, 26:

331–333, Feb 1971. doi: 10.1103/PhysRevLett.26.331. URL https://link.aps.org/doi/10.1103/PhysRevLett.26.331.

[84] Rodrigo Fernández and Brian D. Metzger. Electromagnetic signatures of neutron star mergers in the advanced ligo era. *Annual Review of Nuclear and Particle Science*, 66(1): 23–45, 2016. doi: 10.1146/annurev-nucl-102115-044819. URL https://doi.org/10.1146/annurev-nucl-102115-044819.

[85] D. Davis et al. LIGO detector characterization in the second and third observing runs. *Classical and Quantum Gravity*, 38(13):135014, July 2021. doi: 10.1088/1361-6382/abfd85.

[86] 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. Gravity spy: integrating advanced ligo detector characterization, machine learning, and citizen science. *Classical and Quantum Gravity*, 34(6):064003, 2017. URL http://stacks.iop.org/0264-9381/34/i=6/a=064003.

[87] L. K. Nuttall. Characterizing transient noise in the LIGO detectors. *Philosophical Transactions of the Royal Society of London Series A*, 376(2120):20170286, May 2018. doi: 10.1098/rsta.2017.0286.

[88] Benjamin J. Owen and B. S. Sathyaprakash. Matched filtering of gravitational waves from inspiraling compact binaries: Computational cost and template placement. *Phys. Rev. D*, 60:022002, Jun 1999. doi: 10.1103/PhysRevD.60.022002. URL https://link.aps.org/doi/10.1103/PhysRevD.60.022002.

[89] P.M. WOODWARD. 6 - the mathematical analysis of radar information. In P.M. WOODWARD, editor, *Probability and Information Theory with Applications to Radar (Second Edition)*, International Series of Monographs on Electronics and Instrumentation, pages 100–114. Pergamon, second edition edition, 1953. doi: https://doi.org/10.

1016/B978-0-08-011006-6.50012-2. URL https://www.sciencedirect.com/science/article/pii/B9780080110066500122.

[90] SUBRAMANIAM AMBATI‡. Studies on a data communication system using digital matched-filter techniques part i. theoretical investigations. *International Journal of Electronics*, 36(1):49–71, 1974. doi: 10.1080/00207217408900375. URL https://doi.org/10.1080/00207217408900375.

[91] Nigel T Bishop and Luciano Rezzolla. Extraction of gravitational waves in numerical relativity. *Living Reviews in Relativity*, 19(1):2, 2016. ISSN 1433-8351. doi: 10.1007/s41114-016-0001-9. URL https://doi.org/10.1007/s41114-016-0001-9.

[92] Clifford M. Will. On the unreasonable effectiveness of the post-newtonian approximation in gravitational physics. *Proceedings of the National Academy of Sciences*, 108(15): 5938–5945, 2011. ISSN 0027-8424. doi: 10.1073/pnas.1103127108. URL https://www.pnas.org/content/108/15/5938.

[93] Vitor Cardoso, Leonardo Gualtieri, Carlos Herdeiro, and Ulrich Sperhake. Exploring New Physics Frontiers Through Numerical Relativity. *Living Reviews in Relativity*, 18 (1):1, 2015. ISSN 1433-8351. doi: 10.1007/lrr-2015-1. URL https://doi.org/10.1007/lrr-2015-1.

[94] Andrea Taracchini, Alessandra Buonanno, Yi Pan, Tanja Hinderer, Michael Boyle, Daniel A. Hemberger, Lawrence E. Kidder, Geoffrey Lovelace, Abdul H. Mroué, Harald P. Pfeiffer, Mark A. Scheel, Béla Szilágyi, Nicholas W. Taylor, and Anil Zenginoglu. Effective-one-body model for black-hole binaries with generic mass ratios and spins. *Phys. Rev. D*, 89:061502, Mar 2014. doi: 10.1103/PhysRevD.89.061502. URL https://link.aps.org/doi/10.1103/PhysRevD.89.061502.

[95] Patricia Schmidt. Gravitational waves from binary black hole mergers: Modeling and observations. *Frontiers in Astronomy and Space Sciences*, 7:28, 2020. ISSN 2296-

987X. doi: 10.3389/fspas.2020.00028. URL https://www.frontiersin.org/article/10.3389/fspas.2020.00028.

[96] Lee S. Finn. Detection, measurement, and gravitational radiation. *Phys. Rev. D*, 46:5236–5249, Dec 1992. doi: 10.1103/PhysRevD.46.5236. URL https://link.aps.org/doi/10.1103/PhysRevD.46.5236.

[97] Bruce Allen, Warren G. Anderson, Patrick R. Brady, Duncan A. Brown, and Jolien D. E. Creighton. Findchirp: An algorithm for detection of gravitational waves from inspiraling compact binaries. *Phys. Rev. D*, 85:122006, Jun 2012. doi: 10.1103/PhysRevD.85.122006. URL https://link.aps.org/doi/10.1103/PhysRevD.85.122006.

[98] J. Cooley and J. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[99] S Babak, R Balasubramanian, D Churches, T Cokelaer, and B S Sathyaprakash. A template bank to search for gravitational waves from inspiralling compact binaries: I. physical models. *Classical and Quantum Gravity*, 23(18):5477, 2006. URL http://stacks.iop.org/0264-9381/23/i=18/a=002.

[100] Benjamin J. Owen. Search templates for gravitational waves from inspiraling binaries: Choice of template spacing. *Phys. Rev. D*, 53:6749–6761, Jun 1996. doi: 10.1103/PhysRevD.53.6749. URL https://link.aps.org/doi/10.1103/PhysRevD.53.6749.

[101] S. V. Dhurandhar and B. S. Sathyaprakash. Choice of filters for the detection of gravitational waves from coalescing binaries. ii. detection in colored noise. *Phys. Rev. D*, 49:1707–1722, Feb 1994. doi: 10.1103/PhysRevD.49.1707. URL https://link.aps.org/doi/10.1103/PhysRevD.49.1707.

[102] S. Babak, R. Balasubramanian, D. Churches, T. Cokelaer, and B. S. Sathyaprakash. A template bank to search for gravitational waves from inspiralling compact binaries: I.

Physical models. *Classical and Quantum Gravity*, 23(18):5477–5504, September 2006. doi: 10.1088/0264-9381/23/18/002.

[103] I. W. Harry, B. Allen, and B. S. Sathyaprakash. Stochastic template placement algorithm for gravitational wave data analysis. *Phys. Rev. D*, 80:104014, Nov 2009. doi: 10.1103/PhysRevD.80.104014. URL https://link.aps.org/doi/10.1103/PhysRevD.80.104014.

[104] Duncan A. Brown, Ian Harry, Andrew Lundgren, and Alexander H. Nitz. Detecting binary neutron star systems with spin in advanced gravitational-wave detectors. *Phys. Rev. D*, 86:084017, Oct 2012. doi: 10.1103/PhysRevD.86.084017. URL https://link.aps.org/doi/10.1103/PhysRevD.86.084017.

[105] P. Ajith, N. Fotopoulos, S. Privitera, A. Neunzert, N. Mazumder, and A. J. Weinstein. Effectual template bank for the detection of gravitational waves from inspiralling compact binaries with generic spins. *Phys. Rev. D*, 89:084041, Apr 2014. doi: 10.1103/PhysRevD.89.084041. URL https://link.aps.org/doi/10.1103/PhysRevD.89.084041.

[106] Stephen Privitera, Satyanarayan R. P. Mohapatra, Parameswaran Ajith, Kipp Cannon, Nickolas Fotopoulos, Melissa A. Frei, Chad Hanna, Alan J. Weinstein, and John T. Whelan. Improving the sensitivity of a search for coalescing binary black holes with nonprecessing spins in gravitational wave data. *Phys. Rev. D*, 89:024003, Jan 2014. doi: 10.1103/PhysRevD.89.024003. URL https://link.aps.org/doi/10.1103/PhysRevD.89.024003.

[107] Collin Capano, Ian Harry, Stephen Privitera, and Alessandra Buonanno. Implementing a search for gravitational waves from non-precessing, spinning binary black holes. *arXiv e-prints*, art. arXiv:1602.03509, February 2016.

[108] Ian W. Harry, Alexander H. Nitz, Duncan A. Brown, Andrew P. Lundgren, Evan Ochsner, and Drew Keppel. Investigating the effect of precession on searches for neutron-star–black-hole binaries with advanced ligo. *Phys. Rev. D*, 89:024010, Jan 2014. doi:

10.1103/PhysRevD.89.024010. URL https://link.aps.org/doi/10.1103/PhysRevD.89.024010.

[109] Bruce Allen. χ^2. *Phys. Rev. D*, 71:062001, Mar 2005. doi: 10.1103/PhysRevD.71.062001. URL https://link.aps.org/doi/10.1103/PhysRevD.71.062001.

[110] Collin Capano, Thomas Dent, Chad Hanna, Martin Hendry, Yi-Ming Hu, Chris Messenger, and John Veitch. Systematic errors in estimation of gravitational-wave candidate significance. *arXiv e-prints*, art. arXiv:1601.00130, January 2016.

[111] Michał Wąs, Marie-Anne Bizouard, Violette Brisson, Fabien Cavalier, Michel Davier, Patrice Hello, Nicolas Leroy, Florent Robinet, and Miltiadis Vavoulidis. On the background estimation by time slides in a network of gravitational wave detectors. *Classical and Quantum Gravity*, 27(1):015005, January 2010. doi: 10.1088/0264-9381/27/1/015005.

[112] R. N. Manchester, G. B. Hobbs, A. Teoh, and M. Hobbs. The australia telescope national facility pulsar catalogue. *The Astronomical Journal*, 129(4):1993–2006, apr 2005. doi: 10.1086/428488. URL https://doi.org/10.1086/428488.

[113] R. Abbott et al. All-sky search in early o3 ligo data for continuous gravitational-wave signals from unknown neutron stars in binary systems. *Phys. Rev. D*, 103:064017, Mar 2021. doi: 10.1103/PhysRevD.103.064017. URL https://link.aps.org/doi/10.1103/PhysRevD.103.064017.

[114] B. P. Abbott et al. First low-frequency einstein@home all-sky search for continuous gravitational waves in advanced ligo data. *Phys. Rev. D*, 96:122004, Dec 2017. doi: 10.1103/PhysRevD.96.122004. URL https://link.aps.org/doi/10.1103/PhysRevD.96.122004.

[115] Max Camenzind. *Compact objects in astrophysics : white dwarfs, neutron stars, and black holes*. 2007. doi: 10.1007/978-3-540-49912-1.

[116] R. N. Manchester, G. B. Hobbs, A. Teoh, and M. Hobbs. The Australia Telescope National Facility Pulsar Catalogue. , 129(4):1993–2006, April 2005. doi: 10.1086/428488.

[117] R. Abbott et al. Gravitational-wave Constraints on the Equatorial Ellipticity of Millisecond Pulsars. , 902(1):L21, October 2020. doi: 10.3847/2041-8213/abb655.

[118] Piotr Jaranowski, Andrzej Królak, and Bernard F. Schutz. Data analysis of gravitational-wave signals from spinning neutron stars: The signal and its detection. , 58(6):063001, September 1998. doi: 10.1103/PhysRevD.58.063001.

[119] Réjean J. Dupuis and Graham Woan. Bayesian estimation of pulsar parameters from gravitational wave data. *Phys. Rev. D*, 72:102002, Nov 2005. doi: 10.1103/PhysRevD.72.102002. URL https://link.aps.org/doi/10.1103/PhysRevD.72.102002.

[120] Piotr Jaranowski, Andrzej Królak, and Bernard F. Schutz. Data analysis of gravitational-wave signals from spinning neutron stars: The signal and its detection. *Phys. Rev. D*, 58:063001, Aug 1998. doi: 10.1103/PhysRevD.58.063001. URL https://link.aps.org/doi/10.1103/PhysRevD.58.063001.

[121] P Astone, S D'Antonio, S Frasca, and C Palomba. A method for detection of known sources of continuous gravitational wave signals in non-stationary data. *Classical and Quantum Gravity*, 27(19):194016, sep 2010. doi: 10.1088/0264-9381/27/19/194016. URL https://doi.org/10.1088/0264-9381/27/19/194016.

[122] C. Messenger and G. Woan. A fast search strategy for gravitational waves from low-mass x-ray binaries. *Classical and Quantum Gravity*, 24(19):S469–S480, October 2007. doi: 10.1088/0264-9381/24/19/S10.

[123] S. Walsh, M. Pitkin, M. Oliver, S. D'Antonio, V. Dergachev, A. Królak, P. Astone, M. Bejger, M. Di Giovanni, O. Dorosh, S. Frasca, P. Leaci, S. Mastrogiovanni, A. Miller, C. Palomba, M. A. Papa, O. J. Piccinni, K. Riles, O. Sauter, and A. M. Sintes. Comparison of methods for the detection of gravitational waves from unknown neutron

stars. *Phys. Rev. D*, 94:124010, Dec 2016. doi: 10.1103/PhysRevD.94.124010. URL https://link.aps.org/doi/10.1103/PhysRevD.94.124010.

[124] J. Aasi et al. Directed search for continuous gravitational waves from the Galactic center. , 88(10):102002, November 2013. doi: 10.1103/PhysRevD.88.102002.

[125] Yuanhao Zhang, Maria Alessandra Papa, Badri Krishnan, and Anna L. Watts. Search for Continuous Gravitational Waves from Scorpius X-1 in LIGO O2 Data. , 906(2):L14, January 2021. doi: 10.3847/2041-8213/abd256.

[126] Grant David Meadors, Evan Goetz, and Keith Riles. Tuning into Scorpius X-1: adapting a continuous gravitational-wave search for a known binary system. *Classical and Quantum Gravity*, 33(10):105017, May 2016. doi: 10.1088/0264-9381/33/10/105017.

[127] LIGO Scientific Collaboration and Virgo Collaboration. Narrow-band search for gravitational waves from known pulsars using the second LIGO observing run. , 99(12):122002, June 2019. doi: 10.1103/PhysRevD.99.122002.

[128] Christian D Ott. The gravitational-wave signature of core-collapse supernovae. *Classical and Quantum Gravity*, 26(6):063001, feb 2009. doi: 10.1088/0264-9381/26/6/063001. URL https://doi.org/10.1088/0264-9381/26/6/063001.

[129] Garvin Yim and D. I. Jones. Transient gravitational waves from pulsar post-glitch recoveries. , 498(3):3138–3152, November 2020. doi: 10.1093/mnras/staa2534.

[130] Robert C. Duncan and Christopher Thompson. Formation of Very Strongly Magnetized Neutron Stars: Implications for Gamma-Ray Bursts. , 392:L9, June 1992. doi: 10.1086/186413.

[131] R. X. Xu. Solid quark stars? *The Astrophysical Journal*, 596(1):L59–L62, sep 2003. doi: 10.1086/379209. URL https://doi.org/10.1086/379209.

[132] S. Klimenko, I. Yakushin, M. Rakhmanov, and G. Mitselmakher. Performance of the WaveBurst algorithm on LIGO data. *Classical and Quantum Gravity*, 21(20):S1685–S1694, October 2004. doi: 10.1088/0264-9381/21/20/011.

[133] Neil J. Cornish and Tyson B. Littenberg. Bayeswave: Bayesian inference for gravitational wave bursts and instrument glitches. *Classical and Quantum Gravity*, 32(13):135012, July 2015. doi: 10.1088/0264-9381/32/13/135012.

[134] S. Klimenko, I. Yakushin, A. Mercer, and G. Mitselmakher. A coherent method for detection of gravitational wave bursts. *Classical and Quantum Gravity*, 25(11):114029, June 2008. doi: 10.1088/0264-9381/25/11/114029.

[135] Patrick J Sutton, Gareth Jones, Shourov Chatterji, Peter Kalmus, Isabel Leonor, Stephen Poprocki, Jameson Rollins, Antony Searle, Leo Stein, Massimo Tinto, and Michal Was. X-pipeline: an analysis package for autonomous gravitational-wave burst searches. *New Journal of Physics*, 12(5):053034, may 2010. doi: 10.1088/1367-2630/12/5/053034. URL https://doi.org/10.1088/1367-2630/12/5/053034.

[136] S. Klimenko, S. Mohanty, M. Rakhmanov, and G. Mitselmakher. Constraint likelihood analysis for a network of gravitational wave detectors. *Phys. Rev. D*, 72:122002, Dec 2005. doi: 10.1103/PhysRevD.72.122002. URL https://link.aps.org/doi/10.1103/PhysRevD.72.122002.

[137] Nelson Christensen. Stochastic gravitational wave backgrounds. *Reports on Progress in Physics*, 82(1):016903, January 2019. doi: 10.1088/1361-6633/aae6b5.

[138] Shivaraj Kandhasamy. *Searches for stochastic gravitational waves and long gravitational wave transients in LIGO S5 data*. PhD thesis, Minnesota U., 2013.

[139] Chiara Caprini. Stochastic background of gravitational waves from cosmological sources. *Journal of Physics: Conference Series*, 610:012004, may 2015. doi: 10.1088/1742-6596/610/1/012004. URL https://doi.org/10.1088/1742-6596/610/1/012004.

[140] Joseph D. Romano and Neil J. Cornish. Detection methods for stochastic gravitational-wave backgrounds: A unified treatment. *Living Reviews in Relativity*, 20(1):1–223, 2017. ISSN 14338351. doi: 10.1007/s41114-017-0004-1. URL https://doi.org/10.1007/s41114-017-0004-1.

[141] Bruce Allen and Joseph D. Romano. Detecting a stochastic background of gravitational radiation: Signal processing strategies and sensitivities. *Phys. Rev. D*, 59:102001, Mar 1999. doi: 10.1103/PhysRevD.59.102001. URL https://link.aps.org/doi/10.1103/PhysRevD.59.102001.

[142] Ingrid H Stairs. Testing General Relativity with Pulsar Timing. *Living Reviews in Relativity*, 6(1):5, 2003. ISSN 1433-8351. doi: 10.12942/lrr-2003-5. URL https://doi.org/10.12942/lrr-2003-5.

[143] Evan F. Keane. Pulsar Science with the SKA. In P. Weltevrede, B. B. P. Perera, L. L. Preston, and S. Sanidas, editors, *Pulsar Astrophysics the Next Fifty Years*, volume 337, pages 158–164, August 2018. doi: 10.1017/S1743921317009188.

[144]

[145] J. Neyman. Outline of a theory of statistical estimation based on the classical theory of probability. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 236(767):333–380, 1937. ISSN 00804614. URL http://www.jstor.org/stable/91337.

[146] Salvatore Vitale, Davide Gerosa, Carl-Johan Haster, Katerina Chatziioannou, and Aaron Zimmerman. Impact of bayesian priors on the characterization of binary black hole coalescences. *Phys. Rev. Lett.*, 119:251103, Dec 2017. doi: 10.1103/PhysRevLett.119.251103. URL https://link.aps.org/doi/10.1103/PhysRevLett.119.251103.

[147] Eric Thrane and Colm Talbot. An introduction to Bayesian inference in gravitational-wave astronomy: Parameter estimation, model selection, and hierarchical models. , 36: e010, March 2019. doi: 10.1017/pasa.2019.2.

[148] Joshua S. Speagle. A Conceptual Introduction to Markov Chain Monte Carlo Methods. *arXiv e-prints*, art. arXiv:1909.12313, September 2019.

[149] Samik Raychaudhuri. Introduction to monte carlo simulation. In *2008 Winter Simulation Conference*, pages 91–100, 2008. doi: 10.1109/WSC.2008.4736059.

[150] J. R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997. doi: 10.1017/CBO9780511810633.

[151] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. doi: 10.1063/1.1699114. URL https://doi.org/10.1063/1.1699114.

[152] William A. Link and Mitchell J. Eaton. On thinning of chains in mcmc. *Methods in Ecology and Evolution*, 3(1):112–115, 2012. doi: https://doi.org/10.1111/j.2041-210X.2011.00131.x. URL https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/j.2041-210X.2011.00131.x.

[153] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman. emcee: The mcmc hammer. *PASP*, 125:306–312, 2013. doi: 10.1086/670067.

[154] Will Vousden, Will M. Farr, and Ilya Mandel. Dynamic temperature selection for parallel-tempering in Markov chain Monte Carlo simulations. 2015. doi: 10.1093/mnras/stv2422.

[155] S. Reyes. Detection and inference in gravitational wave astronomy. 2019.

[156] John Skilling. Nested Sampling. In Rainer Fischer, Roland Preuss, and Udo Von Toussaint, editors, *Bayesian Inference and Maximum Entropy Methods in Science and Engineering: 24th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, volume 735 of *American Institute of Physics Conference Series*, pages 395–405, November 2004. doi: 10.1063/1.1835238.

[157] John Skilling. Nested sampling for general bayesian computation. *Bayesian Anal.*, 1(4): 833–859, 12 2006. doi: 10.1214/06-BA127. URL https://doi.org/10.1214/06-BA127.

[158] Kyle Barbary. nestle: Nested sampling algorithms for evaluating Bayesian evidence, March 2021.

[159] Philip Graff, Farhan Feroz, Michael P. Hobson, and Anthony Lasenby. Bambi: blind accelerated multimodal bayesian inference. *Monthly Notices of the Royal Astronomical Society*, 421(1):169–180, 03 2012. ISSN 0035-8711. doi: 10.1111/j.1365-2966.2011.20288.x. URL https://doi.org/10.1111/j.1365-2966.2011.20288.x.

[160] Joshua S Speagle. dynesty: a dynamic nested sampling package for estimating Bayesian posteriors and evidences. *Monthly Notices of the Royal Astronomical Society*, 493(3): 3132–3158, 02 2020. ISSN 0035-8711. doi: 10.1093/mnras/staa278. URL https://doi.org/10.1093/mnras/staa278.

[161] John Veitch, Walter Del Pozzo, Cody Messick, and Matt Pitkin. Cpnest. Jul 2017. doi: 10.5281/zenodo.835874.

[162] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[163] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-netwo pdf.

[164] Artidoro Pagnoni, Kevin Liu, and Shangyan Li. Conditional Variational Autoencoder for Neural Machine Translation. *arXiv e-prints*, art. arXiv:1812.04405, December 2018.

[165] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. *Efficient BackProp*, pages 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-49430-0. doi: 10.1007/3-540-49430-8_2. URL https://doi.org/10.1007/3-540-49430-8_2.

[166] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. 2014. doi: 10.1109/TPAMI.2014.2345390.

[167] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014. doi: 10.1109/CVPR.2014.241.

[168] A. Toshniwal, K. Mahesh, and R. Jayashree. Overview of anomaly detection techniques in machine learning. In *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 808–815, 2020. doi: 10.1109/I-SMAC49090.2020.9243329.

[169] Chunwei Tian, Lunke Fei, Wenxian Zheng, Yong Xu, Wangmeng Zuo, and Chia-Wen Lin. Deep Learning on Image Denoising: An overview. *arXiv e-prints*, art. arXiv:1912.13171, December 2019.

[170] George Papamakarios. Neural Density Estimation and Likelihood-free Inference. *arXiv e-prints*, art. arXiv:1910.13233, October 2019.

[171] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[172] Katarzyna Janocha and Wojciech Marian Czarnecki. On Loss Functions for Deep Neural Networks in Classification. *arXiv e-prints*, art. arXiv:1702.05659, February 2017.

[173] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *arXiv e-prints*, art. arXiv:1811.03378, November 2018.

[174] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv e-prints*, art. arXiv:1609.04747, September 2016.

[175] Siddharth Krishna Kumar. On weight initialization in deep neural networks. *arXiv e-prints*, art. arXiv:1704.08863, April 2017.

[176] Trond Linjordet and Krisztian Balog. Impact of Training Dataset Size on Neural Answer Selection Models. *arXiv e-prints*, art. arXiv:1901.10496, January 2019.

[177] Phillip Koshute, Jared Zook, and Ian McCulloh. Recommending Training Set Sizes for Classification. *arXiv e-prints*, art. arXiv:2102.09382, February 2021.

[178] Vladimir Vapnik. *The Nature of Statistical Learning Theory*, volume 8, pages 1–15. 01 2000. ISBN 978-1-4419-3160-3. doi: 10.1007/978-1-4757-3264-1_1.

[179] Henry W. Lin, Max Tegmark, and David Rolnick. Why Does Deep and Cheap Learning Work So Well? *Journal of Statistical Physics*, 168(6):1223–1247, September 2017. doi: 10.1007/s10955-017-1836-5.

[180] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv e-prints*, art. arXiv:1611.03530, November 2016.

[181] Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *arXiv e-prints*, art. arXiv:1803.03635, March 2018.

[182] J. Sola and J. Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, 44(3): 1464–1468, 1997. doi: 10.1109/23.589532.

[183] Dalwinder Singh and Birmohan Singh. Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, 97:105524, 2020. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2019.105524. URL https://www.sciencedirect.com/science/article/pii/S1568494619302947.

[184] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[185] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv e-prints*, art. arXiv:1502.03167, February 2015.

[186] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, art. arXiv:1512.03385, December 2015.

[187] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. Tune: A Research Platform for Distributed Model Selection and Training. *arXiv e-prints*, art. arXiv:1807.05118, July 2018.

[188] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012. URL http://jmlr.org/papers/v13/bergstra12a.html.

[189] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

[190] Padraig Cunningham and Sarah Jane Delany. k-Nearest Neighbour Classifiers: 2nd Edition (with Python examples). *arXiv e-prints*, art. arXiv:2004.04523, April 2020.

[191] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object Detection with Deep Learning: A Review. *arXiv e-prints*, art. arXiv:1807.05511, July 2018.

[192] Zhaohui Zhang, Xinxin Zhou, Xiaobo Zhang, Lizhi Wang, and Pengwei Wang. A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection. *Security and Communication Networks*, 2018:5680264, 2018. ISSN 1939-0114. doi: 10.1155/2018/5680264. URL https://doi.org/10.1155/2018/5680264.

[193] Walaa N. Ismail, Mohammad Mehedi Hassan, Hessah A. Alsalamah, and Giancarlo Fortino. Cnn-based health model for regular health factors analysis in internet-of-medical things environment. *IEEE Access*, 8:52541–52549, 2020. doi: 10.1109/ACCESS.2020.2980938.

[194] Chen Kong and Simon Lucey. Take it in your stride: Do we need striding in CNNs? *arXiv e-prints*, art. arXiv:1712.02502, December 2017.

[195] W. Wang, Y. Huang, Y. Wang, and L. Wang. Generalized autoencoder: A neural network framework for dimensionality reduction. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 496–503, 2014. doi: 10.1109/CVPRW.2014.79.

[196] Mizuho Nishio, Chihiro Nagashima, Saori Hirabayashi, Akinori Ohnishi, Kaori Sasaki, Tomoyuki Sagawa, Masayuki Hamada, and Tatsuo Yamashita. Convolutional auto-encoder for image denoising of ultra-low-dose ct. *Heliyon*, 3(8):e00393, 2017. ISSN 2405-8440. doi: https://doi.org/10.1016/j.heliyon.2017.e00393. URL http://www.sciencedirect.com/science/article/pii/S2405844016321600.

[197] Q. Meng, D. Catchpoole, D. Skillicom, and P. J. Kennedy. Relational autoencoder for feature extraction. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 364–371, 2017. doi: 10.1109/IJCNN.2017.7965877.

[198] Stephen Odaibo. Tutorial: Deriving the Standard Variational Autoencoder (VAE) Loss Function. *arXiv e-prints*, art. arXiv:1907.08956, July 2019.

[199] I. Higgins, L. Matthey, A. Pal, Christopher P. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[200] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL http://arxiv.org/abs/1708.07747.

[201] J. McGinn, C. Messenger, M. J. Williams, and I. S. Heng. Generalised gravitational wave burst generation with generative adversarial networks. *Classical and Quantum Gravity*, 38(15):155005, August 2021. doi: 10.1088/1361-6382/ac09cc.

[202] Daniel George and E. A. Huerta. Deep neural networks to enable real-time multimessenger astrophysics. *Phys. Rev. D*, 97:044039, Feb 2018. doi: 10.1103/PhysRevD.97.044039. URL https://link.aps.org/doi/10.1103/PhysRevD.97.044039.

[203] Antony C. Searle. Monte-Carlo and Bayesian techniques in gravitational wave burst data analysis. *arXiv e-prints*, art. arXiv:0804.1161, April 2008.

[204] Jingkai Yan, Mariam Avagyan, Robert E. Colgan, Doğa Veske, Imre Bartos, John Wright, Zsuzsa Márka, and Szabolcs Márka. Generalized Approach to Matched Filtering using Neural Networks. *arXiv e-prints*, art. arXiv:2104.03961, April 2021.

[205] Shin Fujieda, Kohei Takayama, and Toshiya Hachisuka. Wavelet Convolutional Neural Networks. *arXiv e-prints*, art. arXiv:1805.08620, May 2018.

[206] Plamen G. Krastev. Real-time detection of gravitational waves from binary neutron stars using artificial neural networks. *Physics Letters B*, 803:135330, 2020. ISSN 0370-2693. doi: https://doi.org/10.1016/j.physletb.2020.135330. URL https://www.sciencedirect.com/science/article/pii/S0370269320301349.

[207] B.S. Everitt and A. Skrondal. *The Cambridge Dictionary of Statistics*. Cambridge University Press, 2010. ISBN 9780521766999. URL https://books.google.co.uk/books?id=C98wSQAACAAJ.

[208] Marlin B. Schäfer, Frank Ohme, and Alexander H. Nitz. Detection of gravitational-wave signals from binary neutron star mergers using machine learning. *Phys. Rev. D*, 102:063015, Sep 2020. doi: 10.1103/PhysRevD.102.063015. URL https://link.aps.org/doi/10.1103/PhysRevD.102.063015.

[209] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *arXiv e-prints*, art. arXiv:1409.4842, September 2014.

[210] Colin Lea, Michael Flynn, Rene Vidal, Austin Reiter, and Gregory Hager. Temporal convolutional networks for action segmentation and detection. pages 1003–1012, 07 2017. doi: 10.1109/CVPR.2017.113.

[211] Roberto Corizzo, Michelangelo Ceci, Eftim Zdravevski, and Nathalie Japkowicz. Scalable auto-encoders for gravitational waves detection from time series data. *Expert Systems with Applications*, 151:113378, 2020. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.113378. URL https://www.sciencedirect.com/science/article/pii/S0957417420301986.

[212] T. Gebhard, N. Kilbertus, G. Parascandolo, I. Harry, and B. Schölkopf. Convwave: Searching for gravitational waves with fully convolutional neural nets. In *Workshop on Deep Learning for Physical Sciences (DLPS) at the 31st Conference on Neural Information Processing Systems (NIPS)*, 2017. URL https://dl4physicalsciences.github.io/files/nips_dlps_2017_13.pdf.

[213] Timothy D. Gebhard, Niki Kilbertus, Ian Harry, and Bernhard Schölkopf. Convolutional neural networks: A magic bullet for gravitational-wave detection? *Phys. Rev. D*, 100: 063015, Sep 2019. doi: 10.1103/PhysRevD.100.063015. URL https://link.aps.org/doi/10.1103/PhysRevD.100.063015.

[214] Dwyer S. Deighan, Scott E. Field, Collin D. Capano, and Gaurav Khanna. Genetic-algorithm-optimized neural networks for gravitational wave classification. *arXiv e-prints*, art. arXiv:2010.04340, October 2020.

[215] Siyu Fan, Yisen Wang, Yuan Luo, Alexander Schmitt, and Shenghua Yu. Improving gravitational wave detection with 2d convolutional neural networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 7103–7110, 2021. doi: 10.1109/ICPR48806.2021.9412180.

[216] Grégory Baltus, Justin Janquart, Melissa Lopez, Amit Reza, Sarah Caudill, and Jean-René Cudell. Detecting the early inspiral of a gravitational-wave signal with convolutional neural networks. *arXiv e-prints*, art. arXiv:2105.13664, May 2021.

[217] P. Astone, P. Cerdá-Durán, I. Di Palma, M. Drago, F. Muciaccia, C. Palomba, and F. Ricci. New method to observe gravitational waves emitted by core collapse supernovae. , 98 (12):122002, December 2018. doi: 10.1103/PhysRevD.98.122002.

[218] Man Leong Chan, Ik Siong Heng, and Chris Messenger. Detection and classification of supernova gravitational wave signals: A deep learning approach. *Phys. Rev. D*, 102: 043022, Aug 2020. doi: 10.1103/PhysRevD.102.043022. URL https://link.aps.org/doi/10.1103/PhysRevD.102.043022.

[219] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv e-prints*, art. arXiv:1406.2661, June 2014.

[220] Vasileios Skliris, Michael R. K. Norman, and Patrick J. Sutton. Real-Time Detection of Unmodeled Gravitational-Wave Transients UsingConvolutional Neural Networks. *arXiv e-prints*, art. arXiv:2009.14611, September 2020.

[221] Christoph Dreissigacker, Rahul Sharma, Chris Messenger, Ruining Zhao, and Reinhard Prix. Deep-learning continuous gravitational waves. *Phys. Rev. D*, 100:044009, Aug 2019. doi: 10.1103/PhysRevD.100.044009. URL https://link.aps.org/doi/10.1103/PhysRevD.100.044009.

[222] K. Wette, S. Walsh, R. Prix, and M. A. Papa. Implementing a semicoherent search for continuous gravitational waves using optimally constructed template banks. *Phys. Rev. D*, 97:123016, Jun 2018. doi: 10.1103/PhysRevD.97.123016. URL https://link.aps.org/doi/10.1103/PhysRevD.97.123016.

[223] Reinhard Prix. Template-based searches for gravitational waves: efficient lattice covering of flat parameter spaces. *Classical and Quantum Gravity*, 24(19):S481–S490, October 2007. doi: 10.1088/0264-9381/24/19/S11.

[224] Karl Wette. Lattice template placement for coherent all-sky searches for gravitational-wave pulsars. *Phys. Rev. D*, 90:122010, Dec 2014. doi: 10.1103/PhysRevD.90.122010. URL https://link.aps.org/doi/10.1103/PhysRevD.90.122010.

[225] Christoph Dreissigacker and Reinhard Prix. Deep-learning continuous gravitational waves: Multiple detectors and realistic noise. , 102(2):022005, July 2020. doi: 10.1103/PhysRevD.102.022005.

[226] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv e-prints*, art. arXiv:1602.07261, February 2016.

[227] Filip Morawski, Michał Bejger, and Paweł Cieciel\{a}g. Convolutional neural network classifier for the output of the time-domain F-statistic all-sky search for continuous gravitational waves. *arXiv e-prints*, art. arXiv:1907.06917, July 2019.

[228] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *arXiv e-prints*, art. arXiv:1703.06870, March 2017.

[229] B. Beheshtipour and M. A. Papa. Deep learning for clustering of continuous gravitational wave candidates. *Phys. Rev. D*, 101:064009, Mar 2020. doi: 10.1103/PhysRevD.101.064009. URL https://link.aps.org/doi/10.1103/PhysRevD.101.064009.

[230] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, 1990. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(90)90005-6. URL https://www.sciencedirect.com/science/article/pii/0893608090900056.

[231] F. Feroz, M. P. Hobson, and M. Bridges. MULTINEST: an efficient and robust Bayesian inference tool for cosmology and particle physics. , 398(4):1601–1614, October 2009. doi: 10.1111/j.1365-2966.2009.14548.x.

[232] Yun Shang and Y. Qiu. A note on the extended rosenbrock function. *Evolutionary Computation*, 14:119–126, 2006.

[233] Elena Cuoco, Jade Powell, Marco Cavaglià, Kendall Ackley, Michał Bejger, Chayan Chatterjee, Michael Coughlin, Scott Coughlin, Paul Easter, Reed Essick, Hunter Gabbard, Timothy Gebhard, Shaon Ghosh, Leïla Haegel, Alberto Iess, David Keitel, Zsuzsa Márka, Szabolcs Márka, Filip Morawski, Tri Nguyen, Rich Ormiston, Michael Pürrer, Massimiliano Razzano, Kai Staats, Gabriele Vajente, and Daniel Williams. Enhancing gravitational-wave science with machine learning. *Machine Learning: Science and Technology*, 2(1):011002, dec 2020. doi: 10.1088/2632-2153/abb93a. URL https://doi.org/10.1088/2632-2153/abb93a.

[234] Alvin J. K. Chua and Michele Vallisneri. Learning Bayes' theorem with a neural network for gravitational-wave inference. *arXiv e-prints*, art. arXiv:1909.05966, Sep 2019.

[235] Alvin J. K. Chua, Chad R. Galley, and Michele Vallisneri. Reduced-order modeling with artificial neurons for gravitational-wave inference. *Phys. Rev. Lett.*, 122:211101, May 2019. doi: 10.1103/PhysRevLett.122.211101. URL https://link.aps.org/doi/10.1103/PhysRevLett.122.211101.

[236] Stephen R. Green, Christine Simpson, and Jonathan Gair. Gravitational-wave parameter estimation with autoregressive neural network flows. *Phys. Rev. D*, 102:104057, Nov 2020. doi: 10.1103/PhysRevD.102.104057. URL https://link.aps.org/doi/10.1103/PhysRevD.102.104057.

[237] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *arXiv e-prints*, art. arXiv:1505.05770, May 2015.

[238] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. *arXiv e-prints*, art. arXiv:1705.07057, May 2017.

[239] Stephen R. Green and Jonathan Gair. Complete parameter inference for GW150914 using deep learning. *arXiv e-prints*, art. arXiv:2008.03312, August 2020.

[240] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural Spline Flows. *arXiv e-prints*, art. arXiv:1906.04032, June 2019.

[241] Michael J. Williams, John Veitch, and Chris Messenger. Nested sampling with normalizing flows for gravitational-wave inference. *Phys. Rev. D*, 103:103006, May 2021. doi: 10.1103/PhysRevD.103.103006. URL https://link.aps.org/doi/10.1103/PhysRevD.103.103006.

[242] Arnaud Delaunoy, Antoine Wehenkel, Tanja Hinderer, Samaya Nissanke, Christoph Weniger, Andrew R. Williamson, and Gilles Louppe. Lightning-Fast Gravitational Wave Parameter Inference through Neural Amortization. *arXiv e-prints*, art. arXiv:2010.12931, October 2020.

[243] Han-Shiang Kuo and Feng-Li Lin. Conditional Noise Deep Learning for Parameter Estimation of Gravitational Wave Events. *arXiv e-prints*, art. arXiv:2107.10730, July 2021.

[244] João D. Álvares, José A. Font, Felipe F. Freitas, Osvaldo G. Freitas, António P. Morais, Solange Nunes, Antonio Onofre, and Alejandro Torres-Forné. Exploring gravitational-wave detection and parameter inference using deep learning methods. *Classical and Quantum Gravity*, 38(15):155010, August 2021. doi: 10.1088/1361-6382/ac0455.

[245] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of Tricks for Image Classification with Convolutional Neural Networks. *arXiv e-prints*, art. arXiv:1812.01187, December 2018.

[246] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *arXiv e-prints*, art. arXiv:1506.02142, June 2015.

[247] B. P. Abbott et al. Binary black hole population properties inferred from the first and second observing runs of advanced LIGO and advanced virgo. *The Astrophysical Journal*, 882(2):L24, sep 2019. doi: 10.3847/2041-8213/ab3800. URL https://doi.org/10.3847/2041-8213/ab3800.

[248] Michael Zevin, Chris Pankow, Carl L. Rodriguez, Laura Sampson, Eve Chase, Vassiliki Kalogera, and Frederic A. Rasio. Constraining formation models of binary black holes with gravitational-wave observations. *The Astrophysical Journal*, 846(1):82,

sep 2017. doi: 10.3847/1538-4357/aa8408. URL https://doi.org/10.3847/1538-4357/aa8408.

[249] Kaze W. K. Wong, Gabriella Contardo, and Shirley Ho. Gravitational-wave population inference with deep flow-based generative network. *Phys. Rev. D*, 101:123005, Jun 2020. doi: 10.1103/PhysRevD.101.123005. URL https://link.aps.org/doi/10.1103/PhysRevD.101.123005.

[250] Zooniverse. URL https://www.zooniverse.org/.

[251] S Chatterji, L Blackburn, G Martin, and E Katsavounidis. Multiresolution techniques for the detection of gravitational-wave bursts. *Classical and Quantum Gravity*, 21(20): S1809–S1818, sep 2004. doi: 10.1088/0264-9381/21/20/024. URL https://doi.org/10.1088/0264-9381/21/20/024.

[252] Florent Robinet, Nicolas Arnaud, Nicolas Leroy, Andrew Lundgren, Duncan Macleod, and Jessica McIver. Omicron: A tool to characterize transient noise in gravitational-wave detectors. *SoftwareX*, 12:100620, 2020. ISSN 2352-7110. doi: https://doi.org/10.1016/j.softx.2020.100620. URL https://www.sciencedirect.com/science/article/pii/S2352711020303332.

[253] Bogumił Kamiński, Michał Jakubczyk, and Przemysław Szufel. A framework for sensitivity analysis of decision trees. *Central European Journal of Operations Research*, 26(1):135–159, 2018. ISSN 1613-9178. doi: 10.1007/s10100-017-0479-6. URL https://doi.org/10.1007/s10100-017-0479-6.

[254] Riccardo Poli, William Langdon, and Nicholas Mcphee. *A Field Guide to Genetic Programming*. 01 2008. ISBN 978-1-4092-0073-4.

[255] Finding the origin of noise transients in ligo data with machine learning. *Communications in Computational Physics*, 25(4):963–987, 2018. ISSN 1991-7120. doi: https://doi.org/10.4208/cicp.OA-2018-0092. URL http://global-sci.org/intro/article_detail/cicp/12886.html.

[256] N. Mukund, S. Abraham, S. Kandhasamy, S. Mitra, and N. S. Philip. Transient classification in ligo data using difference boosting neural network. *Phys. Rev. D*, 95:104059, May 2017. doi: 10.1103/PhysRevD.95.104059. URL https://link.aps.org/doi/10.1103/PhysRevD.95.104059.

[257] Rich Ormiston, Tri Nguyen, Michael Coughlin, Rana X. Adhikari, and Erik Katsavounidis. Noise reduction in gravitational-wave data via deep learning. *Phys. Rev. Research*, 2:033066, Jul 2020. doi: 10.1103/PhysRevResearch.2.033066. URL https://link.aps.org/doi/10.1103/PhysRevResearch.2.033066.

[258] G. Vajente, Y. Huang, M. Isi, J. C. Driggers, J. S. Kissel, M. J. Szczepańczyk, and S. Vitale. Machine-learning nonstationary noise out of gravitational-wave detectors. *Phys. Rev. D*, 101:042003, Feb 2020. doi: 10.1103/PhysRevD.101.042003. URL https://link.aps.org/doi/10.1103/PhysRevD.101.042003.

[259] Rahul Biswas, Lindy Blackburn, Junwei Cao, Reed Essick, Kari Alison Hodge, Erotokritos Katsavounidis, Kyungmin Kim, Young-Min Kim, Eric-Olivier Le Bigot, Chang-Hwan Lee, John J. Oh, Sang Hoon Oh, Edwin J. Son, Ye Tao, Ruslan Vaulin, and Xiaoge Wang. Application of machine learning algorithms to the study of noise artifacts in gravitational-wave data. *Phys. Rev. D*, 88:062003, Sep 2013. doi: 10.1103/PhysRevD.88.062003. URL https://link.aps.org/doi/10.1103/PhysRevD.88.062003.

[260] B. P. Abbott et al. Gw170104: Observation of a 50-solar-mass binary black hole coalescence at redshift 0.2. *Phys. Rev. Lett.*, 118:221101, Jun 2017. doi: 10.1103/PhysRevLett.118.221101. URL https://link.aps.org/doi/10.1103/PhysRevLett.118.221101.

[261] A. Goldstein, P. Veres, E. Burns, M. S. Briggs, R. Hamburg, D. Kocevski, C. A. Wilson-Hodge, R. D. Preece, S. Poolakkil, O. J. Roberts, C. M. Hui, V. Connaughton, J. Racusin, A. von Kienlin, T. Dal Canton, N. Christensen, T. B. Littenberg, K. Siellez, L. Blackburn, J. Broida, E. Bissaldi, W. H. Cleveland, M. H. Gibby, M. M. Giles, R. M. Kippen, S. McBreen, J. McEnery, C. A. Meegan, W. S. Paciesas, and M. Stanbro. An Ordi-

nary Short Gamma-Ray Burst with Extraordinary Implications: Fermi-GBM Detection of GRB 170817A. *ArXiv e-prints*, October 2017.

[262] V. Savchenko, C. Ferrigno, E. Kuulkers, A. Bazzano, E. Bozzo, S. Brandt, J. Chenevez, T. J.-L. Courvoisier, R. Diehl, A. Domingo, L. Hanlon, E. Jourdain, A. von Kienlin, P. Laurent, F. Lebrun, A. Lutovinov, A. Martin-Carrillo, S. Mereghetti, L. Natalucci, J. Rodi, J.-P. Roques, R. Sunyaev, and P. Ubertini. INTEGRAL Detection of the First Prompt Gamma-Ray Signal Coincident with the Gravitational Wave Event GW170817. *ArXiv e-prints*, October 2017.

[263] Kipp Cannon, Romain Cariou, Adrian Chapman, Mireia Crispin-Ortuzar, Nickolas Fotopoulos, Melissa Frei, Chad Hanna, Erin Kara, Drew Keppel, Laura Liao, Stephen Privitera, Antony Searle, Leo Singer, and Alan Weinstein. Toward early-warning detection of gravitational waves from compact binary coalescence. *The Astrophysical Journal*, 748 (2):136, 2012. URL http://stacks.iop.org/0004-637X/748/i=2/a=136.

[264] Tito Dal Canton, Alexander H. Nitz, Andrew P. Lundgren, Alex B. Nielsen, Duncan A. Brown, Thomas Dent, Ian W. Harry, Badri Krishnan, Andrew J. Miller, Karl Wette, Karsten Wiesner, and Joshua L. Willis. Implementing a search for aligned-spin neutron star-black hole systems with advanced ground based gravitational wave detectors. *Phys. Rev. D*, 90:082004, Oct 2014. doi: 10.1103/PhysRevD.90.082004. URL https://link.aps.org/doi/10.1103/PhysRevD.90.082004.

[265] Tito Dal Canton and Ian W. Harry. Designing a template bank to observe compact binary coalescences in Advanced LIGO's second observing run. *arXiv e-prints*, art. arXiv:1705.01845, May 2017.

[266] B. S. Sathyaprakash and S. V. Dhurandhar. Choice of filters for the detection of gravitational waves from coalescing binaries. *Phys. Rev. D*, 44:3819–3834, Dec 1991. doi: 10.1103/PhysRevD.44.3819. URL https://link.aps.org/doi/10.1103/PhysRevD.44.3819.

[267] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv e-prints*, art. arXiv:1409.1556, September 2014.

[268] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. *arXiv e-prints*, art. arXiv:1412.7062, December 2014.

[269] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. *arXiv e-prints*, art. arXiv:1311.2901, November 2013.

[270] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful Image Colorization. *arXiv e-prints*, art. arXiv:1603.08511, March 2016.

[271] Andrej Karpathy and Li Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. *arXiv e-prints*, art. arXiv:1412.2306, December 2014.

[272] Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in Medicine*, 23(1):89 – 109, 2001. ISSN 0933-3657. doi: https://doi.org/10.1016/S0933-3657(01)00077-X. URL http://www.sciencedirect.com/science/article/pii/S093336570100077X.

[273] Mehdi Pirooznia, Jack Y. Yang, Mary Qu Yang, and Youping Deng. A comparative study of different machine learning methods on microarray gene expression data. *BMC Genomics*, 9(1):S13, Mar 2008. ISSN 1471-2164. doi: 10.1186/1471-2164-9-S1-S13. URL https://doi.org/10.1186/1471-2164-9-S1-S13.

[274] Daniel George, Hongyu Shen, and E. A. Huerta. Classification and unsupervised clustering of ligo data with deep transfer learning. *Phys. Rev. D*, 97:101501, May 2018. doi: 10.1103/PhysRevD.97.101501. URL https://link.aps.org/doi/10.1103/PhysRevD.97.101501.

[275] Daniel George and E.A. Huerta. Deep learning for real-time gravitational wave detection and parameter estimation: Results with advanced ligo data. *Physics Letters B*, 778:64 – 70, 2018. ISSN 0370-2693. doi: https://doi.org/10.1016/j.physletb.2017.

12.053. URL http://www.sciencedirect.com/science/article/pii/S0370269317310390.

[276] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, and et al. Prospects for Observing and Localizing Gravitational-Wave Transients with Advanced LIGO and Advanced Virgo. *Living Reviews in Relativity*, 19:1, February 2016. doi: 10.1007/lrr-2016-1.

[277] B. P. Abbott et al. Binary black hole mergers in the first advanced LIGO observing run. *Phys. Rev. X*, 6:041015, Oct 2016. doi: 10.1103/PhysRevX.6.041015. URL https://link.aps.org/doi/10.1103/PhysRevX.6.041015.

[278] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., USA, 1995. ISBN 0198538642.

[279] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[280] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010. ISSN 1041-4347. doi: 10.1109/TKDE.2009.191.

[281] Aleksandar Botev, Guy Lever, and David Barber. Nesterov's Accelerated Gradient and Momentum as approximations to Regularised Update Descent. *arXiv e-prints*, art. arXiv:1607.01981, July 2016.

[282] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, art. arXiv:1412.6980, December 2014.

[283] Leslie N. Smith. Cyclical Learning Rates for Training Neural Networks. *arXiv e-prints*, art. arXiv:1506.01186, June 2015.

[284] 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. Searching for gravitational waves from binary coalescence. , 87(2):024033, January 2013. doi: 10.1103/PhysRevD.87.024033.

[285] Alex Nitz et al. Pycbc software, September 2017. URL https://ligo-cbc.github.io/.

[286] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80023-1. URL https://www.sciencedirect.com/science/article/pii/S0893608005800231.

[287] Kai Ming Ting and Ian H Witten. Stacked generalization: when does it work? Technical report, Hamilton, 1997. URL https://hdl.handle.net/10289/1066.

[288] Xiang-Ru Li, Wo-Liang Yu, Xi-Long Fan, and G Jogesh Babu. Some optimizations on detecting gravitational wave using convolutional neural network. *Frontiers of Physics*, 15 (5):54501, 2020. ISSN 2095-0470. doi: 10.1007/s11467-020-0966-4. URL https://doi.org/10.1007/s11467-020-0966-4.

[289] Juan Pablo Marulanda, Camilo Santa, and Antonio Enea Romano. Deep learning merger masses estimation from gravitational waves signals in the frequency domain. *Physics Letters B*, 810:135790, November 2020. doi: 10.1016/j.physletb.2020.135790.

[290] Rahul Nigam, Amit Mishra, and Pranath Reddy. Transient Classification in low SNR Gravitational Wave data using Deep Learning. *arXiv e-prints*, art. arXiv:2009.12168, September 2020.

[291] Alexander H. Nitz, Thomas Dent, Tito Dal Canton, Stephen Fairhurst, and Duncan A.

Brown. Detecting binary compact-object mergers with gravitational waves: Understanding and improving the sensitivity of the pycbc search. *The Astrophysical Journal*, 849(2): 118, 2017. URL http://stacks.iop.org/0004-637X/849/i=2/a=118.

[292] B P Abbott et al. Effects of data quality vetoes on a search for compact binary coalescences in advanced LIGO's first observing run. *Classical and Quantum Gravity*, 35(6): 065010, feb 2018. doi: 10.1088/1361-6382/aaaafa. URL https://doi.org/10.1088/1361-6382/aaaafa.

[293] B P Abbott et al. Characterization of transient noise in advanced ligo relevant to gravitational wave signal gw150914. *Classical and Quantum Gravity*, 33(13):134001, 2016. URL http://stacks.iop.org/0264-9381/33/i=13/a=134001.

[294] Maximilian Dax, Stephen R. Green, Jonathan Gair, Jakob H. Macke, Alessandra Buonanno, and Bernhard Schölkopf. Real-time gravitational-wave science with neural posterior estimation. *arXiv e-prints*, art. arXiv:2106.12594, June 2021.

[295] Yu-Chiung Lin and Jiun-Huei Proty Wu. Detection of gravitational waves using bayesian neural networks. *Phys. Rev. D*, 103:063034, Mar 2021. doi: 10.1103/PhysRevD.103.063034. URL https://link.aps.org/doi/10.1103/PhysRevD.103.063034.

[296] Grégory Baltus, Justin Janquart, Melissa Lopez, Amit Reza, Sarah Caudill, and Jean-René Cudell. Convolutional neural networks for the detection of the early inspiral of a gravitational-wave signal. *Phys. Rev. D*, 103:102003, May 2021. doi: 10.1103/PhysRevD.103.102003. URL https://link.aps.org/doi/10.1103/PhysRevD.103.102003.

[297] The KAGRA Collaboration, the LIGO Scientific Collaboration, and the Virgo Collaboration. Prospects for observing and localizing gravitational-wave transients with advanced LIGO, advanced Virgo and KAGRA. *Living Reviews in Relativty*, 2013. doi: 10.1007/s41114-018-0012-9.

[298] Antony C. Searle, Patrick J. Sutton, and Massimo Tinto. Bayesian detection of unmodeled bursts of gravitational waves. *Classical and Quantum Gravity*, 26(15):155017, Aug 2009. doi: 10.1088/0264-9381/26/15/155017.

[299] Gracedb — gravitational-wave candidate event database (ligo/virgo o3 public alerts). https://gracedb.ligo.org/superevents/public/O3/. Accessed: 2019-09-16.

[300] Francesco Tonolini, Jack Radford, Alex Turpin, Daniele Faccio, and Roderick Murray-Smith. Variational inference for computational imaging inverse problems. *Journal of Machine Learning Research*, 21(179):1–46, 2020. URL http://jmlr.org/papers/v21/20-151.html.

[301] Michael Coughlin, Paul Earle, Jan Harms, Sebastien Biscans, Christopher Buchanan, Eric Coughlin, Fred Donovan, Jeremy Fee, Hunter Gabbard, Michelle Guy, Nikhil Mukund, and Matthew Perry. Limiting the effects of earthquakes on gravitational-wave interferometers. *Classical and Quantum Gravity*, 34(4):044004, feb 2017. doi: 10.1088/1361-6382/aa5a60. URL https://doi.org/10.1088%2F1361-6382%2Faa5a60.

[302] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1912789117.

[303] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3483–3491. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5775-learning-structured-output-representation-using-deep-conditi pdf.

[304] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2Image: Condi-

tional Image Generation from Visual Attributes. *arXiv e-prints*, art. arXiv:1512.00570, December 2015.

[305] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space. *arXiv e-prints*, art. arXiv:1612.00005, November 2016.

[306] Jaehyeon Kim, Jungil Kong, and Juhee Son. Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech. *arXiv e-prints*, art. arXiv:2106.06103, June 2021.

[307] Alfredo Nazabal, Pablo M. Olmos, Zoubin Ghahramani, and Isabel Valera. Handling Incomplete Heterogeneous Data using VAEs. *arXiv e-prints*, art. arXiv:1807.03653, July 2018.

[308] P Gallinari, Yann LeCun, S Thiria, and F Fogelman Soulie. Mémoires associatives distribuées: une comparaison (distributed associative memories: a comparison). In *Proceedings of COGNITIVA 87, Paris, La Villette, May 1987*. Cesta-Afcet, 1987.

[309] Advanced LIGO sensitivity design curve. https://dcc.ligo.org/LIGO-T1800044/public. Accessed: 2019-06-01.

[310] Sebastian Khan, Katerina Chatziioannou, Mark Hannam, and Frank Ohme. Phenomenological model for the gravitational-wave signal from precessing binary black holes with two-spin effects. *Phys. Rev. D*, 100:024059, Jul 2019. doi: 10.1103/PhysRevD.100.024059. URL https://link.aps.org/doi/10.1103/PhysRevD.100.024059.

[311] Q. Wang, S. R. Kulkarni, and S. Verdu. Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Transactions on Information Theory*, 55 (5):2392–2405, 2009.

[312] G. Ashton and C. Talbot. Bilby-MCMC: An MCMC sampler for gravitational-wave inference. , August 2021. doi: 10.1093/mnras/stab2236.

[313] Rory Smith, Scott E. Field, Kent Blackburn, Carl-Johan Haster, Michael Pürrer, Vivien Raymond, and Patricia Schmidt. Fast and accurate inference on gravitational waves from precessing compact binaries. *Physical Review D*, 94(4):044031, August 2016. doi: 10.1103/PhysRevD.94.044031.

[314] D. Wysocki, R. O'Shaughnessy, Jacob Lange, and Yao-Lung L. Fang. Accelerating parameter inference with graphics processing units. *Physical Review D*, 99(8):084026, April 2019. doi: 10.1103/PhysRevD.99.084026.

[315] Colm Talbot, Rory Smith, Eric Thrane, and Gregory B. Poole. Parallelized inference for gravitational-wave astronomy. *Physical Review D*, 100(4):043030, August 2019. doi: 10.1103/PhysRevD.100.043030.

[316] C. Pankow, P. Brady, E. Ochsner, and R. O'Shaughnessy. Novel scheme for rapid parallel parameter estimation of gravitational waves from compact binary coalescences. *Phys. Rev. D*, 92:023002, Jul 2015. doi: 10.1103/PhysRevD.92.023002. URL https://link.aps.org/doi/10.1103/PhysRevD.92.023002.

[317] B. P. Abbott et al. GW190425: Observation of a Compact Binary Coalescence with Total Mass $\sim 3.4\ M_{\odot}$. , 892(1):L3, March 2020. doi: 10.3847/2041-8213/ab75f5.

[318] Tyson B. Littenberg and Neil J. Cornish. Bayesian inference for spectral estimation of gravitational wave detector noise. , 91(8):084034, Apr 2015. doi: 10.1103/PhysRevD.91.084034.

[319] Oleg Mazonka. Easy as pi: The importance sampling method. *Journal of Reference*, 06 2016.

[320] George Casella, Christian P. Robert, and Martin T. Wells. Generalized accept-reject sampling schemes. *Lecture Notes-Monograph Series*, 45:342–347, 2004. ISSN 07492170. URL http://www.jstor.org/stable/4356322.

[321] D. I. Jones. Parameter choices and ranges for continuous gravitational wave searches for steadily spinning neutron stars. *Monthly Notices of the Royal Astronomical Society*,

453(1):53–66, 08 2015. ISSN 0035-8711. doi: 10.1093/mnras/stv1584. URL https://doi.org/10.1093/mnras/stv1584.

[322] Christos Alexakis, Michael Dowling, Konstantinos Eleftheriou, and Michael Polemis. Textual Machine Learning: An Application to Computational Economics Research. *Computational Economics*, 57(1):369–385, 2021. ISSN 1572-9974. doi: 10.1007/s10614-020-10077-3. URL https://doi.org/10.1007/s10614-020-10077-3.

[323] Golnoosh Babaei and Shahrooz Bamdad. A New Hybrid Instance-Based Learning Model for Decision-Making in the P2P Lending Market. *Computational Economics*, 57(1):419–432, 2021. ISSN 1572-9974. doi: 10.1007/s10614-020-10085-3. URL https://doi.org/10.1007/s10614-020-10085-3.

[324] Elie Bouri, Konstantinos Gkillas, Rangan Gupta, and Christian Pierdzioch. Forecasting Realized Volatility of Bitcoin: The Role of the Trade War. *Computational Economics*, 57 (1):29–53, 2021. ISSN 1572-9974. doi: 10.1007/s10614-020-10022-4. URL https://doi.org/10.1007/s10614-020-10022-4.

[325] Binhua Tang, Zixiang Pan, Kang Yin, and Asif Khateeb. Recent advances of deep learning in bioinformatics and computational biology. *Frontiers in Genetics*, 10:214, 2019. ISSN 1664-8021. doi: 10.3389/fgene.2019.00214. URL https://www.frontiersin.org/article/10.3389/fgene.2019.00214.

[326] Chunming Xu and Scott A Jackson. Machine learning and complex biological data. *Genome Biology*, 20(1):76, 2019. ISSN 1474-760X. doi: 10.1186/s13059-019-1689-0. URL https://doi.org/10.1186/s13059-019-1689-0.

[327] Adi L Tarca, Vincent J Carey, Xue-wen Chen, Roberto Romero, and Sorin Drăghici. Machine learning and its applications to biology. *PLOS Computational Biology*, 3(6): 1–11, 06 2007. doi: 10.1371/journal.pcbi.0030116. URL https://doi.org/10.1371/journal.pcbi.0030116.

[328] Alexandre Tkatchenko. Machine learning for chemical discovery. *Nature Communications*, 11(1):4125, 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-17844-8. URL https://doi.org/10.1038/s41467-020-17844-8.

[329] Volker L Deringer, Miguel A Caro, and Gábor Csányi. A general-purpose machine-learning force field for bulk and nanostructured phosphorus. *Nature Communications*, 11 (1):5461, 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-19168-z. URL https://doi.org/10.1038/s41467-020-19168-z.

[330] Johannes T Margraf and Karsten Reuter. Pure non-local machine-learned density functional theory for electron correlation. *Nature Communications*, 12(1):344, 2021. ISSN 2041-1723. doi: 10.1038/s41467-020-20471-y. URL https://doi.org/10.1038/s41467-020-20471-y.

[331] Sergei Manzhos. Machine learning for the solution of the schrödinger equation. *Machine Learning: Science and Technology*, 1(1):013002, apr 2020. doi: 10.1088/2632-2153/ab7d30. URL https://doi.org/10.1088/2632-2153/ab7d30.

[332] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to Simulate Complex Physics with Graph Networks. *arXiv e-prints*, art. arXiv:2002.09405, February 2020.

[333] Ayya Alieva, Dmitrii Kochkov, Jamie Alexander Smith, Michael Brenner, Qing Wang, and Stephan Hoyer. Machine learning accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences USA*, 2021.

[334] Bai-Jiong Lin, Xiang-Ru Li, and Wo-Liang Yu. Binary neutron stars gravitational wave detection based on wavelet packet analysis and convolutional neural networks. *Frontiers of Physics*, 15(2):24602, 2019. ISSN 2095-0470. doi: 10.1007/s11467-019-0935-y. URL https://doi.org/10.1007/s11467-019-0935-y.

[335] Wei Wei and E.A. Huerta. Deep learning for gravitational wave forecasting of neutron star mergers. *Physics Letters B*, 816:136185, 2021. ISSN 0370-2693. doi: https://doi.

org/10.1016/j.physletb.2021.136185. URL https://www.sciencedirect.com/science/article/pii/S0370269321001258.

[336] Joe Bayley, Chris Messenger, and Graham Woan. Robust machine learning algorithm to search for continuous gravitational waves. *Phys. Rev. D*, 102:083024, Oct 2020. doi: 10.1103/PhysRevD.102.083024. URL https://link.aps.org/doi/10.1103/PhysRevD.102.083024.

[337] Andrei Utina, Francesco Marangio, Filip Morawski, Alberto Iess, Tania Regimbau, Giuseppe Fiameni, and Elena Cuoco. Deep learning searches for gravitational wave stochastic backgrounds. In *2021 International Conference on Content-Based Multimedia Indexing (CBMI)*, pages 1–6, 2021. doi: 10.1109/CBMI50038.2021.9461904.

[338] M Cavaglià, S Gaudio, T Hansen, K Staats, M Szczepańczyk, and M Zanolin. Improving the background of gravitational-wave searches for core collapse supernovae: a machine learning approach. *Machine Learning: Science and Technology*, 1(1):015005, feb 2020. doi: 10.1088/2632-2153/ab527d. URL https://doi.org/10.1088/2632-2153/ab527d.

[339] Ang Li, Ola Spyra, Sagi Perel, Valentin Dalibard, Max Jaderberg, Chenjie Gu, David Budden, Tim Harley, and Pramod Gupta. A Generalized Framework for Population Based Training. *arXiv e-prints*, art. arXiv:1902.01894, February 2019.

[340] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *arXiv e-prints*, art. arXiv:1206.2944, June 2012.

[341] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[342] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Inv-

ernizzi, et al. Keras tuner. https://github.com/keras-team/keras-tuner, 2019.