Real-Time Predictive Artificial Intelligence: Deep Reinforcement Learning for Closed-Loop Control Systems & Open-Loop Signal Processing

Sama Daryanavard

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

School of Engineering
College of Science and Engineering
University of Glasgow

July 25, 2024

Abstract

Reactive mechanisms, such as reflexes, respond to disturbances only after they have occurred. In contrast, learning entities operate on principles of anticipation and prediction, enabling them to preemptively counteract potential disturbances. This research introduces a framework that integrates learning capabilities into traditional reflex systems, creating a comprehensive closed-loop platform tailored for reinforcement learning, particularly in robotics. Central to our approach is the use of backpropagation through deep neural networks. Although inherently an open-loop algorithm, we demonstrate through mathematical derivation that minimising the reflex error is equivalent to minimising the unknown open-loop error. We illustrate how the reflex error can be utilised to train the system.

Our innovative method involves applying backpropagation within closed-loop control systems, utilising z-transformation and intricate mathematical derivations. This approach offers significant advantages over existing algorithms. It functions as an online algorithm that learns in real-time, eliminating the need for pre-training or the use of physics engines. Additionally, it is particularly well-suited to continuous state-space applications, such as robotics, where defining all discrete states is non-trivial or impossible. One of the most striking advantages is the speed of convergence, which approximates one-shot learning due to the availability of the error signal from the reflex at every time-step for training the network, unlike scenarios where the reward or punishment signal is sparse. This makes the developed learning algorithm very fast compared to conventional Reinforcement Learning approaches.

This research culminates in the development of four distinct algorithms: CLDL, SAR, PAM, and Echo learning, each characterised by unique attributes and intricacies. These algorithms are inspired by biological processes and the functioning of the human brain. We rigorously tested these algorithms on a line-following robot, conducting experiments in both real-world scenarios and simulated environments to ensure reproducibility. The outcomes demonstrate successful path navigation by the robot without relying on its inherent reflex mechanism.

Furthermore, we present evidence that our platform can be effectively adapted for unsupervised open-loop systems with minimal adjustments. We show how EMG noise is

removed from EEG signals without any training. This study not only proposes a novel approach to integrating learning into robotic reflex systems but also broadens the potential applications of such algorithms in various automated processes.

This work makes significant contributions to the field of reinforcement learning by providing an open-source C++ logic library containing the algorithms for all aforementioned paradigms. Additionally, it includes two deployment libraries that demonstrate how to implement these algorithms on both simulated and real-world robots, as well as another library for signal processing, such as noise cancellation.

Contents

List of Figures

List of Acronyms

RL reinforcement learning
TD temporal difference
Q-learning Q-learning
DQN deep Q-network
NLP natural language prosessing
ML machine learning
AI artificial intelligence
DL deep-learning
ANN artificial neural network
DNN deep neural network
RNN recurrent neural network
NN neural network
BP back-propagation
BPTT back-propagation through time
GSV grey-scale value
GUI graphical-user interface
RPi raspberry pi 3B+
LDR light dependent resistor
MC motor command
FIR finite impulse response
FB filter bank
FoV field of view
G grey value
SaR sign and relevance
LTP long-term potentiation
LTD long-term depression
T.F. transfer function
GDM gradient descent method
EVGP exploding and vanishing gradient problem
DRL deep reinforcement learning
DNN deep neural network
CNN convolutional neural network
FEL feedback error learning
HSPC hierarchical sensory predictive control
FP forward-propagation
CLDL closed-loop deep learning
RMS root mean square
BP back-propagation

PID proportional integral derivative
GSV gray-scale value
IR infrared
LMS least mean-squares
LSTM long short-term memory
ICO input correlation
PaM prime and modulate
FCL forward propagation closed-loop learning
ADC analog to digital converter
DNF deep neuronal filter
SNR signal to noise ratio
EEG electroencephalogram
EMG electromyogram
EOG electroocoulogram
MDP markov decision problem
MSE mean squared error
ICA independent component analysis
BCI brain computer interface
PCA principal component analysis
EPH equilibrium point hypothesis
STDP spike-timing-dependent plasticity
STFT short-time fourier transform
FLNN functional-link neural network
SGD stochastic gradient descent
OFC orbitofrontal cortex
LTI linear time-invariant
CNS central nervous system
PID proportional integral derivative
GABA gamma-aminobutyric acid
CUDA compute unified device architecture
MOSAIC modular selection and identification for control

Teaser

It is another rainy day in Glasgow, and the temperature outside has dropped to around 3°C. Inside, the central heating system is working hard to warm up the flat. With a continuous hum, it cycles on and off to maintain the indoor temperature at a cosy 19°C. However, has the heating system learned how to maintain the set temperature? No.

On most mornings, the postman knocks on the door, and I typically open the door a few seconds later, resulting in some loss of heat. The heating system does nothing to prevent this. Instead, once the door has been opened and a drop in temperature has been detected, it attempts to compensate for the already-lost heat. In reality, upon closer inspection, the heating system never quite maintains the temperature at a consistent 19°C; it is more of an illusion. It repeatedly falls just below or rises slightly above the set temperature, perpetuating this cycle. This system is limited to the simplistic response pattern of:

"if too much, do less, and if too little, do more."

This does not exemplify intelligence; it resembles more of a reactive mechanism, such as a reflex that is always late. An intelligent system, on the other hand, would truly maintain the temperature at a constant 19°C by executing precisely the right action at the right time. Intelligent systems have a proactive working pattern of:

"do the right thing at the right time, so as to not be too much or too little."

But what would it take for the heating system to truly learn? Section 2.2 answers this question. This thesis is concerned with the study of such learning systems. This is the realm of artificial intelligence (AI), and more specifically, reinforcement learning (RL).

Acknowledgements

I am profoundly grateful to my esteemed supervisor, Bernd Porr, for his unwavering support throughout my academic journey. From the very beginning, he demonstrated immense confidence in my abilities by entrusting me with this research project. His patience and guidance, particularly during my learning of C++ coding and fundamental AI concepts, have been invaluable. What sets him apart is his ability to provide direction while granting me the independence to explore and develop my research. His vast knowledge in AI and engineering, coupled with his exceptional teaching skills, has significantly broadened my understanding. His approachable nature and positive energy have been a crucial source of motivation during the challenging phases of my PhD journey. His steadfast support during personal hardships is a testament to his outstanding character. I am honoured to have had him as my supervisor and eagerly anticipate future collaborations.

I extend my heartfelt gratitude to my second supervisor, Steven Neale, for his invaluable guidance and mentorship. My previous experience working with Steven during my Master's project on Optoelectronic tweezers was transformative. Under his supervision, my project not only thrived but also received two prestigious awards, significantly contributing to the success of my research. His open-door policy for guidance provided constant reassurance and motivation.

I would also like to express my sincere appreciation to Nick Bailey, a colleague of my supervisor, whose insights into coding best practices greatly enriched my research. Nick's expertise and support were pivotal to my academic development. Additionally, I am deeply thankful to our lab technician, Tom O'Hara, who provided the necessary equipment for building my robot. His support was instrumental in the success of this work, and his positive energy fostered a welcoming lab atmosphere. I also wish to acknowledge Hazel Blair for her invaluable support throughout this journey.

Acknowledging my older brother, Moosa Daryanavard, who has been a constant source of inspiration since my youth. His success as an engineer instilled in me a deep appreciation for the field and the importance of curiosity and innovation. Moosa's support has profoundly shaped my academic and professional achievements.

Lastly, I extend my deepest thanks to my partner, Jarez Patel, who provided unwavering support throughout our joint PhD journey. I genuinely believe I would not have completed this challenge without his constant presence, motivation, and collaborative spirit. Our shared journey has been enriching, and I am deeply thankful for having him as my partner.

University of Glasgow

College of Science & Engineering

Statement of Originality

Name: Sama Daryanavard
Registration Number: xxxxxxxx

I certify that the thesis presented here for examination for a PhD degree of the University of Glasgow is solely my own work other than where I have clearly indicated that it is the work of others (in which case the extent of any work carried out jointly by me and any other person is clearly identified in it) and that the thesis has not been edited by a third party beyond what is permitted by the University's PGR Code of Practice.

I declare that the thesis does not include work forming part of a thesis presented successfully for another degree.

I declare that this thesis has been produced in accordance with the University of Glasgow's Code of Good Practice in Research.

I declare that I have used ChatGPT as a tool for proof-reading and its use is in line with the update made to Academic Policy and Governance statement in April 2023.

I acknowledge that if any issues are raised regarding good research practice based on review of the thesis, the examination may be postponed pending the outcome of any investigation of the issues.

Signature: Sama Daryanavard
Date: 25 / 12 / 2023

Research Contributions

My research contributions encompass a diverse range of outputs, each playing a significant role in advancing the field of reinforcement learning and control systems.

Foremost among these contributions is the development of an open-source C++ library that encapsulates the novel backpropagation in z-space algorithm. By making this library open-source, I have enabled widespread accessibility and adaptability, allowing other researchers and developers to utilise, modify, and enhance this tool for various applications.

Additionally, I have developed two open-source application packages tailored for line follower robots. The first package is designed for a simulated robot, providing a safe and controlled setting for testing and refining the algorithm. The second package, in contrast, is developed for a real robot, bringing the theoretical aspects of the research into tangible reality.

My research journey has also been marked by significant academic contributions. I have authored one journal paper, with another currently under peer review and one available as a preprint. The published paper documents the intricacies of the backpropagation in z-space algorithm and the results from the experiments.

Furthermore, I have assembled a physical robot for the research team's use, which serves as a test-bed for ongoing and future experiments. This robot is not just a tool for validation but also an instrument for discovery, allowing the team to explore new ideas, test hypotheses, and refine our understanding of robotic learning mechanisms.

Lastly, a significant portion of my work has been dedicated to developing a comprehensive mathematical framework that underpins this type of learning. This framework serves as a blueprint, guiding researchers and practitioners in understanding, implementing, and advancing learning algorithms in robotics. It lays down the theoretical foundations, mathematical models, and practical guidelines, thus encapsulating the essence of my research in a structured and accessible format.

In summary, my research contributions extend from theoretical frameworks to practical implementations, encompassing open-source software, academic publications, and physical resources, all aimed at advancing the field of reinforcement learning and control systems.

Publications

This work has been gaining increasing recognition and acknowledgment within the research community. The ideas presented here are characterised by their novelty, introducing innovation and advancements to the field of AI.

- **Daryanavard, S.**, & Porr, B. (2020). Closed-loop deep learning: generating forward models with backpropagation. Neural Computation, 32(11), 2122-2144.
- **Daryanavard, S.**, & Porr, B. (2021). Sign and Relevance learning. arXiv preprint arXiv:2110.07292. (Pending publication in Sage Adaptive Behaviour)
- **Daryanavard, S.**, & Porr, B. (2023). Prime and Modulate Learning: Generation of forward models with signed back-propagation and environmental cues. arXiv preprint arXiv:2309.03825.
- Porr, B., **Daryanavard, S.**, Bohollo, L. M., Cowan, H., & Dahiya, R. (2022). Real-time noise cancellation with deep learning. PLOS ONE, 17(11), e0277974.

Notably, the SaR algorithm is under review and pending publication, PaM is available as a preprint, and the Echo algorithm is currently being written.

Chapter 1

Preface

1.1 Objectives

In the Teaser, a question was posed: How can a heating system not only react but adaptively learn to maintain a cosy temperature of 19°C, even in the face of changes such as the brief temperature drop caused by a postman's visit? The concept of reactive systems was introduced, highlighting that while these systems are efficient in their response to immediate changes, they lack the capacity to learn from recurring events. This work proposes a novel framework that integrates learning capabilities into such existing reactive systems.

Biological reflex mechanisms serve as prime examples of reactive systems, inspiring the novel learning paradigm introduced here. Their role in organismal learning is the cornerstone of this approach. In the engineering context of this work, however, a biological reflex is represented by a uniquely designed feedback control loop. Consequently, the incorporation of learning capabilities into the reflex is depicted by a tailored control loop, supported by biological arguments[1].

This project can be broadly segmented into two primary tasks:

1.1.1 Development of the Integrated Platform

The first task is to create a comprehensive platform that seamlessly integrates and interlocks reflex mechanisms with learning processes. The aim is to facilitate collaborative operations and signal exchanges between these components. A key aspect of this development is the derivation of control systems, utilising z-transformation techniques to enhance functionality and responsiveness.

[1] Thus, in this work, the terms "reflex" and "feedback control loop" are used interchangeably.

1.1.2 Construction of the Learning Unit

The second task involves the creation of a learning unit as a critical component to be embedded within the platform. This involves the establishment of a robust mathematical and technical framework. The framework is designed to optimally position the learning unit within the system, ensuring it functions effectively in concert with other elements of the platform.

The deliberate separation between the platform and the learning units underscores the adaptability and flexibility of the final product. This design decision allows for significant versatility: the platform is engineered to be compatible with a wide array of learning algorithms, as long as they meet the specified input-output criteria. This universal design ensures that the platform can be utilised with different learning algorithms, enhancing its utility and scope. As a result of this flexibility, the learning algorithm becomes highly versatile and domain-agnostic, transcending the limitations of being confined to a single use-case scenario. The structure of this system acts as a comprehensive guide or blueprint for incorporating learning capabilities into existing reflex systems. This aspect is pivotal, as it offers a standardised approach to enhancing various applications with advanced learning functionalities. To ensure the effectiveness and reliability of this approach, there is a strong focus on detailed and rigorous mathematical derivations. This foundational work is essential to validate the algorithm's effectiveness and robustness in diverse applications.

1.1.3 Application of the Algorithms

As a proof of concept, the developed algorithm is implemented on a line-following robot, demonstrating its ability to learn and navigate a path without relying solely on reactive actions. However, it is important to note that the applicability of this algorithm extends far beyond this particular task. It can be effectively employed in any reactive system that engages with its environment to maintain a desired state.

With this purpose in mind, the algorithm is transformed into a stand-alone and custom-built C++ logic library designed for versatile applications (Daryanavard and Porr, 2020b). Additionally, to further enhance the practical applicability of these algorithms, this library was also developed as a CUDA logic library for implementation on GPUs, leveraging parallel programming (Porr, 2021).

Additionally, two separate packages are developed for the deployment of the logic library. Specifically, for the line-following task, a deployment package in simulation (Porr and Daryanavard, 2020) and a deployment package on a physical robot (Daryanavard and Porr, 2020c) are developed, serving as an illustrative example of how this library can be effectively applied for the broader objective of predictive learning in various contexts.

1.2 Thesis Outline

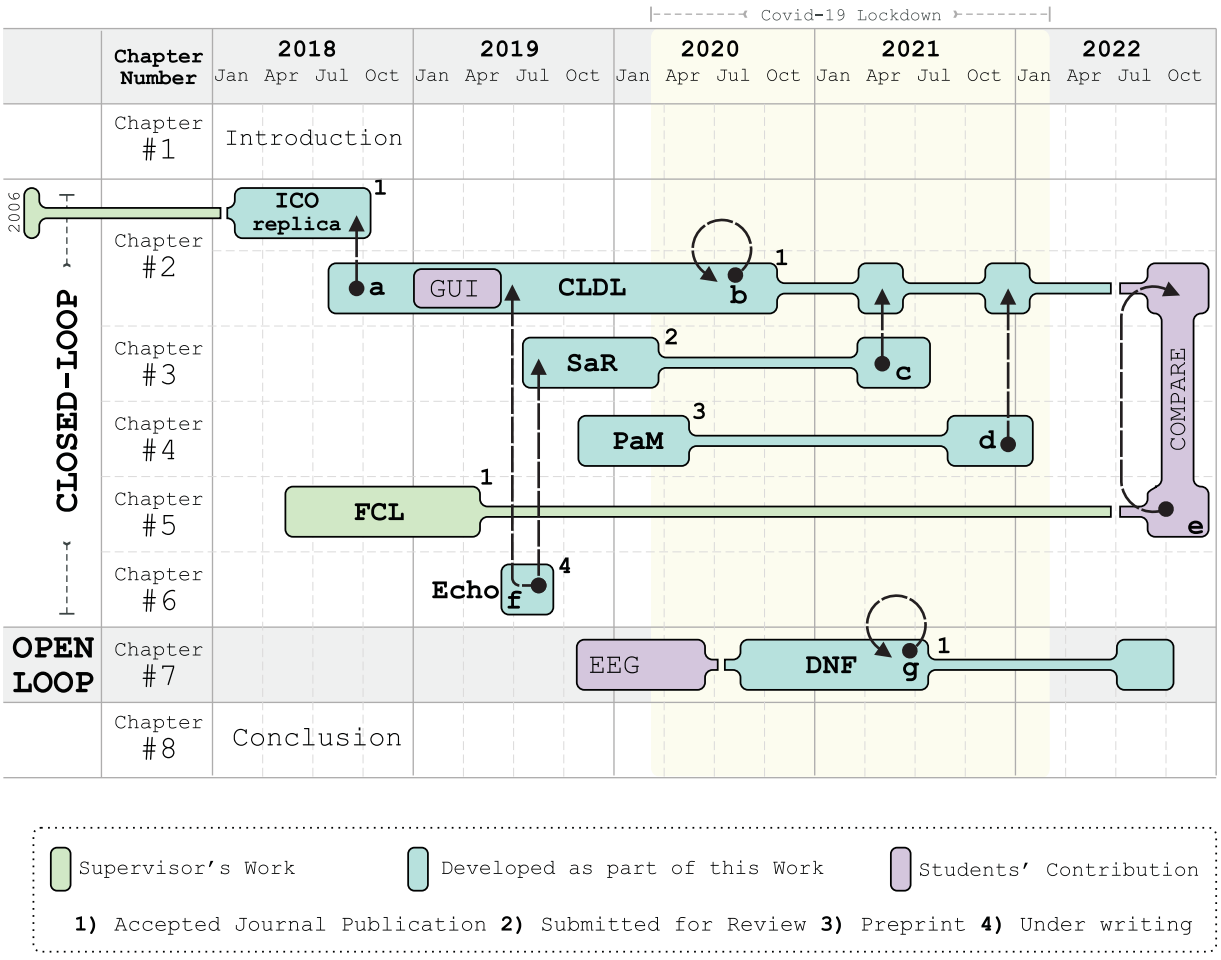

Figure 1.1: *Timeline of projects in this work and organisation and order of its chapters.*

The above objective was realised during the second year, marked by a publication that verified the novelty, validity, and acceptance of the algorithm. Subsequently, in the following years, three additional algorithms were created, building upon the foundation established by the initial one, driven by fresh research insights and innovative ideas. This thesis comprises six chapters, each representing a substantial portion of the work as independent projects. These chapters follow a predominantly chronological order, with a few exceptions.

Figure 1.1 provides a visual representation of the timeline for each major project, illustrating their progression from inception through development, experimentation, and, where applicable, publication. In this figure, the dark grey segments denote the relevant contributions made by the supervisor, while the white segments represent the work undertaken as part of this research. The light grey portions signify the involvement of undergraduate students in the projects. The arrows in Figure 1.1 depict the experimental results and their relation to the comparative analysis of different algorithms.

- Chapter 2 delves into the ICO learner and CLDL, offering comparative results between the two (a), as well as a comparison of CLDL learning to its own reflex (b). Replicating input correlation (ICO) learner, initially introduced by the supervisor in 2006 (Porr and Wörgötter, 2006), served as an ideal warm-up project for this research.

  This led to the creation of the CLDL algorithm, which constituted the primary objective and focus of this work. Subsequent projects emerged as supplementary outcomes of this effort. The development of the initial GUI interface was undertaken by Bruno Manganelli and his team as part of their undergraduate project.

- Chapter 3 introduces an algorithm rooted in CLDL but inspired by neuroscience principles. This chapter presents the results of the comparison between these two algorithms (c).

- Chapter 4 presents the PaM algorithm, which extends the concepts from SaR with additional features and functionalities. This chapter includes a comparison of PaM to CLDL (d).

- Chapter 5 introduces the FCL algorithm (Porr and Miller, 2020). Although this was developed by the supervisor, it is presented here due to its significance in relation to other algorithms within this work. Additionally, this chapter features a comparison between FCL and CLDL (e), as conducted by Innes Aitken.

- Chapter 6 presents the Echo learning algorithm, which draws inspiration from FCL and CLDL. The chapter showcases the results of this algorithm (f) in comparison to SaR and CLDL.

All the aforementioned algorithms are discussed within the context of closed-loop control, which will be extensively explored.

- Chapter 7 demonstrates how this platform can be adopted for open-loop applications, such as signal processing. This adaptation led to the development of the DNF algorithm, which was applied to EEG noise removal as a proof of concept. The setup and execution of experiments for this project were carried out by Henry Cowan and Lucia Munoz Bohollo as part of their undergraduate projects.

Finally, Chapter 8 provides a conclusion of this project.

Chapter 2

Closed-Loop Deep Learning (CLDL)

2.1 Introduction

In the teaser section, we introduced a heating system that operates reflexively. We posed a pivotal question: how can this system evolve to reliably and precisely maintain a set temperature? We promised to provide an answer. Essentially, the heating system must discern a correlation between the postman's arrival and the subsequent temperature drop. Then, at the sight of the postman arriving, it must adjust the heat output with precision, both in timing and intensity, to preemptively counteract the expected heat loss when the door opens.

This project aims to design a methodology for integrating a predictive learning module into an existing reflex-based system. In the following motivation section, we conceptualise and develop such an adaptive learning system. We will review relevant literature and historical developments of intelligent systems, situating our project within the broader context of similar endeavours. Throughout the following section, we will persistently revisit the heating system example to render our reasoning and arguments more concrete and relatable.

2.2 Motivation: Comprehensive Literature Exploration

2.2.1 Adaptive Learning in Nature

Learning and adaptation are omnipresent in nature, evident among humans, animals, and even microscopic organisms. These entities exhibit an exceptional ability to learn from their surroundings, adjusting adeptly to unexpected changes and anomalies. For example, after experiencing a burn, we instinctively become cautious around hot surfaces and avoid similar harmful situations. This behaviour prompts the question: how do living beings navigate and adapt to their environments with such ease? More relevant to this work, how can we design machines to emulate this kind of adaptive learning? The answer lies in our aspiration to create machines capable of achieving human-like proficiency in tasks without the need for explicit programming. This is the very essence of artificial intelligence (AI) (Russell, 2010).

2.2.2 Artificial Intelligence (AI)

2.2.2.1 Simulating Learning & Adaptation

AI strives to simulate the extraordinary learning and adaptive capacities seen in nature, aiming to equip machines with the ability to learn autonomously, adapt, and make decisions similar to human cognition. For decades, researchers have faced the challenge of capturing the essence of human-like learning within machines. In their pursuit of understanding and replicating this phenomenon, experts have sought inspiration from a plethora of disciplines. Consequently, AI has evolved as a multidisciplinary domain, intertwining elements of cognitive psychology, neuroscience, control theory, mathematics, statistics, and more (Winston, 1984). As a result, this field has emerged to encompass many subfields (Samoili et al., 2020). machine learning (ML) (Alpaydin, 2016), deep-learning (DL) (Goodfellow et al., 2016; LeCun et al., 2015), natural language prosessing (NLP) (Grosz et al., 1986), computer vision (Szeliski, 2022), robotics (Murphy, 2019), and expert systems (Turban, 1995) are a few examples of the breadth and depth of AI's reach. Within this vast landscape, this work, deeply rooted in the paradigm of trial and error, is predominantly anchored in reinforcement learning (RL), a prominent and influential subfield of AI (Sutton and Barto, 2018; Wiering and Van Otterlo, 2012; Li, 2017).

2.2.2.2 History, Developments & Milestones

The history of AI is a tale of human ingenuity and perseverance, spanning several decades of groundbreaking research and technological advancements. This began when a group of pioneering researchers such as John McCarthy, Marvin Minsky, and others set out to

create machines that could perform tasks traditionally done by humans (McCarthy, 1959; Nilsson, 2009). They are often honoured as the fathers of AI (Andresen, 2002).

One of the earliest works in AI was the Turing Test, proposed by Alan Turing in 1950. It was designed to determine whether a machine could exhibit intelligent behaviour equivalent to, or indistinguishable from, that of a human. It was a revolutionary idea that captured the imagination of many, and set the stage for a new era of AI research (Turing, 2012).

In the 1960s, AI research shifted its focus towards symbolic reasoning and expert systems. These systems used rules-based approaches to simulate human decision-making and marked a major milestone in the field. However, this approach eventually gave way to the development of machine learning algorithms, which could learn from data and improve over time (Buchanan and Smith, 1988).

The greatest achievement in AI came in 1997, when IBM's Deep Blue chess computer defeated world champion Garry Kasparov (Clark, 1997; Campbell et al., 2002). It was a historic moment that showcased the power of AI to out-perform even the best human minds. From there, machine learning continued to power breakthroughs in areas such as natural language processing (Mishra and Kumar, 2020) and computer vision, leading to advances in applications like speech recognition, image classification, and autonomous driving (Szeliski, 2022; Siciliano et al., 2008).

The journey to make machines think like humans has been paved with many notable works and achievements. The development of neural networks, the creation of the first autonomous vehicles, and the use of RL to teach machines to play games like Go are a few examples (Sutton and Barto, 2018; Silver et al., 2016). Today, natural language processing models like Google's BERT and OpenAI's GPT-3 are pushing the boundaries of what machines can do (Kublik and Saboo, 2022; Brown et al., 2020; Lund and Wang, 2023).

2.2.2.3 Interdisciplinary Approaches in AI

When developing advanced learning algorithms, it has become increasingly evident that interdisciplinary approaches, borrowing insights from fields as diverse as psychology and robotics, hold the promise of breakthroughs (Dwivedi et al., 2021). Naturally, this work also navigates through various subjects, weaving multiple threads of reasoning through time. Specifically, explorations in the psychology of adaptive behaviour and the neurophysiological study of the brain and the nervous system provide valuable insights into how organisms learn and adapt to their environments. From there, the realms of RL, control theory, and cybernetics offer technical frameworks and mathematical tools to design machines that can simulate the remarkable learning observed in organisms. In essence, we embark on a journey that spans both the depths of biological understanding and the

heights of computational excellence. Furthermore, we employ a robotic task to serve as a platform where theoretical models meet tangible actions in real-time.

Figure 2.1 presents a map of the aforementioned fields. The x-axis positions each field on a spectrum with biological fields on the left and engineering fields on the right, while the y-axis represents the spectrum from open-loop to closed-loop systems. Artificial intelligence is depicted at the centre as an overarching umbrella spanning the entire y-axis, though it does not extend to the biologically realistic end of the x-axis. Other fields are appropriately positioned on this map, with arrows indicating their flexibility and blocked ends showing their boundaries. The various algorithms, which will be introduced in the upcoming chapters, are placed around the edges of this map where they respectively sit.

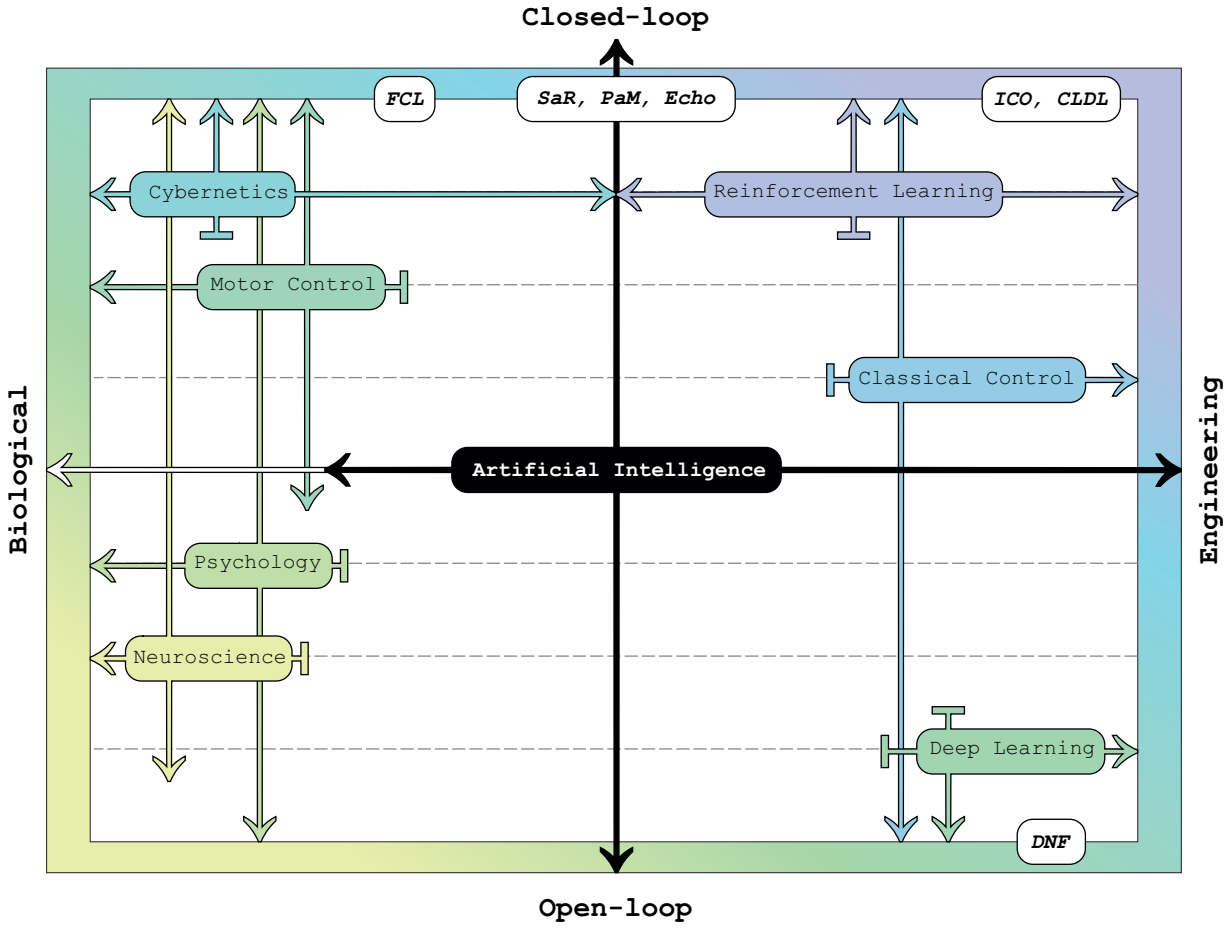

Figure 2.1: *A comprehensive map detailing the spectrum of fields discussed in the literature review. On the x-axis, fields are positioned from biological (left) to engineering (right). The y-axis represents the spectrum from open-loop to closed-loop systems.*

2.2.3 Concept of Disturbances

Returning to the heating system analogy, before we explore how a learning module can be designed and added to the reflex, a fundamental question to consider is: how can we

differentiate a learning system from a reflexive one? To put it simply, would we know it, if we were faced with an intelligent system?

Imagine a steady state scenario where external conditions remain unchanged, no postman arrives, and no doors or windows are opened. This allows the temperature to stay at a steady 19°C. In this static scenario, it is inconsequential whether the heating system is a basic reflex or an intelligent one; it simply does not need to act. In other words, in the absence of the postman, or any other disturbances, a learning heating system and a basic one are indistinguishable.

The concept of disturbances, as they relate to control systems, was first introduced by James Watt when he developed a feedback mechanism for controlling the speed of steam engines (Ghosh and Ghosh, 2015). However, in the field of cybernetics, Ashby defined disturbance as the phenomenon that causes a deviation from the desired state in an organism (Ashby, 1956; Porr and Wörgötter, 2005). Disturbances play a central role in Ashby's conception of cybernetics. For a system to deal with disturbances from its environment and maintain its stability, it must have a sufficient variety of responses. Learning, in this context, can be seen as a mechanism by which a system increases its internal variety to match or surpass the variety in disturbances it faces from its environment (Ashby, 1956).

The postman represents an unpredictable disruption that offsets the system from its balance. The agent or organism's primary goal is to preserve and uphold its desired state. What differentiates a simple reflex from a learning system is the manner in which they address the disturbances.

2.2.4 Reflexive Systems in Response to Disturbances

Let us consider this for the reflex. The desired state for reflex essentially equates to a desired input; that being a temperature of 19°C. This is because an agent can only infer its state in the environment via its sensory inputs (Porr and Miller, 2020). Reflexes are involuntary reactions to specific stimuli, allowing an organism to regain equilibrium after it is disturbed. At any given moment of time, the reflex senses the temperature of the room and compares this to its desired input temperature. The difference of the two generates an error signal which indicates that the desired state is lost. The reflex responds to this disturbance reactively after it has occurred. This inherent delay is the fundamental characteristic of a reflex; it is always late (Porr and Wörgötter, 2002a). Von Foerster termed this the "blind spot" of reflex (Porr and Wörgötter, 2005). The delay between the occurrence of a stimulus and the system's response can lead to unintended consequences, as the system might be responding to outdated information. We will show in this work how a learning module can overcome this challenge.

The concept of reflexes and desired state is fundamental to many disciplines, including motor control and classical control theory. The study of motor control has deep roots,

beginning in the 19th century when Paul Broca identified the brain's motor cortex as responsible for movement and speech (Gross, 2007). Later, Charles Sherrington explored how sensory information affects motor actions, introducing the concept of reflex and the term "synapse" (Swazey, 1968).

2.2.5 Reflex in Biology

Motor control's genesis emerged from two distinct arenas. One centred on neurophysiology, focusing on neural processes without considering the movement. In contrast, the second sprouted from psychology, examining high-level skills without delving into neurology. These threads intertwined in the 1970s, marking the birth of motor control (Schmidt et al., 2018; Shadmehr et al., 2010). A significant influencer was Anatol Feldman who presented the equilibrium point hypothesis (EPH), suggesting voluntary movements arise from shifting the limb's equilibrium position, not from direct commands for muscle forces or joint torques (Feldman, 1986). This emphasises reflexes' role in motor control. According to Feldman, the central nervous system (CNS) does not dictate muscle activations but modulates muscle reflex sensitivities to achieve movements. This perspective reshapes our understanding of motor learning, suggesting that practising refines equilibrium points and reflex sensitivities.

2.2.6 Reflex in Control Theory

In classical control theory, a reflex can be likened to a fixed feedback controller which measures system outcomes, contrasts them with desired outcomes, and makes necessary adjustments (Phillips and Harbor, 1991). However, key differences lie in their focal points: biological reflexes centre on the desired input (or state), whereas closed-loop controllers are geared towards the desired output (Porr and Miller, 2020). In fact, organisms only realise their outputs when sensory inputs provide feedback about them. The outputs and desired actions are only evident from the standpoint of an external observer (Porr and Wörgötter, 2005). This closed loop established between action and sensory inputs is crucial. In 1987, R.A. Schmidt pioneered this interrelation between action and perception (Schmidt, 1987, 2016). Prinz emphasised the significance of sensory inputs in what is termed as "ideo-motor" actions. This concept refers to movements that are executed in alignment with perceived movements, highlighting situations where actions appear to be directly guided by perception (Prinz, 2016). Furthermore, traditional control requires that the properties of the controlled system is known. In contrast, organisms, and the subject of this work, neither possess nor rely on such information about the environment (Phillips and Harbor, 2000). Instead, the learning module independently forms an understanding of the environment and its properties. This renders control theory approaches unsuitable

for explaining the learning in organisms. Nevertheless, the analytical framework provided by control theory will guide the subsequent sections, where we aim to derive mathematical representations for both the reflex and the learning entity.

2.2.7 Learning Systems in Response to Disturbances

Consider the heating system again, the postman's arrival momentarily disrupts the temperature, and the system's reflex restores the balance, then stays dormant until another disturbance takes place. While reflexes act post-event, a learning system proactively acts in anticipation of these disturbing events.

Just like the reflex, the learning system also continually processes environmental inputs, such as the temperature and a view of the street outside. However, a unique input to the learner is the error signal generated by the reflex during disturbances. This distinguishes between sensory input from the environment and the body itself. This concept was first introduced by RA Schmidt (Schmidt et al., 2018).

Unlike the reflex, the learner does not aim for a desired state, instead, it strives to consistently receive an error signal of zero. To achieve their respective goals, the reflex employs a fixed and pre-programmed response, whilst the learner learns to fine-tune its predictive actions (Schmidt et al., 2018).

2.2.8 Importance of Correlation in Learning

Initially, the learner has no skills but continuously observes its environment, retaining a short memory of these observations. This short memory, facilitated by low pass FIR filters (Proakis, 2007), becomes pivotal to the learning abilities of the learner which will be discussed and derived in the following Section 2.4.3.3. When the first disruption occurs, the inexperienced learner remains passive, allowing the disturbance to unsettle the reflex, causing it to react and rectify. The generated error is noticed by the learner. Lacking a predefined response, what does the learner do?

Confronted with the error, the learner consults its recent memory, identifying events preceding the error. Perhaps it observed the postman approaching and a bird flying off the tree. It notes a correlation between these events and the error. On receiving the next error, it notes the postman again but perhaps no bird activity. Over time, it solidifies the correlation between the postman and the error, dismissing other uncorrelated events.

2.2.9 Classical Conditioning

Observing correlations between events is prevalent in both human and animal behaviours. The learning system described above bears some resemblance to classical conditioning, a learning mechanism famously exemplified by Ivan Pavlov's experiments with dogs (Pavlov,

1932). In this study, a dog was trained to associate a bell's ring with impending food. While the dog initially salivated only at the sight or scent of food, after numerous instances of hearing the bell before being fed, it began to salivate just at the bell's sound. This demonstrated how a previously neutral stimulus (the bell) could, through association with an unconditioned stimulus (food), evoke a conditioned response — salivation. In classical conditioning, however, the agent remains passive, lacking any direct interaction with either one of the stimuli. This means, its actions do not influence subsequent events. This represents an open-loop scenario, which does not adequately capture the learning model we aim to put forward in this study.

2.2.10 Operant Conditioning

Unlike classical conditioning that centres on associating two stimuli, operant conditioning emphasises correlating behaviours with their ensuing consequences. In this learning approach, the agent actively engages with its environment, responds to stimuli, and assesses the outcomes of its actions. This dynamic interplay forms a closed-loop learning framework; this is a fundamental aspect of RL. B.F. Skinner, a renowned American psychologist, pioneered the concept of operant conditioning using the 'Skinner Box', a device that rewarded rats for pressing a lever (Skinner, 1963). Through techniques like "shaping", the rats were trained to associate pressing the lever with receiving food.

2.2.11 Learning & Early Disturbance Response

Applying this to the heating analogy, it is, therefore, crucial for the learner to do more than just observe correlations. The learner must act upon the initial stimuli, such as seeing the postman's arrival, to influence the subsequent stimulus, which is preventing the error signal. To successfully accomplish this, the learner acts with foresight. For instance, it might preemptively increase the temperature a few seconds after the postman rings the bell. By doing so, it effectively counteracts any potential heat loss when the door opens.

In order for the learner to form anticipatory actions, it requires a broader range of environmental inputs compared to the reflex. While the reflex only becomes aware once the door has been opened, the learner possesses the capability to observe the external environment and track unfolding events. This grants the learner a head start, allowing it to detect disturbances before the reflex, thus providing an opportunity to act and fend off the disturbance. In fact, the activation of the reflex serves as a temporal marker; events that take place before reaching the reflex are deemed early, while those occurring after the reflex are categorised as late (Porr and Wörgötter, 2005). Thus, the reflex action itself is by definition always late. In this way, the learner has knowledge of earlier disturbances which allows it to form predictive actions. This concept is mathematically formulated in

Section 2.4 using z-transformation (Diniz et al., 2010).

At a high level, this framework describes how a predictive learning module can be added to operate in conjunction with the reflex to proactively mitigate errors. A pressing question that emerges now is: how does the learner internally craft these anticipatory actions? It is the intricacies of the learner's internal architecture and mechanisms that empower it to undertake such adaptive learning.

Before delving into the intricacies of the learner's inner workings, it is essential to establish a comprehensive understanding of the learning platform. This platform outlines the interaction between the agent, encompassing both its reflex and learning modules, and the environment, including disturbances. It provides a blueprint for how inputs and outputs of these components are connected. A pivotal feature of this platform is its closed-loop learning system. While we have discussed closed-loop adaptive behaviour from a psychological perspective, translating this into machine learning requires an analytical framework. This is precisely what RL and control theory provide.

2.2.12 Markov Decision Problem

These fields both date back to 1950s when Richard Bellman pioneered the markov decision problem (MDP) as a methodology for modelling decision-making scenarios characterised by sequential actions and unpredictable outcomes. The "Markovian" property refers to the idea that the future state of the environment depends only on the current state and action, and not on prior states or actions (Bellman, 1957). This property simplifies the decision-making process as the agent only needs to consider the current state to determine the best action and not the history of states or actions. Bellman's mathematical framework laid the foundation for both RL and optimal control and subsequently became a cornerstone concept in decision theory and the broader realm of AI (Bellman, 1966).

2.2.13 Optimal Control

Optimal control is a subfield of control theory, offering a mathematical framework to devise control policies that optimise specific performance metrics of a system, such as minimising costs or maximising efficiency (Kirk, 2004). Its foundations predate those of RL, tracing back to the pioneering work by Euler and Lagrange in the 18th century on the calculus of variations (Bliss, 1930). By the 1960s, the field had evolved further, with Lev Pontryagin introducing 'Pontryagin's Maximum Principle' (Kopp, 1962). This principle provided the necessary conditions for achieving optimal control, marking a significant advancement in the domain (Boltyanskii et al., 1960).

Bellman's equation is a fundamental concept in RL which describes the recursive relationship between the value of a state and the values of its successor states (Bellman,

1966). This concept is analogous to the principle of optimality in dynamic programming, a method in optimal control (Lewis and Vrabie, 2009). In a conventional feedback control system, for example, there is a similar interplay between the controller (agent) and its controlled entity (environment) (Wörgötter and Porr, 2005). The distinguishing factor lies in their operational mechanisms. In control theory, the controller is predefined and the dynamics and properties of the environment are fully or partially known. Conversely, in RL, the agent *learns* how to function and does so in the backdrop of an unknown environment.

2.2.14 Reinforcement Learning (RL)

The history of RL dates back to the early days of AI research, but it was not until the 1990s that significant progress was made in the field (Sutton and Barto, 1998). This journey can be divided into two main threads that eventually merged to form modern RL. The first thread is related to learning by trial and error, which originated from the psychology of animal learning, as explained above. This thread influenced early work in AI and led to the revival of RL in the 1980s. The second thread focuses on optimal control and the use of value functions and dynamic programming to solve the problem of optimal control. The two threads remained largely independent, except for a third thread that involved temporal-difference methods. In the late 1980s, these three threads converged, giving rise to the field of RL as it is known today (Sutton and Barto, 2018).

Fundamentally, RL faces the question: *what to do to achieve a specific goal?* Drawing from our prior example, this involves determining the timing and extent of temperature adjustments when the postman is seen so as to prevent reflex errors when the door is opened. In this example, there is no direct instruction dictating the exact actions to undertake. Instead, the learner is intimately connected to its environment through its sensorimotor pathways. Positioned within a given state in the environment, the agent executes an action. This action affects the environment, leading the agent to a subsequent state and it receives a form of reinforcement signal — reward or penalty. Recognising this new state via sensory feedback, the agent deduces implications about its actions and surrounding conditions, setting the stage for its next move. This cyclical interaction describes a closed-loop system between the agent and the environment, embodying the foundational framework of trial-and-error learning (Sutton and Barto, 2018).

2.2.15 Trial-and-Error Learning

Learning by trial-and-error, has been a major focus in the history of RL. This idea can be traced back to Edward Thorndike's 'Law of Effect', which states that actions followed by satisfying outcomes are more likely to be repeated, while actions followed by discomfort

are less likely to occur (Thorndike, 1927).

Early computational investigations of trial-and-error learning were conducted in the 1950s and 1960s. Researchers explored the use of analogue machines and neural networks for RL tasks (McCulloch and Pitts, 1943). However, there was some confusion between trial-and-error learning and supervised learning, leading to a lack of research in genuine trial-and-error learning during this time and arguably today. There were a few exceptions to this trend, such as Donald Michie's development of RL systems for playing tic-tac-toe and balancing a pole (Michie, 1963).

The learning approach discussed in this context resonates with reinforcement learning (RL) principles where agents learn from feedback in the form of success or failure signals rather than predetermined training examples in supervised learning. Using the heating analogy, learning-by-example method would necessitate instructing the agent on all potential scenarios concerning the postman's arrival. Each situation would come with a precise labelled action for the agent to execute. For instance, the agent would be told, should the postman approach the door while it's raining, increase the temperature by precisely 2°C, and do so exactly 3 seconds after the doorbell rings. This approach quickly becomes unfeasible due to the innumerable combinations of parameters. Some of these parameters might be unidentified, or they may be governed by complex, unpredictable rules. Conversely, with the trial-and-error approach intrinsic to RL, the agent autonomously decides on an action, such as raising the temperature by 1°C. Subsequently, it receives a feedback signal, which could be zero, small, or substantial, representing the accuracy of its action. Over time, the agent adjusts its actions based on this feedback, striving for optimal performance without needing predefined labelled examples (Sutton and Barto, 2018).

In supervised learning, there is a clear separation between training and performance. The agent undergoes an extensive training period, learning from numerous scenarios and their desired outcomes. Once trained, the agent then operates based on this accumulated knowledge. Contrastingly, in this work, there is no distinct training and performance phases. The learner begins with no task knowledge and progressively learns within each trial, seamlessly integrating learning and performance to achieve its objective.

The revival of the trial-and-error practice in RL can be attributed to Harry Klopf, who emphasised the importance of hedonic aspects and the drive to achieve desired outcomes in adaptive behaviour. Klopf's ideas highlighted the distinction between supervised learning and RL (Klopf, 1986; Sutton and Barto, 2018).

In recent years, RL has led to breakthroughs in areas such as game playing (Mnih et al., 2015; Silver et al., 2016, 2017), robotics (Lillicrap et al., 2015), and NLP (Mishra and Kumar, 2020). Some of the key achievements in the field of RL include the development of AlphaGo, a computer program that defeated a human world champion at the game of Go

(Silver et al., 2016), and AlphaZero, a program that achieved superhuman performance in multiple board games without any prior knowledge of the rules (Silver et al., 2017). RL has also been applied to robotics, where agents have learned to perform tasks such as grasping objects and navigating environments (Lillicrap et al., 2015).

2.2.16 Temporal Difference (TD) Learning

Temporal difference (TD) learning is a foundational concept within RL. In the 1980s, Richard Sutton introduced TD learning, which has become a cornerstone technique in RL (Sutton, 1988b; Sutton and Barto, 2018). This approach allows algorithms to learn from each experience and adjust their behaviour accordingly by updating the expected reward of an action based on the difference between the expected and actual reward received. While TD learning has proven to be a powerful and flexible approach to RL, it is not without its drawbacks. One major limitation is that TD learning relies on a fixed learning rate, which can lead to slow convergence or instability in some situations. Additionally, TD learning can suffer from overfitting and may not generalise well to new or unseen environments. Despite these, TD learning remains a popular and widely used approach in RL research (Mnih et al., 2015; Sutton and Barto, 2018), which has been refined and extended in numerous works, including the Q-learning (Q-learning) (Watkins and Dayan, 1992), and the SARSA algorithm (Rummery and Niranjan, 1994).

2.2.17 Q-Learning

Q-learning is one of the most well-known and foundational algorithms in the RL domain. Introduced by Peter Dayan (Watkins and Dayan, 1992) in the 1990s, Q-learning established itself as one of the most widespread RL algorithms. At its core, it is a method for finding optimal policies for MDPs by learning a Q-function that assigns a value to each state-action pair. The Q-function represents the expected reward for taking a particular action in a given state and following the optimal policy thereafter. This is a value-based method. These methods estimate the value of different actions or states and use this information to make decisions about which actions to take (Sutton and Barto, 2018).

Despite its success, Q-learning has some limitations. Namely, it requires complete knowledge of the MDP, which is often not available in practice, and it can suffer from overestimation of Q-values, which can lead to suboptimal policies (Sutton and Barto, 2018). One important limitation of Q-learning is that it assumes a discrete state space, which restricts its applicability to problems with a finite number of states. In many real-world problems, however, the state space may be continuous, making it impossible to represent the state space using a discrete grid (Sutton and Barto, 2018).

Unlike Q-learning, our work employs an RL approach in a continuous state space which

is more readily applicable to robotic applications. This allows our system to better handle complex environments where there are infinite number of states, and state transitions occur smoothly and continuously. This enables the agent to learn and adapt more effectively to changing conditions in real-world scenarios.

While conventional RL approaches are designed for discrete state and action spaces with discrete-time dynamics, optimal control offers solutions for continuous state and action spaces and with continuous-time dynamics. The modern landscape of control theory and RL research often blurs the boundaries between these fields, leading to more holistic and powerful approaches to decision-making problems (Lewis et al., 2012; Lillicrap et al., 2015), as is the case in this work.

However, continuous RL is also an emerging field that aims to bridge this gap (Gu et al., 2016). To address this limitation, researchers have developed extensions of Q-learning that can handle continuous state spaces. For example, the use of function approximation or discretisation techniques can be used to approximate the Q-function in continuous state spaces (Mnih et al., 2015). Other approaches, such as actor-critic methods and policy gradient methods, are specifically designed to handle continuous state and action spaces (Sutton and Barto, 2018).

In the context of RL and, more specifically, in actor-critic methods, 'actor' and 'critic' refer to two distinct components of the learning system that work together to optimise the agent's behaviour. These methods aim to combine the advantages of value-based learning and policy-based learning.

The actor is responsible for determining the best action given a particular state, essentially defining the policy. The critic, on the other hand, evaluates the action taken by the actor by computing the value function of the state, effectively providing feedback on the actor's decisions. The actor uses this feedback to refine its policy. Thus, in this method, while the actor is learning the optimal policy, the critic assists by evaluating the goodness of the actions taken based on the expected rewards (Szepesvári, 2022).

The learning approach in this study also utilises a dual mechanism, enabling the learner to refine its actions based on continuous feedback from the reflex error. What sets our work apart from the actor-critic method, is the multifaceted role of the reflex error. Not only does it offer feedback for the learner's improvement, but it also actively intervenes to re-establish the desired state.

Let us recap what has been explained about the platform so far. The platform consists of a biologically-inspired reflex, conceptualised as a fixed feedback loop in control. Disturbances are directed to the reflex's input. The output of the reflex is guided by the error it perceives which acts on the environment. Additionally, the platform integrates a learning unit inspired by the psychology of adaptive behaviour. It receives early disturbances as input. Its output comprises predictive actions which also act on the en-

vironment. The learner is formulated as an outer closed loop which surrounds the inner reflex. At this point, they remain decoupled except for the mutual influences they exert on the environment. Crucially, to couple these loops and facilitate learning, the reflex's error is channelled to the learner as an input. Having provided a comprehensive outline of the platform, we will now delve into the intricate inner mechanics of the learner.

2.2.18 Neuroscientific Foundations of Learning

At the beginning of this section, we explained the operational mechanics of a learning module through the lens of the psychology of adaptive behaviour and the principles of RL. However, the intricate internal mechanisms of such learners are derived from neuroscience (Uden and Guan, 2022). The multifaceted architecture and nuanced operations of the human brain and the entire nervous system have provided researchers with a wealth of insights and inspirations for designing AI algorithms. These biological concepts not only offer an understanding of cognition and neural communication but also serve as invaluable blueprints, driving innovation and sophistication in the design of AI algorithms. This synergy between the realms of neuroscience and AI paves the way for more biologically-aligned and efficient learning systems, bridging the gap between nature's design and human-engineered intelligence.

Neuroscience is the study of the nervous system and its functions and has a rich history that spans centuries. Early civilisations recognised the significance of the brain. The Greek physician Hippocrates proposed that the brain was the seat of intelligence and consciousness (Breitenfeld et al., 2014). The brain is the central organ of perception, cognition, emotion, and action, and it operates through intricate electrochemical signalling in a complex, dynamic, and adaptable network of neurons (Uden and Guan, 2022).

2.2.18.1 Regions of the Brain

The brain is organised into distinct regions, both structurally and functionally, with each area dedicated to specific tasks. The Cerebral Cortex, the outermost layer of the brain, plays a pivotal role in higher-order cognitive functions such as thinking, planning, and language (Mountcastle, 1998). It is segmented into the frontal, parietal, temporal, and occipital lobes. Wilder Penfield made significant contributions to understanding the brain's layout by pioneering the mapping of the human cortex (Penfield and Rasmussen, 1950). He achieved this by stimulating various brain areas in epilepsy patients during surgery. Another remarkable revelation about the brain came from a 19th-century incident involving a railroad worker. After surviving an accident where a tamping iron pierced through his brain, the man exhibited profound personality changes. This unexpected event shed light on the crucial role that the frontal lobe plays in governing personality and behaviour

(Damasio et al., 1994).

2.2.18.2 Neurons & Synapses

Neurons are the primary signalling units of the brain, organised into complex networks or circuits. In the late 19th and early 20th century, scientists developed staining techniques to visualise these neurons. While Golgi believed in the reticular theory, suggesting that the nervous system was a single continuous network, Cajal's observations supported the neuron doctrine by revealing individual nerve cells (Glickstein, 2006). These nerve cells communicate with one another through junctions known as synapses. Sir Charles Sherrington and Edgar Adrian further described the nature of the synapse and the transmission of impulses between neurons (Swazey, 1968). Through these synapses, signals propagate from one neuron to another, resulting in the "all-or-none" firing of the neurons (McCulloch and Pitts, 1943). This process gives rise to thoughts, memories, and all cognitive processes. Information processing in the brain is a result of the collective activity of these intricate networks.

2.2.18.3 Plasticity

Eric Kandel demonstrated that learning and memory involve changes at the synaptic level. This is the concept of plasticity, meaning the structure and function of the brain can change in response to experience, learning, or injury (Hawkins et al., 1993). Synaptic plasticity, changes in the strength of connections between neurons, is considered a key mechanism underlying learning and memory (Abbott and Nelson, 2000). The brain's ability to change and adapt in response to experiences, has been a key source of inspiration for AI and machine learning algorithms (Lillicrap et al., 2020).

2.2.18.4 Hebbian Learning

One of the simplest of these algorithms is Hebbian learning. As mentioned above, one aspect of the learning paradigm explained in the heating example is the classical conditioning where the learner observes a correlation between the postman's arrival and the subsequent error. Underlying this behavioural phenomena, there are changes happening in the brain at the level of neural circuits. It is here that Hebbian learning can play a role. The ability of synapses to strengthen or weaken over time based on activity is termed synaptic plasticity. Donald Hebb is often credited for this concept which is captured by the famous phrase "cells that fire together, wire together", this is known as Hebb's rule or Hebbian theory (Hebb, 1949a). It describes the homosynaptic learning rules where changes in synaptic strength occur at synapses that have recently experienced activity. This is in contrast to heterosynaptic learning rules which involve changes in synapses which were not directly

activated by recent presynaptic or postsynaptic activities (Kulvicius et al., 2010). This helps distinguish between correlation-based learning and value-based learning, evaluative and non-evaluative learning (Wörgötter and Porr, 2005).

While Hebbian learning can provide a mechanism for synaptic changes in associative learning, classical conditioning at the behavioural level involves more complex processes, and the entirety of classical conditioning cannot be reduced solely to Hebbian learning. Furthermore, in this form of associative plasticity, the synaptic wiring can grow indefinitely and become extremely strong. This is because, as a connection strengthens, it is more likely to be further strengthened in the future due to its increased influence on post-synaptic firing (Porr and Wörgötter, 2006; Zenke et al., 2017). To counteract this, various normalisation mechanisms, such as synaptic scaling, are believed to be in place in the brain to ensure that synaptic strengths remain within reasonable limits and maintain network stability (Turrigiano, 2008). An extension of Hebbian learning, Oja's rule introduces a normalisation factor to the learning process, ensuring that the weights of the connections do not grow indefinitely (Oja, 1997).

Spike-timing-dependent plasticity (STDP) is a more refined version of Hebbian learning (Markram et al., 2011). Here, the change in synaptic weight is determined not only by simultaneous activations but also by the precise timing of spikes. If a post-synaptic neuron fires immediately after a pre-synaptic neuron, the synapse is strengthened; if the order is reversed, the synapse is weakened (Saudargiene et al., 2005).

2.2.18.5 Neurotransmitters

Beside simultaneous activations of neurons and the timing of their spikes, neurotransmitters also influence synaptic plasticity. These are fundamental chemical messengers that facilitate the transfer of information within the nervous system. They are released in synapses from one neuron and are received by receptors on another. Various chemical substances, including dopamine, serotonin, glutamate, and gamma-aminobutyric acid (GABA), act as neurotransmitters (Webster, 2001). Their pivotal roles in signalling and modulating brain activity have a profound influence on behaviour. Drawing inspiration from the intricate workings of neurotransmitters, particularly the influence of dopamine on synaptic plasticity in the brain, new algorithms have emerged. These algorithms utilise a modulating signal to determine the adaptability of neuron's weights. This modulating signal, much like the reward signals in our brains, serves as an essential feedback mechanism.

2.2.19 Model-Based Learning

Up to this point, we emphasised our objective of introducing a learning module to supplement an existing reflex mechanism. We have explored the learning operation from both psychological and neuroscientific perspectives. We have explained how RL and optimal control provide a framework for this learning. Although we have frequently referenced to the learner and the learning process, a vital question remains: what exactly is the learner learning? An internal model.

In the context of the heating system analogy, the system operates with foresight. By leveraging this anticipatory awareness, it proactively adjusts its actions to ensure its primary objective — consistently maintaining an error-free equilibrium. The human brain's ability to predict and simulate the world around it has been a subject of fascination and study across various disciplines. Central to this idea is the concept of 'internal models' (Imamizu et al., 2000). These are representations or simulations of the external environment that the brain uses to guide perception, cognition, and action (Imamizu and Kawato, 2009). The history and understanding of these models span several fields, from neuroscience to control theory and artificial intelligence.

2.2.19.1 Internal Models

The seeds of the internal models concept can be traced back to the 19th century when Hermann von Helmholtz proposed that the brain uses internal models to interpret sensory data, effectively acting as an inference machine. Helmholtz suggested that our perception of the world is a result of the brain's best guess based on these internal representations (von Helmholtz, 1925).

2.2.19.2 Internal Models in Neuroscience & Psychology

As our understanding of the brain deepened in the 20th century, the idea of internal models evolved. One of the fields that is primarily concerned with this phenomena is motor control. The field of motor control learning is concerned with how reflex and learnt movements are controlled, specifically how the CNS is organised to coordinate and allow for such movements (Schmidt et al., 2018).

Historically, the desired movement trajectory has been a primary focal point for many researchers. However, Todorov argued for a broader view by suggesting that sensory feedback should be integrated into our understanding. Instead of solely focusing on the trajectory, taking sensory feedback into account can offer a more comprehensive understanding of how movements are controlled (Todorov, 2004).

Later, researchers started to think of movement control as an interactive communication between the organism and the environment, in the form of feedback loops, similar to

control theory. The combining of the control and communication theory was later called Cybernetics by Wiener (Wiener and von Neumann, 1949). Later, in the late 20th century, researchers looked deeper into how different regions of the brain play a role in movement control.

The cerebellum, responsible for motor control, as well as Basal Ganglia which is involved in movement regulation and reward-based learning, became focal points for this discussion (Larkum, 2013). Mitsuo Kawato, in the 1990s, built on earlier theories to propose that the cerebellum operates as an adaptive controller, refining internal models of the body and the environment (Tsukahara and Kawato, 1982). These models allow for smoother, more coordinated movements. Kawato introduced the feedback error learning (FEL) technique when studying the cerebellum's involvement in motor control (Tsukahara and Kawato, 1982). Later he applied this technique to adaptive non-linear feedback control (Kawato et al., 1987) which became a widely used technique.

MacKinnon in 2007 proposed that activities as routine as walking are not just reactionary or reflexive. Instead, they can be seen as anticipatory adjustments, suggesting that the brain always predicts the next state (MacKinnon et al., 2007).

In 2011, Wolpert formulated several principles for motor control (Wolpert et al., 2011). One significant point he highlighted was that our brain encodes what we have learned about movements, influencing our reactions when errors occur. These cognitive representations can be understood as either intricate models or broader concepts about the expected functioning of movements. In the context of motor control and planning, two main types of internal models are often discussed: forward models and inverse models. Both play crucial roles, but they serve distinct functions.

2.2.19.3 Forward Models

A forward model predicts the next state of a system given a current state and an action, meaning the sensory consequences of a motor command (Popa and Ebner, 2019). For instance, if you decide to move your arm, your brain's forward model can predict the expected sensory feedback such as the feeling of the arm moving, or the visual input of seeing the arm move. One of the benefits of forward models is error correction. By comparing the actual sensory feedback to the predicted feedback from the forward model, the brain can correct for any discrepancies.

A compelling rationale for the necessity of an internal model is the inherent delay between issuing a motor command and receiving sensory feedback in real-time motor control. To navigate this challenge, organisms are compelled to construct adaptive internal models, encompassing both their own bodies and the external world. The forward model provides immediate predictions, while the inverse model can adjust future actions based on these predictions and actual feedback. Forward models are only useful if they produce

unbiased predictions. Evidence shows that forward models remain calibrated through motor adaptation which is a type of learning driven by sensory prediction errors (Shadmehr et al., 2010).

2.2.19.4 Inverse Models

An inverse model, on the other hand, determines the necessary action to achieve a desired outcome. If you know where your hand is and you want to touch your nose, the inverse model figures out the motor command required to get your hand from its current position to your nose. This is beneficial for goal-directed actions. It allows the brain to determine the necessary movements to achieve a specific sensory outcome or goal. Having an inverse model allows for a more direct route to achieving a desired outcome.

2.2.19.5 Interplay of Forward & Inverse Models

The interplay between forward and inverse models is crucial for effective motor control and learning. The synergy of the two allows for learning through prediction errors. When we learn a new motor skill, our forward model's predictions may not match the actual outcomes. These prediction errors can be used to refine the inverse model, making it more accurate over time.

Before executing a movement, the brain can use the forward model to simulate various actions and predict their outcomes. The results of these simulations can then inform the inverse model to select the best action.

In 1998, Kawato and Wolpert presented the modular selection and identification for control (MOSAIC) model, a modular paradigm for motor control (Wolpert and Kawato, 1998). Within this framework, forward models influence the contribution of each inverse model's output, integrating them into the ultimate motor command. Subsequently, they showcased this concept in the context of grasping various objects (Haruno et al., 2001)

Fast forward to 2017, Maffei introduced a novel perspective. Contrary to the prevalent belief that motor errors predominantly drive the prediction or learning for anticipatory control, he proposed that the brain or human behaviour learns more from future sensory errors. In other words, it is not just the errors in our actions that inform our future movements but also the errors in what we sense and perceive (Maffei et al., 2017). To test this theory, he simulated and compared both approaches and found that adaptation rooted in the sensory domain results in control that robustly mirrors that of biological systems. The intricacies of this anticipatory motor control were further highlighted using the example of pole balancing with hierarchical sensory predictive control (HSPC) (Maffei et al., 2017).

Later, Kawato and his team implemented a hierarchical structure of the neural model to enhance trajectory control in an industrial robot manipulator (Miyamoto et al., 1988).

Notably, the neural network model demonstrated the capability to generalise the movements it had learned. This approach presents a significant advantage over traditional control methods where the error is essentially the control output of the feedback controller. This controller calculates a forward model (known as the inverse dynamics) through heterosynaptic plasticity, utilising a preceding signal, such as the impact or a cue (Miyamoto et al., 1988). Kawato suggested in 1999 that both kinetic and dynamic internal models are keys to motor control (Kawato, 1999).

2.2.19.6 Internal Dynamics in Control Theory

In control theory, systems are optimised based on feedback (Harbor and Phillips, 2000). The concept of an "adaptive controller", as suggested by Kawato in relation to the cerebellum, can be seen as a biological counterpart to engineered control systems. The forward and inverse dynamics serve complementary roles in many control systems. While the forward dynamics predict the consequences of actions, the inverse dynamics determine the actions required to achieve desired consequences. In advanced motor control systems, like those found in some robots or humans, the predictions of the forward dynamics can be used to refine or adjust the control commands determined by the inverse dynamics. This iterative feedback process can improve accuracy and adaptiveness.

2.2.19.7 Internal Models in Reinforcement Learning (RL)

The concept of internal models is pivotal in the field of RL, categorising agents into two distinct learning approaches: model-based learning and model-free learning (Sutton and Barto, 2018). Agents utilising model-based methods maintain an internal model of the environment, adapting and refining their actions based on received reward signals to optimise outcomes.

Possessing an internal model of the environment allows the application of planning methods such as dynamic programming to deduce an optimal policy. The model's predictive capabilities regarding future states and rewards enable the simulation of potential sequences of actions and states, aiding in identifying the most advantageous actions to pursue. Techniques like Value Iteration or Policy Iteration employ the model explicitly to derive the optimal policy (Bellman, 1957).

Furthermore, the internal model can be harnessed to compute or update value functions. In dynamic programming methods, for instance, the Bellman equation is utilised in conjunction with the model to update value estimates. Knowing the behaviour of the environment through the model and possessing an estimate of the value of future states through the value function allows for the combination of these elements to estimate the value of current states.

2.2.19.8 Internal Models in Cybernetics

Cybernetics, the study of systems, communication, and control in animals and machines, directly taps into the idea of internal models. Pioneers like Norbert Wiener and Ross Ashby hypothesised that both living organisms and machines utilise feedback mechanisms to adapt and reach goals, mirroring the brain's use of internal models to predict and adjust to its environment.

2.2.20 Practical Implementations & Case Studies

Thus far, we have examined both the platform and the learner from broad psychological and intricate neurophysiological perspectives. Now, we shift our attention to instances where such a learner is integrated into this platform for continuous online learning.

In the early 2000s, Bernd Porr and colleagues pioneered one of the initial frameworks illustrating how the agent's adaptive behaviour can be structured within a closed-loop platform for continuous learning. Their work highlighted the process of learning a reflex's forward model. By enhancing the reflex and the ideal set point with the agent's predictive capabilities, a cohesive closed-loop learning environment was established (Porr and Wörgötter, 2002a). Utilising a linear learning algorithm, they demonstrated the system's ability to apply feed-forward control, and analytically proved that slower feedback loops could effectively be substituted by their corresponding feed-forward controllers, thereby crafting a forward model.

Following this work, they illustrated that this platform could harness temporal-sequence learning to produce proactive anticipatory actions. This was realised by leveraging a second sensor event that occurred earlier and was causally coupled. This principle evolved into the development of the ISO learning algorithm, which is geared towards unsupervised temporal sequence learning (Porr and Wörgötter, 2003). The algorithm's efficacy was assessed in both open- and closed-loop behavioural feedback scenarios. Their experiments showcased that a robot, by recognising the correlation between its preliminary range-finder signals and subsequent collision signals, could adeptly avoid collisions.

Later, they demonstrated that this platform could leverage error feedback from the reflex and associate it with contextual environmental input, facilitating predictive learning. This was termed ICO learning, which shares similarities with Hebbian learning. However, the exclusive use of input correlations to the neurons yielded superior stability compared to traditional Hebbian methods (Porr and Wörgötter, 2006). In this learning paradigm, the error signal combines with the weighted neuron activations to produce a command for both the reflex and learning processes. A notable benefit of this approach is the behavioural relevance of the error signal. However, the direct combination of the error and activations introduces information loss, which limits the platform, making it challenging to adapt to

more intricate structures.

Their future works showed how a network also employing sensor predictions was able to improve the steering actions of a car where a non-optimal hard-wired steering is then quickly superseded by a forward model, based on camera information of the road ahead (Kulvicius et al., 2007). This group of learning paradigms shows that it is possible to achieve one-shot learning because at every step the error signal from the reflex, meaning the proportional integral derivative (PID) controller, is available for learning. This platform's limitation to a single layer is evident in its dedicated chained architecture, which restricts network topology design due to the error signal merging with activation.

Subsequent research by the same group demonstrated the effectiveness of incorporating sensor predictions in refining the steering actions of a car. In their experiments, an initially non-optimal hardwired steering mechanism was rapidly overtaken by a forward model using camera-based data of the upcoming road (Kulvicius et al., 2007). Such learning paradigms suggest the potential for one-shot learning, as the error signal from the reflex, provided by the PID controller, is consistently available for learning. Similar to ICO learning, this platform's restriction to a single layer results in a specialised chained architecture, showing that the network's design is constrained by directly merging the error signal and activation.

2.2.21 Neural Networks

As highlighted earlier, the foundational mechanisms of the learners in these studies draw inspiration from the functioning of neurons and synapses in the brain. For instance, ICO learning exhibits a marked enhancement in stability compared to traditional Hebbian learning. However, the brain utilises multiple layers of neurons to achieve its remarkable learning capabilities. To harness this potent attribute for machine learning, it is imperative to construct more intricate architectures with deeper layers of artificial neurons. Artificial neural networks (ANNs) serve as computational frameworks inspired by the intricate workings of biological neural processes (Bishop, 1995). Just as the human brain comprises interconnected neurons, neural networks (NNs) consist of layers of nodes, or artificial neurons, designed to simulate these biological counterparts. Much like synapses in the brain, connection weights within NNs play a pivotal role in determining how nodes communicate and influence each other. These connection weights, analogous to synaptic strengths, are fine-tuned through a learning process.

In the realm of NNs, each node processes its input and makes activation decisions through the application of non-linear functions, closely resembling the firing behaviour of biological neurons. This non-linearity allows NNs to capture complex relationships within data, enabling them to model intricate patterns and make sophisticated decisions.

Moreover, ANNs, particularly DNNs, demonstrate a hierarchical approach to handling

information, mirroring how the human brain processes sensory data in stages. This hierarchical feature allows DNNs to extract increasingly abstract and meaningful features as data progresses through the network layers.

2.2.22 Deep Learning

Here we employ a DNN for the learner. Deep learning is a subfield of machine learning that focuses on algorithms inspired by the structure and function of the brain, such as ANNs. This involves training multi-layered neural networks (Goodfellow et al., 2016). Deep-learning (DL) was introduced by researchers such as Warren McCulloch and Walter Pitts in the 1940s (McCulloch and Pitts, 1943). Later, Frank Rosenblatt introduced the Perceptron, one of the earliest neural network models (Rosenblatt, 1958). The idea of using DNN existed at this time, but the challenge was finding a feasible way to train these networks. In the early 1970s, Seppo Linnainmaa wrote about the "reverse mode of automatic differentiation", which is essentially the basis for back-propagation (BP), but it was not applied to neural networks at that time (Schmidhuber, 2014). Later in the 1980s, Werbos introduced the BP algorithm in his Harvard Ph.D. thesis, "Beyond Regression" (Werbos, 1974). Later in 1986, in a landmark paper titled "Learning representations by back-propagating errors", David Rumelhart, Geoffrey Hinton, and Ronald Williams popularised the BP algorithm in the context of neural networks (Rumelhart et al., 1986). Their work brought significant attention to the algorithm in the machine learning and cognitive science communities. However, the term "deep-learning" started gaining traction in the 2010s. A breakthrough came when a DNN called "AlexNet" achieved a record-breaking performance on the ImageNet Large Scale Visual Recognition Challenge, a prestigious image classification competition (Deng et al., 2009).

The multi-layered architecture of DNNs enables them to learn a hierarchical representation of data. In tasks like image recognition, for instance, the initial layers might capture basic features like edges and textures, while deeper layers synthesise these into more complex patterns like shapes and objects (Bishop, 1995). This makes it a suitable choice for the robotic application in this work, where the camera view is fed into a DNN to generate the predictive motor actions.

2.2.23 The Synergy of Deep Learning & Reinforcement Learning

One of the key advancements in Q-learning has been the use of DNNs to approximate the Q-function in high-dimensional state spaces (Guo et al., 2014; Gu et al., 2016). This has led to the development of deep Q-networks (DQNs), which have been shown to outperform traditional Q-learning on a range of tasks (Mnih et al., 2015). The success of RL is owed to the synergy between DL and Q-learning, which has enabled RL agents to learn

complex policies from high-dimensional sensory inputs (Sewak, 2019). DL has provided a powerful framework for function approximation, feature learning, and representation learning, which are essential for dealing with high-dimensional states and complex actions in RL problems. DNN can learn to represent the state and action spaces in a compact and efficient way, enabling RL agents to make decisions based on complex and abstract information (Goodfellow et al., 2016; Gu et al., 2016).

2.2.24 Back-Propagation (BP)

BP works by computing the gradient of a loss function with respect to each weight in the network by applying the chain rule of calculus (Chauvin and Rumelhart, 2013). After feeding input data forward through the network to obtain a prediction, the algorithm calculates the difference between the prediction and the true target; this is the open-loop error. This error is then propagated backward through the network, adjusting the weights to minimise the error. The process is iteratively repeated using many input-output pairs until the network's weights converge to values that minimise the prediction error. BP is inherently an open-loop (supervised) learning algorithm used for training multi-layer neural networks. Section 2.6.3 shows how back-propagation is utilised in this work for closed-loop (reinforcement learning) applications.

2.2.25 Closed-Loop BP

Previously we detailed the integration of the learner within the platform, resulting in a closed-loop interactive system between the agent and its environment. We also highlighted our decision to employ DL as the foundation for this learner. The current challenge is to align the input-output relationships of the learner within the platform to those of the DNN.

The DNN primarily processes camera inputs, which provide a visual representation of the environment, specifically the road ahead. This is consistent with the platform's design, where the learner also uses the camera feed as its primary input. However, a significant difference lies in the handling of errors. While the DNN is designed to handle output errors, the learner is not configured to do so. Moreover, in a closed-loop context, the error resulting from the agent's actions or outputs is not directly discernible. As mentioned, such errors are typically identifiable only by an external observer or the supervisor in supervised learning (Porr and Wörgötter, 2005).

In this study, we demonstrate how DL with BP remains a viable candidate for this application. We argue that minimising the closed-loop error is analogous to minimising the open-loop output error, refer to Section 2.6.2. Our claims are supported through mathematical derivations, programming, and a combination of simulated and real-world

experiments.

2.2.26 Evaluation Criteria & Research Gap Analysis

In this section, we identify criteria essential for evaluating learning algorithms in terms of their application and usability. We will present a table summarising these points for the algorithms discussed above. This table will highlight the current research gaps and illustrate how the novel algorithm introduced in this work addresses and potentially closes these gaps. Through this comprehensive evaluation, we aim to demonstrate the superiority and innovation of our proposed algorithm in the context of these key criteria.

	Online / Real-Time	Continuous Space	Closed-Loop System	Allows Deep Network	No Training Data	Suitable for Robotics	Biologically Realistic	No Properties of Env.	Inherently Stable	Model-Based	One-Shot Learning
CLDL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Continuous RL	-	✓	✓	x	x	✓	x	✓	-	-	x
Deep Learning	-	-	x	✓	x	-	x	✓	-	-	x
Deep Q-Nets	-	x	✓	✓	x	✓	x	✓	-	-	x
Hebbian Learning	x	-	✓	✓	x	-	✓	✓	x	✓	x
ICO Learning	✓	✓	✓	x	✓	✓	✓	✓	✓	✓	✓
ISO Learning	✓	✓	✓	x	✓	✓	✓	✓	✓	✓	✓
Optimal Control	✓	✓	✓	x	x	✓	x	x	✓	x	x
PID Controller	✓	✓	✓	x	x	✓	x	x	✓	x	x
Q-Learning	-	x	✓	x	x	✓	x	✓	-	-	x
RL	-	x	✓	x	x	✓	x	✓	-	-	x
SARSA	-	x	✓	x	x	✓	x	✓	-	-	x
TD Learning	-	x	✓	x	x	✓	x	✓	-	-	x
Temporal Sequence	✓	✓	✓	✓	-	✓	-	-	-	-	-

Table 2.1: *Learning & Control Algorithms Capability Matrix*

2.3 Neural Networks: Theoretical Concepts & Fundamentals

The architecture of artificial neural networks is designed to mimic the interconnections found in the human brain. A neural network comprises numerous individual neurons that are organised in layers and interconnected in a structured arrangement.

2.3.1 Structure of Neural Networks (NNs)

Figure 2.2 illustrates an example of a neural network architecture. The network receives a set of inputs, which are initially directed into the first layer of neurons. These neurons perform computations on the inputs and transmit their results to the subsequent layer of neurons. This iterative process continues through the network's various layers until it reaches the final layer, which generates the network's final output.

In the network, each neuron receives input from multiple other neurons and uses these inputs to calculate its own output. Subsequently, the output of each neuron is transmitted to other neurons within the next layer. This process is iteratively carried out for each layer of the network until the final output is generated.

The configuration and connectivity of neurons within a neural network can be flexibly modified to accomplish various objectives, including tasks such as classification, regression, or pattern recognition. Learning involves fine-tuning the strength of connections between neurons. This adjustment process enables the network to learn and generate the desired output when provided with a specific set of inputs.

2.3.2 Indexing of Layers & Neurons

In a neural network, neurons are structured into layers, where each layer comprises a group of neurons responsible for processing input data in a specific manner. To facilitate precise indexing of neurons within the network, we define the total number of layers as L. These layers are indexed sequentially from the input to the output layer using the set of integers $\{0, 1, \ldots, \ell-1, \ell, \ell+1, \ldots, L-1, L\}$. The total number of neurons within each layer is denoted by capital letters like I, J, or K, and individual neurons are assigned indices from the set of integers $\{0, \ldots, i, \ldots, I\}$, $\{0, \ldots, j, \ldots, J\}$, and $\{0, \ldots, k, \ldots, K\}$, respectively. Each neuron is uniquely identified using the notation $\textcircled{n}_j^{\ell}$, where the subscript j corresponds to the neuron's index, and the superscript ℓ represents the layer index in which the neuron resides.

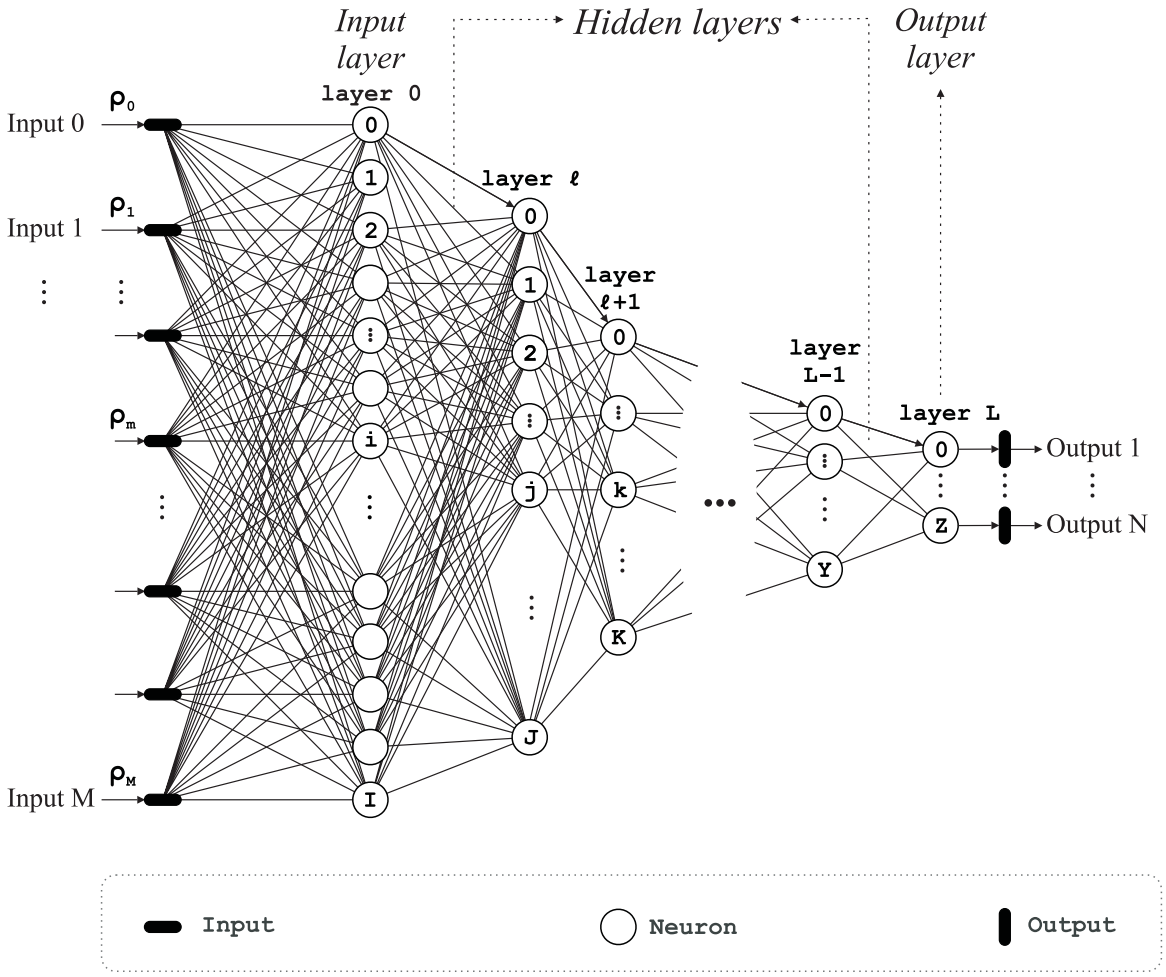

Figure 2.2: *Depicts the architecture of an artificial neural network. The input layer (layer 0) receives user inputs ρ, which are processed through the hidden layers (layers ℓ to $L-1$), culminating in the generation of outputs at the output layer. Circles represent individual neurons, vertical groupings indicate the distinct layers, and solid lines illustrate the interconnections between neurons across layers.*

2.3.3 Neuron Connections

Neural networks can be categorised based on the connectivity patterns among their neurons.

2.3.3.1 Matrix of Connectivity C_ℓ

The inter-neuron connections between adjacent layers can be conveniently represented using a matrix denoted as C_ℓ, with ℓ signifying the layer index. Within this matrix, the entry at position (j, k), denoted as $C_\ell(j, k)$, indicates whether neuron j in layer ℓ is connected to neuron k in layer $(\ell + 1)$. Specifically, if neuron j in layer ℓ is indeed connected to neuron k in layer $(\ell + 1)$, then the value of $C_\ell(j, k)$ is set to 1, otherwise, it is set to 0.

$$C_\ell = \begin{array}{c} \\ \ell\downarrow \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} (\ell{+}1)\rightarrow & & & & \\ 0 & ... & k & ... & K \end{array} \\ \begin{bmatrix} 1 & \dots & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & \dots & 0 & \dots & 1 \end{bmatrix}_{JK} \end{array} \begin{array}{c} \\ 0 \\ ... \\ j \\ ... \\ J \end{array}$$

A neural network is categorised as fully connected when all entries in the connectivity matrix C_ℓ are set to 1 for each layer $\ell = 0, \dots, L-1$. This implies that every neuron in each layer is connected to every neuron in the adjacent deeper layer.

Conversely, if the connectivity matrix C_ℓ contains non-unity entries, the network is classified as a convolutional neural network (CNN). Such networks are commonly employed in image processing tasks where the input data exhibits a grid-like structure, such as pixels in an image. In CNN, neurons in one layer are exclusively connected to a localised region of neurons in the adjacent layer, rather than establishing connections with all neurons in that layer. This approach enables the network to efficiently process substantial volumes of data while reducing the number of parameters requiring learning.

2.3.3.2 Matrix of Recursivity R_ℓ

The matrix of recursivity, denoted as R_ℓ, characterises the connections of neurons in layer ℓ to their own inputs. If a neuron's output is fed back into its own input, the corresponding entry in R_ℓ is assigned a value of 1; otherwise, it is set to 0. When all entries in R_ℓ are 0 for every layer ℓ, the network is referred to as "feed-forward". In such networks, the output of neurons in one layer does not loop back into their own input or any previous layer's input.

Conversely, if one or more entries in R_ℓ are non-zero for any layer ℓ, the network is labelled as "recurrent". This indicates that the output of neurons can feed back into their own input or the input of previous layers, thereby creating a loop or feedback mechanism within the network.

$$R_\ell = \begin{array}{c} \\ \ell\downarrow \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \\ \begin{bmatrix} 0 \\ \dots \\ 1 \\ \dots \\ 0 \end{bmatrix}_{J1} \end{array} \begin{array}{c} \\ 0 \\ ... \\ j \\ ... \\ J \end{array}$$

For all algorithms in this work, a fully connected feed-forward neural network is employed. This signifies that all neurons within a layer establish connections with all neurons

in the preceding layer, and there are no connections that loop back into the same layer or any previous layers. In other words, the matrix of connectivity C_ℓ consists of all 1's, while the matrix of recursivity R_ℓ comprises entirely of 0's for every layer ℓ.

2.3.4 Inside Neurons: The Building Blocks of NNs

Having provided an overview of the network and detailed the connections among neurons, the following delves deeper into the inner workings of individual neurons. Neurons serve as the fundamental building blocks of neural networks, mirroring the function of biological nerve cells in the brain.

In the context of ANN, neurons play a role inspired by their biological counterparts. In machine learning, neurons can be thought of as mathematical functions that perform two main operations: they calculate a weighted sum of inputs and then apply a non-linear function to the result. Each neuron, as a mathematical function, produces a unique output for a given set of inputs. Figure 2.3 provides an illustration of a typical neuron, denoted as $\textcircled{n}_j^\ell$.

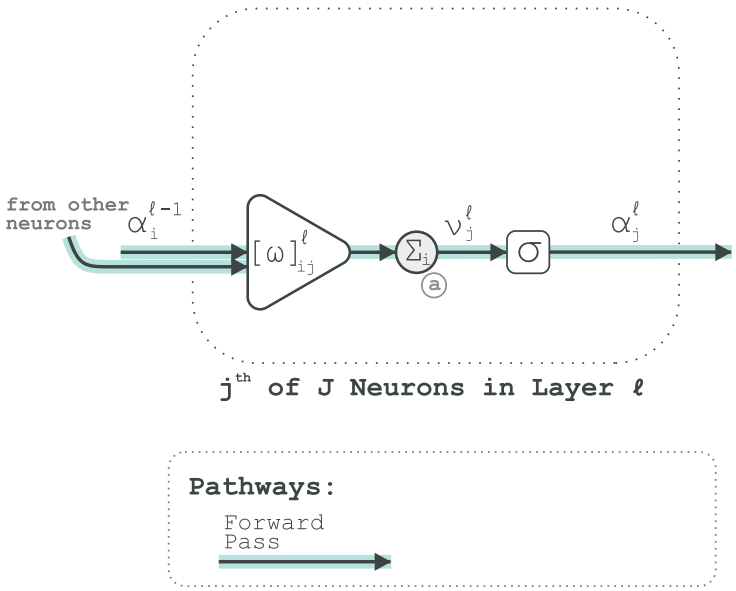

Figure 2.3: *Illustrates the forward pass of signals within the j^{th} neuron in layer ℓ, progressing from input values $\alpha_i^{\ell-1}$, to their weighted summation at node ⓐ, through the activation function σ, and ultimately producing the output α_j^ℓ.*

2.3.4.1 The Output of the Neuron

The term used to describe the output of neurons is "activation". This terminology draws inspiration from biology, where neurons are often considered either active, resulting in an output of 1, or inactive, yielding an output of 0. However, in many machine learning

applications, artificial neurons generate a continuous value within the range of 0 to 1. These activations are represented by α and are indexed as α_j^ℓ for the $\textcircled{n}_j^\ell$ neuron.

2.3.4.2 The Inputs to the Neuron

The inputs to each neuron are derived from the activations of neurons in the preceding layer, with the exception of the first layer where the inputs are initially supplied by the user or the environment. As depicted in Figure 2.3, for the neuron $\textcircled{n}_j^\ell$, the inputs are specifically represented as $\alpha_{i:0\rightarrow I}^{\ell-1}$.

2.3.4.3 Neuron's Function

As previously mentioned, neurons perform a computation involving a weighted sum of their inputs, which is then passed through a non-linear function. Each input to a neuron is associated with a distinct weight, representing the strength of the connections between neurons and the extent of their influence on one another. These weights are represented by ω. Specifically, the weight that signifies the connection strength between neurons $\textcircled{n}_i^{\ell-1}$ and $\textcircled{n}_j^\ell$ is denoted as ω_{ij}^ℓ.

The result of this weighted sum of inputs is referred to as the accumulation, represented by the symbol ν, and it is indexed as ν_j^ℓ for neurons $\textcircled{n}_j^\ell$. This operation is depicted in Figure 2.3 at node $\textcircled{a}$, and it is formulated as follows:

$$\nu_j^\ell = \Sigma_{i=0}^{I}(\omega_{ij}^\ell \cdot \alpha_i^{\ell-1}) \quad + \quad b_j^\ell \tag{2.1}$$

Here, b represents the bias of the neuron. Nevertheless, it is worth noting that in this work, biases are deliberately set to zero to ensure the production of DC-free outputs.

2.3.4.4 Neuron's Non-Linearity

The accumulation ν undergoes a transformation through a non-linear function to generate the neuron activations. This function is commonly referred to as the activation function of neurons. In most cases, an activation function maps an input space ranging from $-\infty$ to $+\infty$ to an output space of $(0, 1)$. Examples of such functions include the hyperbolic tangent, arctangent, and the logistic function.

In this work, the conventional logistic function, often referred to as the sigmoid function σ, is employed. Please refer to Figure 2.4 for visualisation. For the purposes of this work, a vertical translation of -0.5 is applied to achieve a symmetrical output space spanning from -0.5 to 0.5, encompassing both positive and negative values:

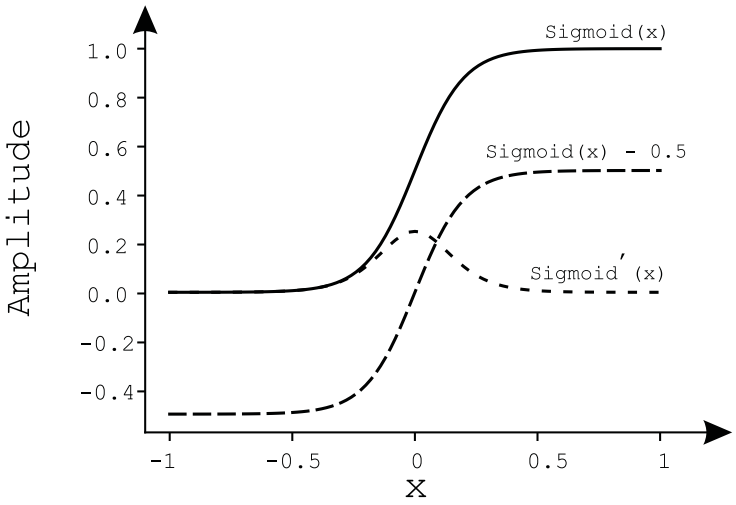

Figure 2.4: *The solid line represents the conventional logistic function. The long-dashed line illustrates the activation function employed in this study, a logistic function shifted vertically by -0.5 units. Notably, both functions produce identical derivatives depicted by the short dashed line. y-axis: Amplitude of the neuron's output. x-axis: Range of accumulated input values for the activation function, spanning from negative to positive infinity.*

$$\sigma(x) = \frac{1}{1 + e^{-x}} - 0.5 \tag{2.2}$$

With these considerations, the calculation of neuron activations is as follows:

$$\alpha_j^{\ell} = \sigma(\nu_j^{\ell}) = \sigma(\Sigma_{i=0}^{I}(\omega_{ij}^{\ell} \cdot \alpha_i^{\ell-1})) \tag{2.3}$$

2.3.5 Forward Propagation of Inputs

Forward propagation entails the iterative forward movement of signals through the network. This process commences with the reception of inputs ρ_m at the first layer, followed by the propagation of activations through the hidden layers. Finally, it concludes with the return of outputs to the user at the final layer. These three stages are summarised as follows:

$$\alpha_n^0 = \sigma(\nu_n^0) = \sigma(\Sigma_{m=0}^{M}(\omega_{mn}^0 \cdot \rho_m)) \quad \text{input layer} \tag{2.4}$$

$$\dots$$

$$\alpha_j^\ell = \sigma(\nu_j^\ell) = \sigma(\Sigma_{i=0}^{I}(\omega_{ij}^\ell \cdot \alpha_i^{\ell-1})) \quad \text{hidden layer} \tag{2.5}$$

$$\alpha_k^\ell = \sigma(\nu_k^\ell) = \sigma(\Sigma_{j=0}^{J}(\omega_{jk}^\ell \cdot \alpha_j^{\ell-1})) \quad \text{hidden layer} \tag{2.6}$$

$$\dots$$

$$\alpha_z^L = \sigma(\nu_z^L) = \sigma(\Sigma_{y=0}^{Y}(\omega_{yz}^\ell \cdot \alpha_y^{L-1})) \quad \text{output layer} \tag{2.7}$$

The forward propagation through the hidden layers involves a recursive process, which is visually depicted in Figure 2.5 for two arbitrary neurons, namely $(n)_j^\ell$ and $(n)_k^{\ell+1}$.

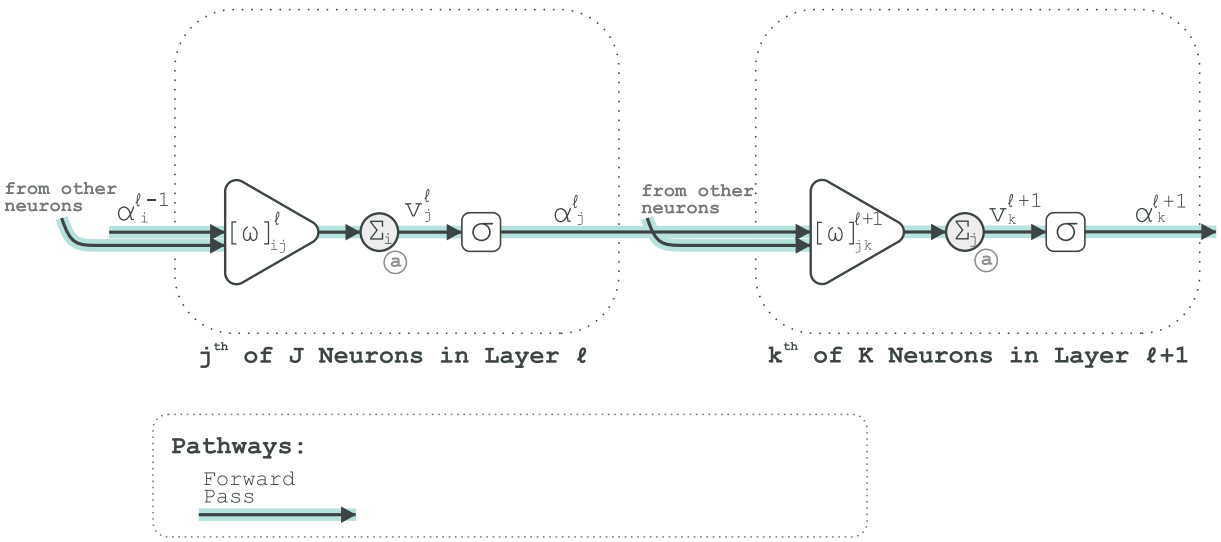

Figure 2.5: *Illustrates the interconnections between neurons and the forward propagation of activations. The activation from the j^{th} neuron in layer ℓ, together with that of other neurons in this layer, are channelled to the k^{th} neuron in layer $\ell + 1$.*

2.3.6 Contextual Application of Neural Networks

Neural networks can be categorised based on how their output is utilised within the context of their application and use-case, primarily contingent on the manner in which their output is evaluated.

2.3.6.1 Open-Loop Context: Unsupervised Learning

Unsupervised learning, a fundamental neural network context, primarily revolves around data clustering and feature extraction. In this paradigm, the network's output does not undergo evaluation for correctness or error. Instead, the network's task is to sort input

data into clusters based on inherent patterns and similarities. This context finds use in applications where the primary objective is to reveal hidden structures within data, such as image segmentation, anomaly detection, and recommendation systems.

2.3.6.2 Open-Loop Context: Supervised Learning

Supervised learning stands in stark contrast to unsupervised learning, as it involves the evaluation of network outputs. In an open-loop context, the network operates on labelled data, where each input is associated with a known target or label. The network's primary purpose here is to learn a mapping from inputs to outputs that minimises the prediction error.

Crucially, the evaluation of output does not influence subsequent inputs, making it a classic example of open-loop learning. Applications abound in fields like image classification, natural language processing, and regression tasks, where accuracy is paramount.

2.3.6.3 Closed-Loop Context: Reinforcement Learning

The most intriguing and dynamic application of neural networks is found in RL, where the output evaluation has a direct impact on the next set of inputs, hence the learning is closed-loop. RL is a form of minimal supervision, and it is central to many cutting-edge AI applications. In RL, an agent interacts with an environment and learns to make sequences of decisions to maximise a cumulative reward signal. This process is iterative and involves a continuous feedback loop, where the network's actions directly affect future observations and rewards. As mentioned in the introduction, this is the type of learning employed in this work.

2.3.6.4 Training vs. Performing Phase

In conventional RL, an agent typically goes through two phases: training and performing. During training, the agent explores its environment, learns a policy, and adjusts its neural network weights to optimise its decision-making process. Once the training is complete, the agent transitions to the performing phase, where it utilises its learned policy to interact with the environment. This dichotomy is crucial for ensuring safe and effective RL.

2.3.6.5 Online vs. Offline Learning

In this work, all presented algorithms operate *online*, meaning there is no training phase to the learning; the agent starts by performing, and this performance improves over time. This type of learning in RL involves continuous adaptation of the agent's policy based on real-time feedback, making it suitable for dynamic and evolving environments. In

contrast, offline learning involves training the agent on a fixed dataset, which may not adapt to changes in the environment.

2.3.6.6 Discrete vs. Continuous Actions

Agents in RL can make discrete or continuous actions. Discrete actions involve choosing from a predefined set of actions such as moving left or right, while continuous actions allow for a broader range of choices such as continuous motion control. The choice between these action spaces depends on the nature of

the task and the environment. In this work, a robot learns to navigate its environment in a continuous manner.

2.3.7 Mathematical Derivations of Open-Loop Deep Learning

Equation 2.4 shows the computation of the outputs of a neural network based on user inputs and associated weights. In supervised learning, the target outputs are predefined, here represented as β_n^L. These target outputs serve as benchmarks to assess the performance of the network, enabling the quantification of the associated errors e_n^L as follows:

$$e_n^L = \beta_n^L - \alpha_n^L \tag{2.8}$$

2.3.7.1 Cost Function

The error above may manifest as either positive or negative, with the ultimate goal being its reduction to zero. The cost function serves as a metric quantifying the divergence from the desired output. It is most frequently characterised by mean squared error (MSE):

$$C = \frac{1}{2N}\Sigma_{n=0}^{N}(e_n^L)^2 \tag{2.9}$$

$$\tag{2.10}$$

The cost function is always positive and measures the deviation from the desired output, irrespective of the sign of the error. Given a continuous error signal, the cost function is differentiable which allows for the use of calculus optimisation techniques to minimise the cost function. This in turn ensures that the error signal is kept as close to zero as possible.

2.3.7.2 Gradient Descent Method (GDM)

Gradient descent is a widely-used optimisation algorithm in machine learning and deep learning, aimed at minimising the cost function above, which quantifies the disparity between a model's predictions and actual values. The objective is to identify the network's weights that will minimise this function and, in turn, enhance the model's accuracy. In the context of this research, the focus is on determining the optimal weight configuration that ensures the cost function is minimised across all outputs.

This method operates iteratively: the system commences with a set of arbitrary parameters arising from a random weight initialisation. Subsequently, the gradient of the cost function is computed, yielding a vector that indicates the direction of the most pronounced increase in the loss. The parameters are then adjusted by taking a step counter to this gradient direction.

Various versions of gradient descent exist, including stochastic gradient descent (SGD), mini-batch gradient descent, and advanced optimisation techniques like Adam and RMSprop Zou et al. (2019). These techniques modify the learning rate throughout training, aiming for quicker and steadier convergence. However, for the purposes of this research, we adhere to the conventional gradient descent method (GDM) described above.

2.3.7.3 Backpropagation: A Mathematical Approach

Backpropagation is the algorithm employed to determine the gradients of the cost function in relation to the network's weights. It provides insights into how alterations in the parameters influence the cost. This relationship underscores the sensitivity of the cost function with respect to a specific weight, which is expressed as:

$$\frac{\partial C}{\partial \omega_{ij}^{\ell}} \tag{2.11}$$

This signifies that an adjustment of $-\frac{\partial C}{\partial \omega_{ij}^{\ell}}$ to that specific weight will exert the most substantial influence on the minimisation of the cost function.

2.3.7.3.1 Chain Rule Our aim is to derive an expression that addresses weight changes across all deeper layers. To achieve this, we further elaborate on Equation 2.11 using the chain rule, with the accumulation of the neurons serving as the connecting term:

$$\frac{\partial C}{\partial \omega_{ij}^{\ell}} = \frac{\partial C}{\partial v_{j}^{\ell}} \frac{\partial v_{j}^{\ell}}{\partial \omega_{ij}^{\ell}} \tag{2.12}$$

The second derivative is the activation of the neuron linked to that particular weight, as elaborated in Equation 2.4:

$$\frac{\partial v_j^\ell}{\partial \omega_{ij}^\ell} = \alpha_j^{\ell-1} \tag{2.13}$$

To elaborate on the first partial derivative, we can re-express Equation 2.4 in the following manner:

$$v_k^{\ell+1} = \Sigma_{j=0}^{J}(\omega_{jk}^{\ell+1} \cdot \sigma(v_j^\ell)) \tag{2.14}$$

With the re-expressed equation in place, we can now expand the first partial derivative:

$$\frac{\partial C}{\partial v_j^\ell} = \left(\Sigma_{k=0}^{K}(\omega_{jk}^{\ell+1}\frac{\partial C}{\partial v_k^{\ell+1}})\right) \cdot \sigma'(v_j^\ell) \tag{2.15}$$

It is important to highlight that the summation now iterates over k, representing the number of neurons in the $\ell + 1$ layer.

2.3.7.3.2 Learning Rule Integrating the two partial derivatives back into Equation 2.12 yields:

$$\frac{\partial C}{\partial \omega_{ij}^\ell} = \left(\Sigma_{k=0}^{K}(\omega_{jk}^{\ell+1}\frac{\partial C}{\partial v_k^{\ell+1}})\right) \cdot \sigma'(v_j^\ell) \quad \cdot \quad \alpha_i^{\ell-1} \tag{2.16}$$

This concludes the mathematical derivation of the learning rule, ensuring that every term in the preceding equation is known.

2.3.7.4 Backpropagation: A Computational Approach

From a computational perspective, not all terms in Equation 2.16 are accessible to each neuron for evaluation. This necessitates a data flow capable of disseminating the requisite information for this computation across the network.

Examining the parameters in Equation 2.16, a neuron readily accesses its own activation, denoted as v_j^ℓ, and the inputs to the neuron, represented as $\alpha_j^{\ell-1}$. However, it does not have direct access to weights in the subsequent layer $\omega_{kt}^{\ell+1}$ or to the sensitivity of the cost function concerning the accumulations of neurons in the following layer. It is these parameters that must be disseminated throughout the network.

2.3.7.4.1 Recursion The recursive step is evident in Equation 2.15. Here, the computation of $\frac{\partial C}{\partial v_j^\ell}$ is contingent on the value of $\frac{\partial C}{\partial v_k^{\ell+1}}$. This term, which is called the neuron's internal error, is symbolised by δ.

$$\delta_j^\ell = \frac{\partial C}{\partial v_j^\ell} = \left(\Sigma_{k=0}^{K}(\omega_{jk}^{\ell+1}\delta_k^{\ell+1})\right) \cdot \sigma'(v_j^\ell) \tag{2.17}$$

This internal error, δ, computed for every neuron, becomes a crucial piece of information stored within that neuron.

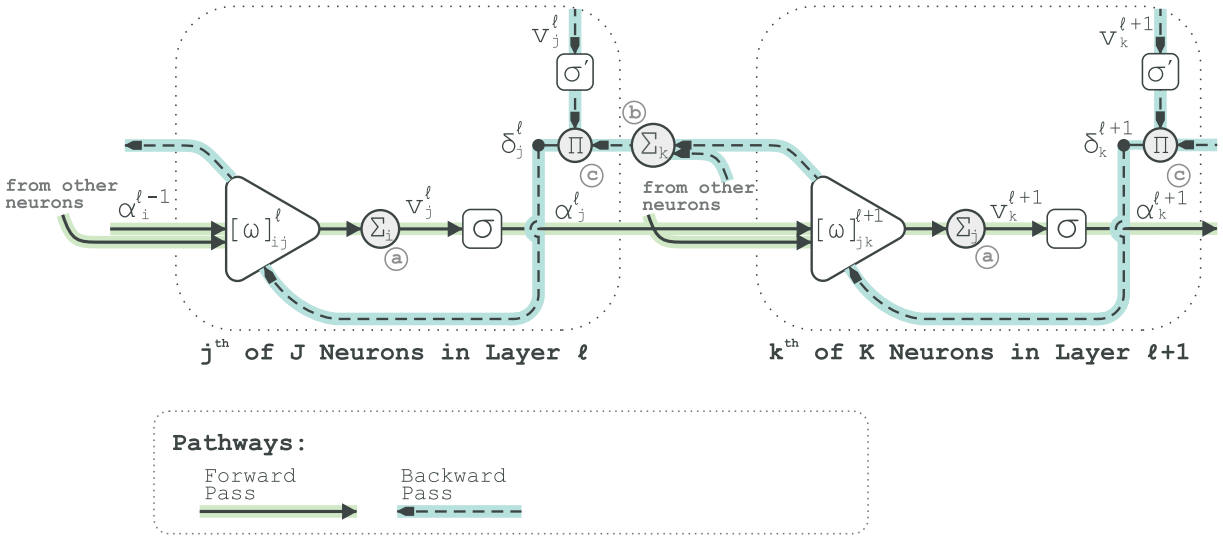

Figure 2.6: *Illustrates the backpropagation of the internal error δ through the network. The green lines highlight the path taken by the internal error values as they move backwards across layers, starting from the output layer and reaching the input layer.*

In Figure 2.6 the green pathways show how this internal error is propagated backwards through the layers. As the backpropagation algorithm progresses, this δ value is systematically propagated backwards throughout the layers of the network. By maintaining and transmitting this error term, each neuron plays its part in the iterative process of fine-tuning the network's weights, helping optimise the overall network for the given task.

2.3.7.4.2 Learning Rule The computational learning rule (or the update rule), using the internal error, transforms the previous heavy expression in Equation 2.16 into a more computationally friendly one. Replacing the chain rule derivatives with a single term of internal error δ, yields:

$$\underset{OPDL}{\Delta\omega_{ij}^\ell} = \eta\delta_j^\ell \cdot \alpha_i^{\ell-1} \tag{2.18}$$

where $\Delta\omega_{ij}^{\ell}$ represents the change to be applied to this weight, η is the learning rate, determining the

size of the weight update step, δ_j^{ℓ} is the internal error for this neuron, and $\alpha_i^{\ell-1}$ is the activation received from the previous layer weighted by this weight. This is the open-loop learning. In essence, it shows how much and in what direction the weight ω_{ij}^{ℓ} should be adjusted in order to reduce the error of the network. The magnitude of the adjustment is influenced by the product of the internal error of the current neuron and the activation of the neuron from the previous layer. The learning rate η ensures that the adjustments are made in controlled increments.

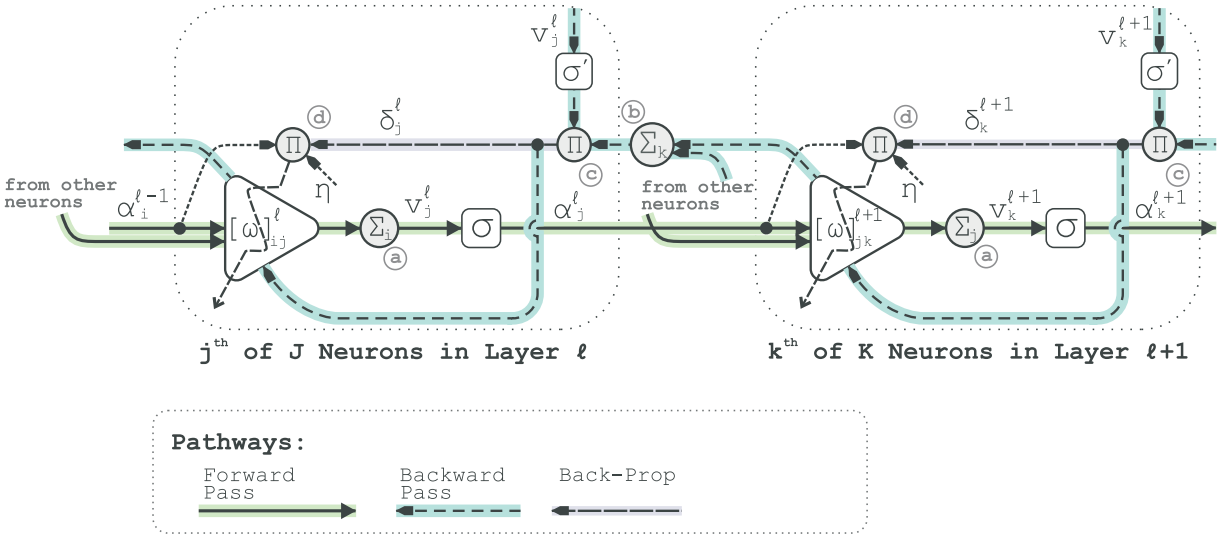

Figure 2.7: *Illustrates the incorporation of the learning rule pathway into the forward and back propagation pathways to complete the signal flow within the network. Node ⓓ highlights the point at which weight adjustments are made, driven by internal error δ computations.*

With the addition of the learning rule pathway, Figure 2.7 shows the comprehensive flow of signals within the neural network. This flow begins with the forward propagation of activations (in blue), followed by the backward propagation of errors (in green), and culminating in the application of the learning rule (in purple). The learning rule, integral to the weight adjustment process, is denoted by node ⓓ. This visualisation offers a holistic perspective of the neuron's operations, encompassing both the information processing and the weight optimisation stages.

2.4 Design & Derivation of Learning Platform

In this work, all presented learning algorithms are based on a common platform that incorporates both a reflex and a learning loop. Before introducing the unique aspects of

each algorithm, we define and explain these common loops.

2.4.1 Reflex: an Agent without Intelligence

In biology, a reflex is an inherent driver of an organism's behaviour, triggered by involuntary actions in response to stimuli. Reflexes provide immediate responses, ensuring the organism's survival and success. In this work, we apply the presented learning algorithm to a navigational task where the robot follows a path. However, the system is not designed to track the path explicitly. Instead, the algorithm responds to any disturbances that deviate from the desired state of the reflex.

For instance, a bend in the path is considered a disturbance. When the robot encounters a bend, its innate reflex mechanism prompts an immediate reaction to recover from the disturbance. Technically, the reflex mechanism is modelled as a fixed closed-loop controller.

2.4.1.1 Reflex Platform

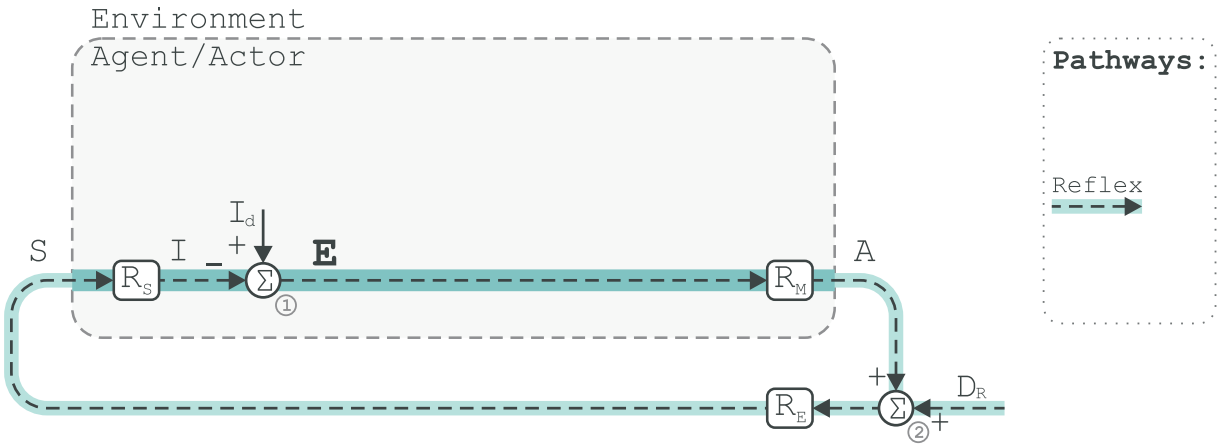

Figure 2.8: The Reflex platform: Illustrates the reflex mechanism within an agent-environment boundary marked by a grey dashed rectangle. The reflex loop is highlighted in dark blue within the agent and in light blue in the environment. Node ① represents the reflex innate mechanism, comparing input I to the desired input I_d, generating an error signal E. This error signal acts on the environment through the motor command transfer function R_M, resulting in action A. Node ② serves as the summation point where this action counteracts disturbance D. After passing through the reflex environment's transfer function R_E, it creates a new state S sensed by the reflex sensory transfer function R_S, leading to a new input I and completing the loop.

Figure 2.8 illustrates the block diagram of the reflex loop. A biological reflex is modelled using a fixed feedback controller. A key distinction is that biological agents lack an external observer or operator. Consequently, the set point or desired state remains constant and

cannot be controlled. Thus, it is modelled within the boundary of the agent rather than as an input to the system, as would be the case with conventional feedback controllers.

This diagram shows the connection between the reflex mechanism within an agent and its environment. The agent and the environment interact directly, exchanging inputs and outputs. A grey dashed rectangle delineates the boundary between the agent and the environment. The reflex pathway is illustrated in dark blue within the agent and in light blue within the environment. This configuration is consistent across all platforms presented in this work.

2.4.1.2 Reflex Objective & the Error Signal

The disturbance acting on the reflex is denoted as D_R. The state of the agent, represented by the variable S, is perceived through one or multiple sensory inputs. The transfer function that converts the states into sensory inputs is illustrated with $\boxed{\mathbb{R}_S}$. As a result, a desired state can be likened to a desired sensory input, denoted as I_d. To maintain its ideal state, the agent must compare any incoming sensory input, I, with its desired input at the summation node ①. If the actual input deviates from the desired input, a non-zero signal known as the error, E, is produced. Hence, the aim of the reflex can be concisely stated as maintaining a zero error signal.

$$E = I_d - I \quad \textit{Therefore:} \begin{cases} \textit{if } E = 0 \rightarrow I = I_d & \textit{desired} \\ \textit{if } E \neq 0 \rightarrow I \neq I_d & \textit{undesired} \end{cases} \tag{2.19}$$

2.4.1.3 Reflex Action

Reflexes detect environmental changes and trigger motor actions to modify their state in response. When a reflex experiences a deviation from its desired state, indicated by a non-zero error, it takes an appropriate action, denoted as A, to restore its desired state. This restorative action is informed by a transfer function that translates the error into motor actions, represented by $\boxed{\mathbb{R}_M}$ for the reflex.

These restorative actions counteract environmental disturbances, which occur at node ②, where the agent's actions and the environmental disturbances interact to create the next state. The environment transforms the agent's actions and disturbances into new states, described by the environment's transfer function, represented by $\boxed{\mathbb{R}_E}$.

2.4.1.4 z-space Derivation of Error

We employ z-transformation to derive the expression of the closed-loop error in the z-domain. Detailed explanations of z-transformation are provided in Appendix A.4. This approach is chosen due to the intricate recursive characteristics of closed-loop platforms,

which make it challenging to formulate an expression of the error signal directly in the time domain. Z-transformation translates this recursion into simple algebraic equations. Our objective is to obtain an expression for E in terms of the closed-loop signals and functions. By substituting for I in Equation 2.19, we obtain:

$$E = I_d - \overbrace{SR_S}^{=I} \tag{2.20}$$

Referring to node ②, we can substitute for S to obtain:

$$E = I_d - \overbrace{(A + D_R)R_E}^{=S} R_S \tag{2.21}$$

Substituting for A completes the recursion for the error signal:

$$E = I_d - (\overbrace{ER_M}^{=A} + D_R)R_E R_S \tag{2.22}$$

Rearranging the above equation with respect to E gives:

$$E(1 + R_M R_E R_S) = I_d - D_R R_E R_S \tag{2.23}$$

Finally, E is found as:

$$E = \frac{I_d - D_R R_E R_S}{1 + R_M R_E R_S} \tag{2.24}$$

Prior to deriving the closed-loop expression for the error in the platform with a learning loop, it is crucial to acknowledge that the error E is impacted by the disturbance D_R, without any counteracting factor that could potentially nullify the effect of the disturbance. In other words, the agent has no control over other variables in the equation, namely R_E, R_S, and R_M, therefore, a non-zero disturbance D_R directly results in an error.

2.4.2 In Search of an Intelligent Agent

Suppose the agent can generate an output O that pre-emptively counteracts the disturbance before it enters the system. We add this term to the expression of E as follows:

$$E = \frac{I_d - (O + D_R)R_E R_S}{1 + R_M R_E R_S} \tag{2.25}$$

In such a scenario, the O term would be capable of neutralising (or cancelling out) the disturbance even before it is detected by the reflex. Precisely, if we let $O = \frac{I_d}{R_E R_S} - D_R$, then $E = 0$. In the following section, we demonstrate how a learning loop can be designed and incorporated into the platform to furnish the agent with the ability to generate such O term.

2.4.3 The Learner: an Agent with Intelligence

2.4.3.1 Learning Platform

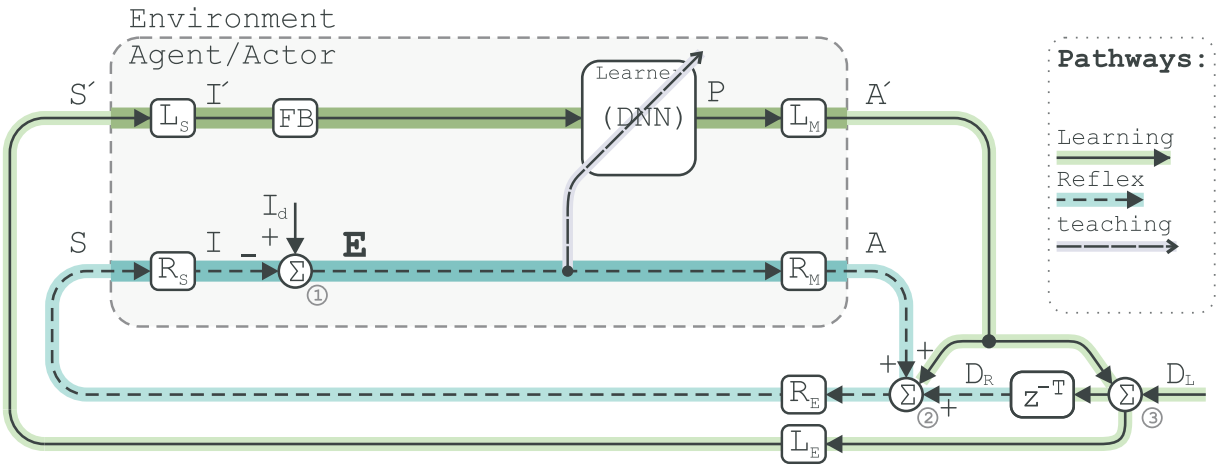

Figure 2.9: *A generic learning platform is integrated with the reflex platform: The learning loop depicted in dark green within the agent boundaries and light green in the environment. At node 3, the learner's action is combined with the early disturbance, delayed by T time steps using z^{-T} in z-space. This summation creates a new learner state, S', sensed by the sensory transfer function L_S to generate inputs I'. These inputs are filtered through a FIR filter bank FB to optimise correlation with the error. The filtered signals pass through the network to produce output P, which is then processed by the learner's motor unit, L_M. This action is directed to node 2, where it attempts to counteract the disturbance at the reflex loop's entry point. If successful, no error is generated. If unsuccessful, a non-zero error signal is generated and sent to the learner for training via the purple pathway.*

Figure 2.9 shows a general learning platform where a learning loop is integrated into the previously illustrated reflex loop (Figure 2.8). The learning loop is depicted with solid lines highlighted in dark green within the agent and light green in the environment.

Like the reflex loop, the learning loop interacts with its environment by receiving sensory inputs and producing motor outputs. The goal of the learning system is to prevent

disturbances from reaching the reflex, similar to the previously mentioned O term. To achieve this, the learning mechanism must have: 1) prior knowledge of the disturbance, and 2) the ability to act upon it before it reaches the reflex.

Therefore, it is important to differentiate between the disturbance as it appears to the learning loop (D_L) and the disturbance once it reaches the reflex (D_R). The difference between these two disturbances is the time delay between them. This is because a disturbance first appears in the field of view (FoV) of the learner before it can reach the reflex sensors. The transfer function $\boxed{z^{-T}}$ in z-space represents this delay. In the z-space, this relationship can be expressed as:

$$D_R = D_L \cdot z^{-T} \tag{2.26}$$

Just like in the reflex loop, an early disturbance (D_L) in the learner's environment (L_E) leads to a new state (S') for the learner. The learner's sensory unit (L_S) translates this new state into sensory inputs (I'), which the learner uses to generate an output through its motor effector. This results in an action (A') exerted on the environment.

2.4.3.2 Error Feedback for Learning

In contrast to a reflex, a learner does not possess an innate, fixed response to disturbances. Rather, it acquires the ability to improve its actions through a process of learning. The learner requires instructive external feedback on its performance. As previously mentioned, in a reflex loop, a non-zero error signal is generated when the desired state is lost. This error signal serves as feedback for training the learner. However, this approach provides minimal feedback, only indicating whether the action taken was adequate or not, without offering detailed instructions on the correct actions to take.

2.4.3.3 The Importance of Delays & Correlation for Learning

Unlike the reflex mechanism, the learning loop does not rely on a preferred or desired input. Instead, it monitors the inputs and the subsequent error signals to learn about their correlation, with the goal of generating an output that will result in zero subsequent error. The key concept here is correlation. For the learner to identify the correlation between inputs and their corresponding errors, the inputs must not be transient.

To achieve this, a filter bank (FB) is required to introduce a sufficient amount of delay to the inputs, allowing them to persist long enough to be correlated with their corresponding errors. This delay must match the delay between the two disturbances explained above. However, the delay between when events are perceived by the learner and when they reach the reflex cannot be modelled with a precise number of sampling

delays. This is because these events travel through the non-linear environment, which affects the delay unpredictably. The actions taken by the learner can influence this delay. In fact, once the learner has fully adapted, the disturbing events never reach the reflex, effectively making this delay infinite. Additionally, various factors such as the speed of the robot and the angle of the bend can also influence this time delay. Due to these complexities, the delay is modelled using a filter bank, which allows for a wide range of time delays to be covered. Further details about the FB are presented in later sections.

2.4.3.4 z-space Derivation of Error

With the addition of the learning loop, we can derive an updated expression for the error signal. Referring to Equation 2.20 and the summation node ② in Figure 2.9, we substitute S with:

$$E = I_d - \overbrace{(A + A' + D_R)R_E}^{=S} R_S \tag{2.27}$$

Substituting for A and A' yields:

$$E = I_d - (\overbrace{ER_M}^{=A} + \overbrace{PL_M}^{=A'} + D_R)R_E R_S \tag{2.28}$$

We have now completed the recursion for the error signal. Rearranging to solve for E, we obtain:

$$E(1 + R_M R_E R_S) = I_d - (PL_M + D_R)R_E R_S \tag{2.29}$$

Finally, E is calculated as:

$$E = \frac{I_d - (PL_M + D_R)R_E R_S}{1 + R_M R_E R_S} \tag{2.30}$$

As seen earlier in Equation 2.25, the learning loop has now equipped the agent with a factor $O = PL_M$ that can mitigate the effects of later disturbance D_R before it reaches the reflex. This means that by generating an appropriate output $P = \frac{I_d}{R_E R_S L_M} - \frac{D_R}{L_M}$, the learner can ensure zero error.

It is important to note that this output depends upon the transfer functions of both the learner and the reflex. Thus, to generate this output, the neural network constructs a

forward model of the environment, encompassing these transfer functions.

2.4.4 Towards Specific Learning Paradigms

The generic learning platform defines the relationships between all inputs, outputs, disturbances, and errors, with the ultimate aim of generating the correct output P. However, this output is a function of the intrinsic connections and inner workings of the learner. The learner must find the correct internal parameters to generate the correct output. The error signal plays a crucial role in this process by fine-tuning the internal parameters, which in turn affects the error signal itself. In other words, learning can be thought of as a continuous "dialogue" between the internal parameters of the learner and the error signal. The specific ways in which the error signal is utilised to adjust the internal parameters give rise to unique learning rules and algorithms. In the upcoming sections, we will introduce and delve into these specialised learning paradigms: ICO and CLDL in this chapter, and SaR, PaM, FCL, and Echo in the following chapters.

2.5 Design & Derivation of Input Correlation (ICO) Learning

The first learning algorithm examined in this study is the ICO learner. This paradigm aligns with the overarching learning framework presented above, featuring a learner comprising a single layer of neurons responsible for input correlation. The development of this algorithm was not a primary aspect of this undertaking; credit goes to Bernd Porr for developing this algorithm (Porr and Wörgötter, 2006). However, replicating this algorithm provided an introductory step and a preparatory exercise for the principal project - CLDL.

2.5.1 Motivation: An Overview

The ICO learner operates by adjusting weights based on input correlations, drawing a comparison to the well-established Hebbian learning principle. In Hebbian learning, the strength of synaptic connections is modulated by correlating input and output activities. However, Hebbian methods are known to encounter issues related to self-correlation, potentially causing instability and restricting the feasible learning rate (Porr and Wörgötter, 2006).

In contrast, the ICO learner follows a unique learning rule related to the ISO-learning rule of Porr and Wörgötter (2003). However, the ICO learner deviates by substituting the derivative of the reflex input for the derivative of the output within the learning rule. This

innovative approach relies exclusively on input correlations, effectively embracing a distinct heterosynaptic learning mechanism. Consequently, this strategy yields significantly improved outcomes.

By excluding the output in the learning rule, the problematic autocorrelation element that leads to destabilisation is eliminated. This elimination enables the use of higher learning rates. Mathematically, it has been demonstrated that this new rule can achieve the theoretical optimum of one-shot learning under ideal conditions.

2.5.2 Learning Platform

Figure 2.10 illustrates the positioning of the ICO learner within the overarching generic learning platform. At the interface between the learner and the platform are the predictive inputs denoted as I', predictive action represented as P, and the error signal E.

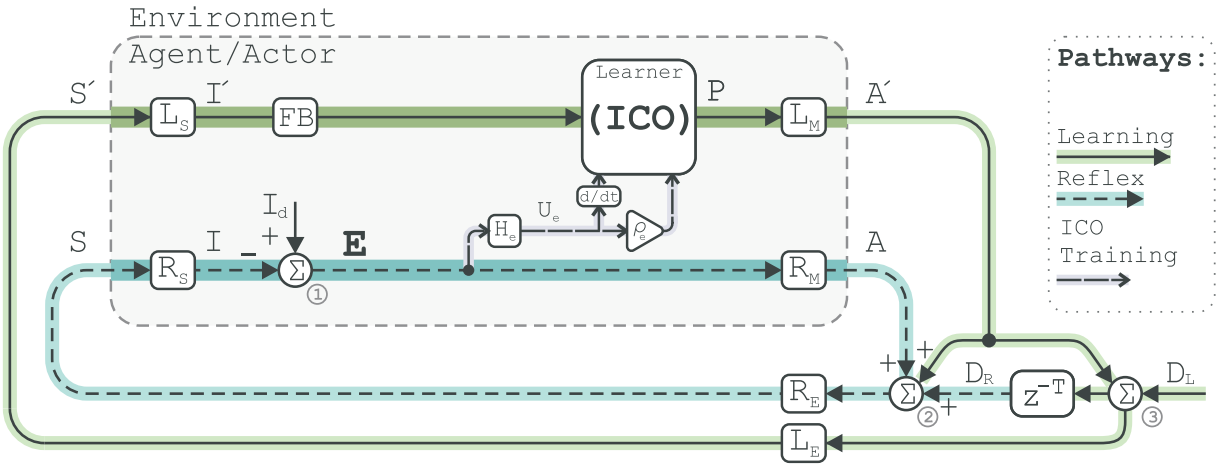

Figure 2.10: *Displays the ICO learner platform: The error signal is first filtered by H_e, and the derivative of the filtered result U_e is transmitted to the neurons of the ICO learner. A weighted signal derived from U_e is subsequently sent to the summation point within the ICO learner.*

2.5.3 Inner Working of Neurons

Figure 2.11 provides insight into the internal configuration of an ICO learner, designed to produce a singular output. The ICO learner encompasses a single layer of neurons, with its inputs designated as I'_i. These inputs undergo filtration by the impulse response H_{ij}. Filters H_{i1} are in effect in this case, and the dotted filters H_{i2} show how more filters can be added to the learner.

In the z-domain, this relationship can be articulated as:

$$I'_i H_{ij} = U_i \tag{2.31}$$

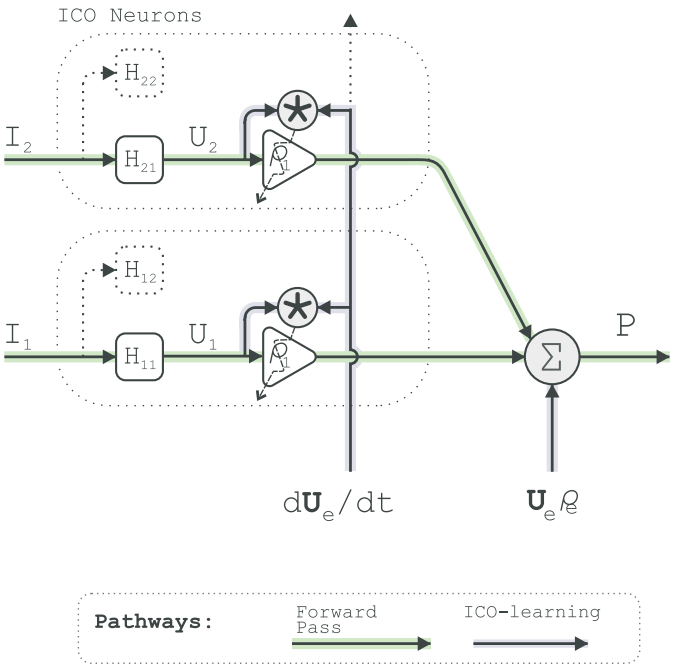

Figure 2.11: *Illustrates the inner mechanisms of the ICO learner: The predictive inputs I_i undergo filtering via filters H_{ij} resulting in U_i, each weighted by ρ_i, followed by summation with the filtered and weighted error term $U_e\rho_e$. This process yields the ICO output denoted as P, which aligns with the platform output. During the training process, the derivative of the filtered error U_e is correlated with the filtered inputs, driving adjustments in the weights.*

The feedback error signal, denoted as E, serves as an additional input to the neuron. Note that the error is filtered by H_e, meaning it is altered prior to being processed by the neuron.

$$EH_e = U_e \tag{2.32}$$

The neuron generates an output by aggregating the weighted filtered inputs, encompassing both the predictive inputs and the error signal:

$$P = \rho_e U_e + \Sigma_{i=1}^{N}(\rho_i U_i) \tag{2.33}$$

Here, P represents the learner's output, which functions as the predictive steering command for the robot. The dashed lines in Figure 2.11 demonstrate how the ICO learner could be expanded to incorporate additional predictive inputs and individual filter banks for each input. It is crucial to note that the structure of the ICO learner is confined to a single layer of neurons. Each filter in the filter bank is tailored to approximate a specific time interval between the occurrence of the predictive signal and the generation of the corresponding error signal. By employing a sufficiently extensive filter bank, it becomes

feasible to approximate all suitable time intervals, as demonstrated in Porr and Wörgötter (2003).

2.5.4 Learning Rule

In a mathematical context, learning unfolds through the adjustment of neurons' weights, with learning deemed successful when these weights reach a stable state. The alteration of the weights is governed by the correlation between the inputs, encompassing both the error and predictive signals, hence the term ICO. The update rule for the weights of the ICO learner can be expressed as follows:

$$\frac{d\rho_i}{dt} = \mu u_i \frac{du_e}{dt} \tag{2.34}$$

In the equation above, μ symbolises the learning rate, while the presence of the derivative signifies the utilisation of the differential learning mode, as elaborated in Porr and Wörgötter (2006).

2.5.5 Limitation

A significant drawback of the ICO learner is its confinement to a single layer. This limitation curtails the network's ability to effectively process large input sets and extract intricate, complex features. The differentiation and correlation of the error signal with neuron inputs render this signal unsuitable for propagation within multi-layer configurations. This prepares the scene for the introduction of the main project, wherein the shallow structure is substituted with a deep network of neurons. This deep architecture proves effective at handling extensive input sets and extracting complex and hidden features due to its multi-layer architecture. This approach, termed CLDL, will be comprehensively examined in the forthcoming section.

2.6 Design & Derivation of CLDL Algorithm

2.6.1 Learning Platform

As explained earlier, the closed-loop learning platform relies on the error signal derived from the reflex loop to provide feedback and enable the learning process. Yet, the precise manner in which this feedback is harnessed hinges upon the architecture and operational aspects of the learning unit. In this section, we delve into the application of a deep neural network as the learning unit within the framework of the closed-loop learning platform. This concept is depicted in Figure 2.12, which showcases the utilisation of a deep neural network employing the backpropagation (BP) technique. In direct contrast to the ICO

learner, the error signal is fed to the learner at the output layer and is subsequently propagated through the various layers of the network.

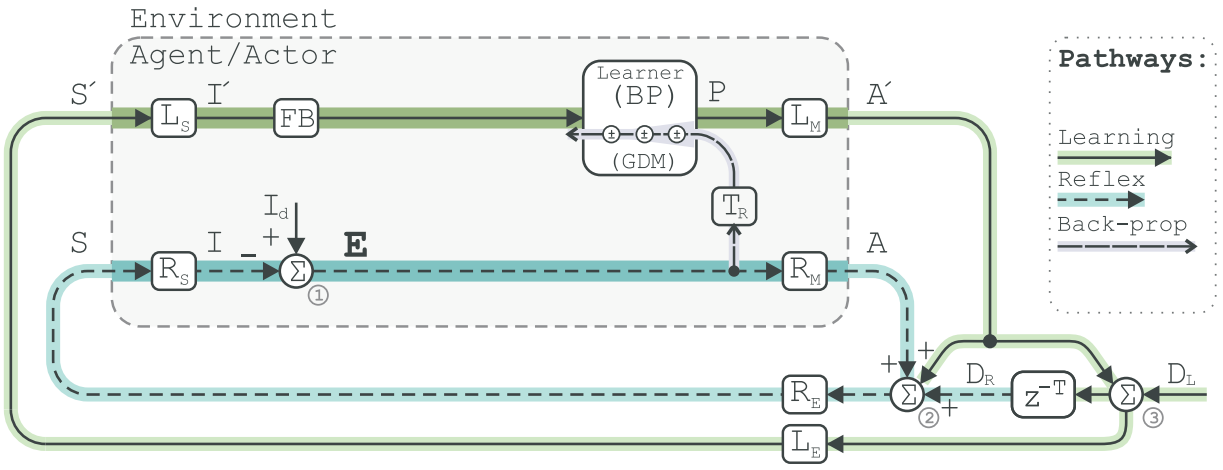

Figure 2.12: *Displays the closed-loop deep learning (CLDL) platform. It illustrates that the error signal undergoes transformation through the T_R transfer function and is subsequently introduced into the deep network at its output. This, in turn, triggers weight adjustments in each layer through the process of backpropagation.*

Section 2.3 outlined a mathematical derivation of the backpropagation technique in the context of open-loop deep neural networks. In this section, we will illustrate the customisation of this technique for implementation in closed-loop learning platforms in which the error linked with the output of the network P is not a known entity. This implies that neither the agent nor the environment is aware of the desired output P_d. Instead, the desired input to the reflex mechanism of the agent is known, therefore, the error can be defined at the point of reflex input. This is called input-control. To establish this distinction, we define the closed-loop cost function as the quadratic form of the closed-loop error:

$$C_c = E^2 \tag{2.35}$$

The objective is to formulate a learning rule that adjusts the weights to minimise the closed-loop cost function. This minimisation process is instrumental in attaining zero error and accomplishing the overarching learning goal. Adhering to the approach outlined in Mehta and Merhav (1986), we perform partial derivatives within the z-domain and operate under the assumption that weight adjustments over time occur at a relatively gradual

pace compared to the dynamics of the closed-loop system (Porr and Miller, 2020). This allows us to represent the sensitivity of the cost function concerning the weights as follows:

$$\frac{\partial C_c}{\partial \omega} = 2|E|\frac{\partial E}{\partial \omega}\Big|_{\omega_{min}} = 0 \begin{cases} E = 0, \textit{ learning goal} \\ E \neq 0, \textit{ local minima} \end{cases} \tag{2.36}$$

To derive an update rule, it is advantageous to distinguish between the closed-loop context and the internal dynamics of the network. This distinction is imperative due to the unique characteristics and the rate of change of these components. While closed-loop signals exhibit rapid changes, the inner parameters are presumed to undergo slower variations. The network's output P acts as the intermediary signal or factor bridging these two domains. By employing the chain rule, as explained in the theory section, we can mathematically represent this relationship as follows:

$$\frac{\partial C_c}{\partial \omega} = \frac{\partial C_c}{\partial P} \cdot \frac{\partial P}{\partial \omega} = G_c G_n \tag{2.37}$$

The initial partial derivative, termed the closed-loop gradient, exclusively concerns the closed-loop dynamics, whereas the second derivative, the network gradient, is solely relevant to the network's internal mechanisms. With this distinction established, we can now delve into the examination of each of these gradients within the framework of their respective systems.

Commencing with the closed-loop gradient and referencing Equation 2.35, we acquire:

$$G_c \coloneqq \frac{\partial C_c}{\partial P} = 2|E|\frac{\partial E}{\partial P} \tag{2.38}$$

By referencing Equation 2.30, the derivative of E with respect to P is determined as follows:

$$G_c = 2|E|\frac{-L_M R_E R_S}{1 + R_M R_E R_S} = 2|E|T_R \tag{2.39}$$

Where T_R is the transfer function of the reflex loop.

2.6.2 Drawing Parallels to Open-Loop Learning

This section explores the connection between closed-loop deep learning and the conventional, widely accepted open-loop deep learning methodologies. It serves to demonstrate that minimising our closed-loop error is equivalent to minimising the open-loop error. In practical applications like robot navigation and other real-life robotic tasks, the open-loop

error is unavailable. Thus, it becomes imperative to address this discrepancy and show how the closed-loop error can effectively be employed for training purposes and minimised by the learner.

In contrast to closed-loop applications, the open-loop error E_o is determined at the network's output. This is output-control. It represents the difference between the network's output P and the desired output P_d, the latter being a known quantity in open-loop applications. This is calculated as:

$$E_o = P_d - P \tag{2.40}$$

Therefore, the open-loop cost function is defined as follows:

$$C_o = {E_o}^2 = (P_d - P)^2 \tag{2.41}$$

The network's objective is to minimise this cost function, a goal accomplished through the utilisation of the gradient descent technique. Similar to Equation 2.37 for the closed-loop scenario, the minimisation of this cost function with respect to the network's weights results in:

$$\frac{\partial C_o}{\partial \omega} = \frac{\partial C_o}{\partial P}\frac{\partial P}{\partial \omega} = G_o G_n \tag{2.42}$$

Here, we encounter the network gradient again. However, the first partial derivative corresponds to the open-loop gradient. By applying equations 2.41 and 2.40, this can be calculated as:

$$G_o = 2E_o\frac{\partial E_o}{\partial P} = 2E_o = 2(P_d - P) \tag{2.43}$$

Given this definition, we can explore the interconnection between the open-loop and closed-loop errors. In the closed-loop scenario, solely the intended input to the reflex is ascertainable. Yet, it is understood that this intended input can solely be generated by the desired output of the network. To articulate this interrelation, we substitute for I_d in Equation 2.28 in the ensuing manner:

$$E = (ER_M + P_d L_M + D_R)R_E R_S - (ER_M + PL_M + D_R)R_E R_S \tag{2.44}$$
$$= (P_d - P)L_M R_E R_S \tag{2.45}$$
$$= E_o \quad L_M R_E R_S \tag{2.46}$$

This signifies that the deviation in parameter P cannot be measured directly. Rather, it traverses through the motor actions of the learner L_M, the reflex's environmental component R_E, and the sensory unit of the reflex R_S before becoming evident as the closed-loop error E. As depicted in Equation 2.46, it is evident that the minimisation of the open-loop error mirrors the minimisation of the closed-loop error, and vice versa. Nevertheless, this is contingent upon the product (or sequence) of the aforementioned transfer functions not equating to zero:

$$L_M R_E R_S \neq 0 \quad \textit{, therefore:} \quad E = 0 \iff E_o = 0 \tag{2.47}$$

Therefore, it becomes viable to substitute the open-loop error and cost function with their closed-loop counterparts for the gradient descent technique.

Having established the dynamics of the closed loop, we transition to investigating the intrinsic interconnections and parameters within the network. This involves understanding how these elements acquire the ability to produce desired outputs through the process of learning.

2.6.3 Inner Working of Neurons

Forward propagation and back-propagation were explained in previous sections. The key distinction in this context is that the closed-loop cost function is based on the closed-loop error. The sensitivity of this cost function to the weights is determined by separating the dynamics of the platform from those of the network.

We previously explored the platform gradient and linked it to the open-loop gradient. In this section, we will delve into the network gradient, denoted as G_n. When considering a specific weight, this is defined as:

$$G_n = \frac{\partial P}{\partial \omega_{ij}^{\ell}} \tag{2.48}$$

This is then unravelled using the chain rule, with the linking term being the activation of the corresponding neuron:

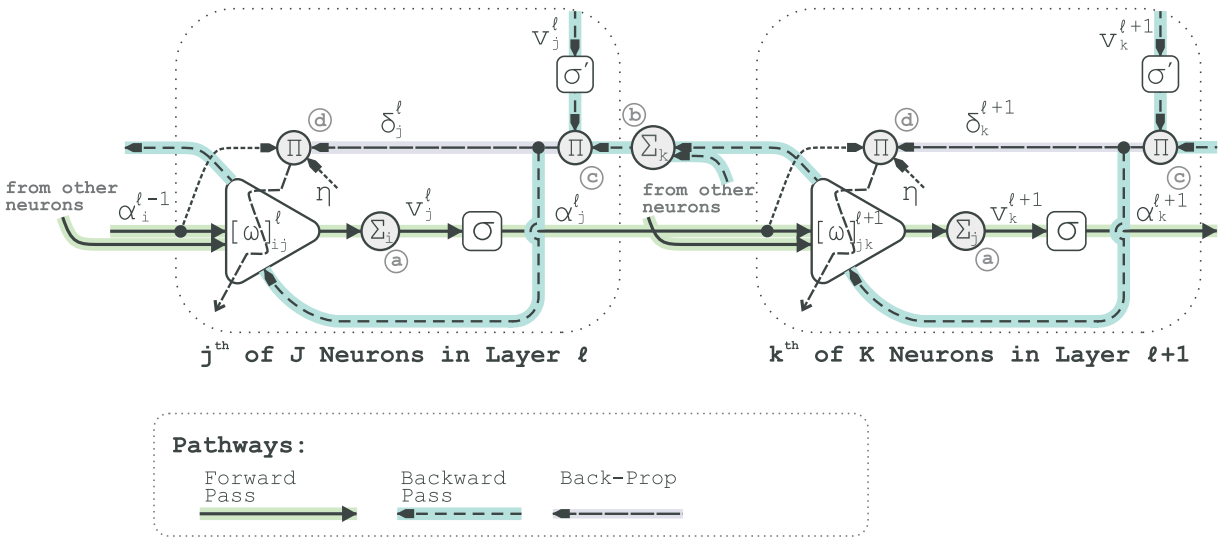

Figure 2.13: *Displays the internal connections between two neighbouring neurons within PaM network. Forward propagation of inputs in shown with the left-to-right solid lines highlighted in green. σ is the sigmoid activation function and A_j^ℓ denotes the activation of the j^{th} neuron in layer ℓ. $[\omega]_{ij}^\ell$ is the weight matrix associated with inputs I to this layer. The summation node ⓐ corresponds to Equation 2.4. Backpropagation pathway is shown with right-to-left dashed lines highlighted in blue. The summation at node ⓑ and product at node ⓒ correspond to Equation 2.16. The internal error at node ⓓ, together with the learning rate η and the input to the neuron $A_i^{\ell-1}$, join to drive the learning rule, corresponding to Equation 2.54.*

$$G_n = \frac{\partial P}{\partial v_j^\ell} \cdot \frac{\partial v_j^\ell}{\partial \omega_{ij}^\ell} \tag{2.49}$$

The latter partial derivative is the activation of the neuron associated with the weight:

$$\frac{\partial v_j^\ell}{\partial \omega_{ij}^\ell} = \alpha_i^{\ell-1} \tag{2.50}$$

The former partial derivative contains the linking term related to the closed-loop platform. This term is termed the "linking error", denoted by γ. It is computed using the back-propagation technique explained in Section 2.3:

$$\gamma_j^\ell = \frac{\partial P}{\partial v_j^\ell} = \frac{\partial P}{\partial \alpha_j^\ell} \cdot \frac{\partial \alpha_j^\ell}{\partial v_j^\ell} = \Sigma_{k=0}^{K}(\omega_{jk}^{\ell+1}\gamma_k^{\ell+1}) \cdot \sigma'(v_j^\ell) \tag{2.51}$$

Recall the derivation of the open-loop internal error presented in Equation 2.17. By definition, the internal error represents the sensitivity of the cost function with respect to

the accumulation within a neuron. In the closed-loop context, this is expressed as:

$$\delta_j^\ell = \frac{\partial C}{\partial v_j^\ell} = \frac{\partial C}{\partial P} \cdot \frac{\partial P}{\partial v_j^\ell} \tag{2.52}$$

Based on the derivations presented in Equations 2.39 and 2.51, we can conclude the following:

$$\delta_j^\ell = 2|E| \frac{-L_M R_E R_S}{1 + R_M R_E R_S} \frac{\partial P}{\partial \alpha_j^\ell} = 2|E| T_R \cdot \gamma_j^\ell \tag{2.53}$$

2.6.4 Learning Rule

The learning rule for closed-loop deep learning can be formally articulated as:

$$\underset{\text{CLDL}}{\Delta \omega_{ij}^\ell} = \eta \frac{\partial C}{\partial \omega_{ij}^\ell} = \eta \delta_j^\ell(z) \alpha_i^{\ell-1}(-z), \quad \eta << 1 \tag{2.54}$$

Wherein the learning rate, denoted by η, is sufficiently small to ensure that the network's dynamics are inconsequential in relation to those of the closed-loop platform. The employment of the complex variable z serves to underscore for the reader that this learning paradigm operates within the z-space, in contrast to the open-loop approach.

To authenticate the back-propagation mechanism in the z-space, we can dissect the update rule for an arbitrary weight component and subsequently investigate expressions pertinent to the propagation dynamics.

$$\Delta \omega_{ij}^\ell = \eta \cdot \frac{\partial C}{\partial \omega_{ij}^\ell} \tag{2.55}$$

$$= \eta \cdot \frac{\partial C}{\partial P} \cdot \frac{\partial P}{\partial v_j^\ell} \cdot \frac{\partial v_j^\ell}{\partial \omega_{ij}^\ell} \tag{2.56}$$

$$= \eta \cdot 2|E| \frac{-L_M R_E R_S}{1 + R_M R_E R_S} \cdot \Sigma_{k=0}^K (\omega_{jk}^{\ell+1} \gamma_k^{\ell+1}) \cdot \sigma'(v_j^\ell) \cdot \alpha_i^{\ell-1} \tag{2.57}$$

Distributing the closed-loop term inside the summation and subsequently refining the expression, we obtain:

$$\Delta\omega_{ij}^{\ell} = \eta \cdot \Sigma_{k=0}^{K}(\omega_{jk}^{\ell+1}\delta_{k}^{\ell+1})\alpha_{i}^{\ell-1} \tag{2.58}$$

$$= \eta \cdot \Sigma_{k=0}^{K}(\omega_{jk}^{\ell+1}\frac{\Delta\omega_{jk}^{\ell+1}}{\eta \cdot \alpha_{j}^{\ell}})\alpha_{i}^{\ell-1} \tag{2.59}$$

$$= \eta \cdot \Sigma_{k=0}^{K}(\omega_{jk}^{\ell+1}\Delta\omega_{jk}^{\ell+1})\frac{\alpha_{i}^{\ell-1}}{\eta \cdot \alpha_{j}^{\ell}} \tag{2.60}$$

$$= \Sigma_{k=0}^{K}(\omega_{jk}^{\ell+1}\Delta\omega_{jk}^{\ell+1})\frac{\alpha_{i}^{\ell-1}}{\alpha_{j}^{\ell}} \tag{2.61}$$

It is evident that the weight modifications in deeper neural layers exert an influence on the weight adjustments in the proximal layer, thereby illustrating the propagation mechanism inherent to the closed-loop system.

2.7 Application & Experimental Setup

The learning paradigms developed in the previous sections are implemented and evaluated using a line-following robot, encompassing both simulation-based and custom-built physical robot experiments. In both scenarios, the robot is positioned on a canvas and tasked with navigating a looped path.

2.7.1 Simulation Environment

The simulation environment was constructed using Enki (Porr and Daryanavard, 2020), a software framework developed in the Qt C++ programming language. This choice facilitated the integration of physics principles into the simulation framework. Figure 2.14 depicts the simulation environment, showcasing the canvas—a JPEG image simulating a 100 cm by 100 cm playground incorporating the c-shaped looped pathway. The primary objective for the robot involves adhering to this trajectory with a symmetrical orientation relative to the path.

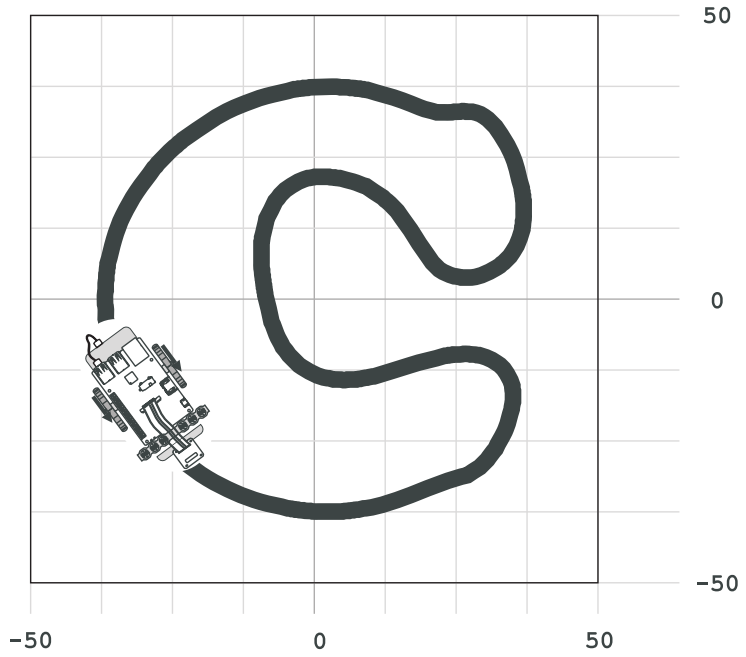

Figure 2.14: *The environment for simulations modelled with Qt: The canvas, the c-shaped path, and the robot (not to scale) placed on it.*

2.7.2 Real Experimental Setup

2.7.2.1 The Playground: Canvas

Figure 2.15 showcases the experimental setup. The canvas measures 80cm by 100cm, featuring a looped path resembling the shape of a butterfly. This canvas was employed as

the physical setting for our empirical investigations. Within this setting, the primary aim of the robot is to trace the path while maintaining a symmetrical position relative to the path.

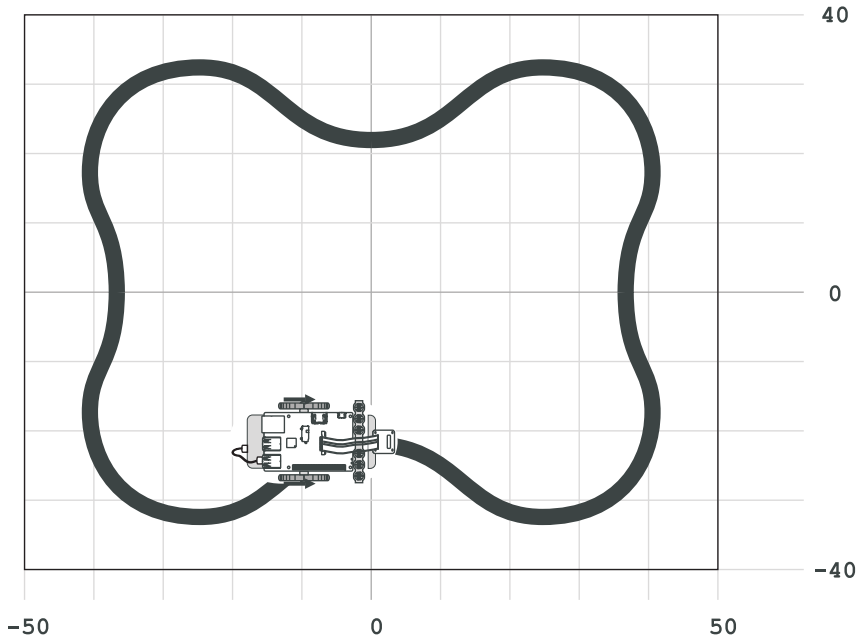

Figure 2.15: *The environment for the real experiments: The canvas, the path, and the robot (not to scale) placed on it.*

2.7.2.2 The Custom-Made Physical Robot

The robot was constructed using a Parallax SumoBot as a mechanical test-bed. A RPi served as the central computational unit, and an Arduino Nano was utilised as the motor controller. Figures 2.16 portray schematic drawings of the robot (not to scale). The robot's chassis accommodates a battery and houses component wiring. It incorporates two wheels, an array of light sensors, and a camera. The RPi functions as the central processing unit for hosting learning algorithms, which can be accessed remotely over WiFi.

The robot can navigate utilising either its reflex mechanism, its learning algorithm, or a combination of both. In the following sections, we will detail the attributes of the reflex and learning mechanisms. These components are shared between the simulated robot and the custom-built robot. Pictorial representation of the robot are shown in Figure 2.17.

2.7.2.3 Wireless Transmission

The robot is equipped with a wireless navigation system powered by a battery bank. This setup enables the robot to navigate its designated path smoothly. For the purpose of monitoring and controlling the experiments, a wireless communication arrangement was

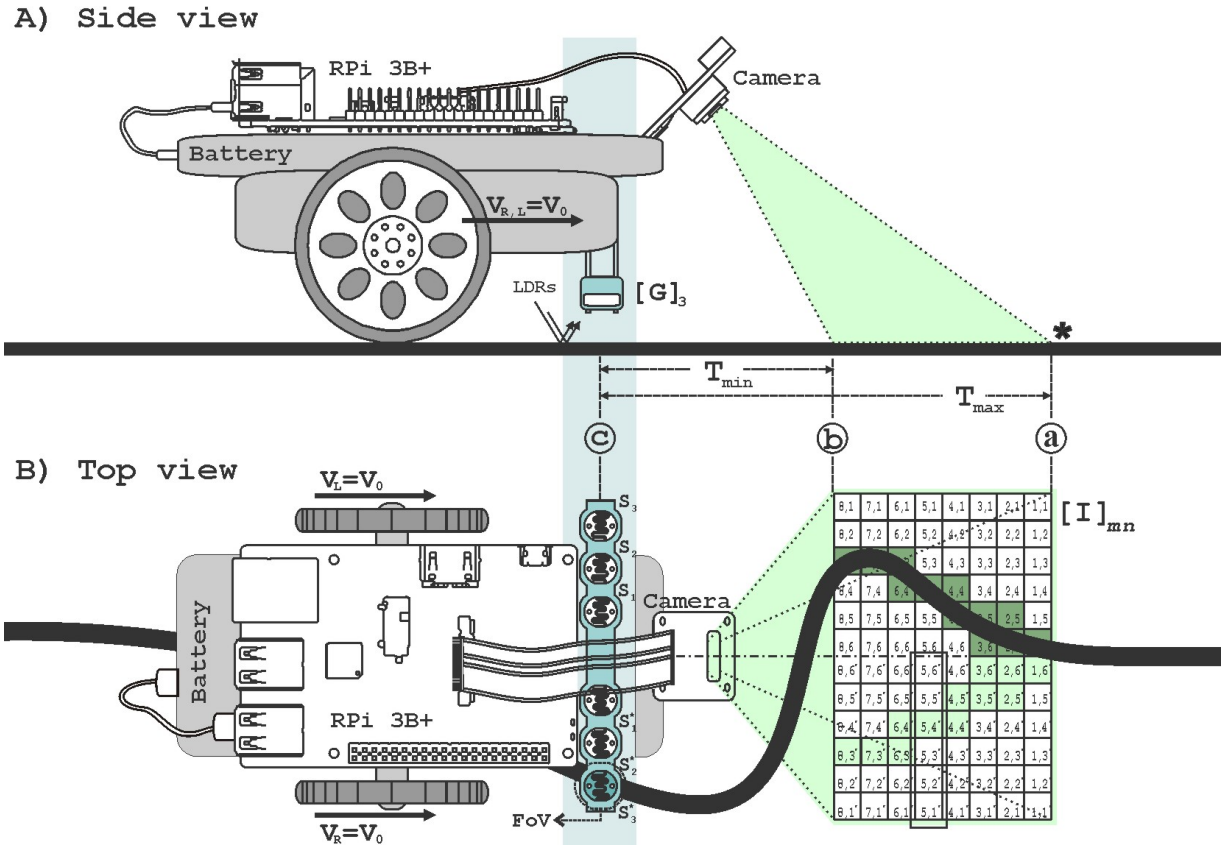

Figure 2.16: *Shows the configuration of the robot and the canvas on which it navigates. A battery bank is placed on the chassis that powers the raspberry pi 3B+ (RPi) and provides power to the array of light sensors* $[G]_6$ *and the motors. The camera provides vision of the path ahead from point ⓐ to ⓑ. The star sign marks a disturbance in the path, such as a bend.* A) *Shows the side view* B) *Shows the top view*

established between a personal computer and the RPi enclosed within the robot. This configuration facilitates real-time tracking of the robot's movements, parameter adjustment, and data collection throughout the course of the experiments.

2.7.2.4 Graphical User Interface (GUI)

The experiments were monitored using a custom-designed GUI, depicted in Figure 2.18. This tool is essential for manipulating the physical parameters of the setup to determine the optimal configuration for the experiments. Specifically, the error multiplier and net output multiplier can be adjusted via a sliding bar to identify a configuration that enables stable trials across a broad range of learning rates. Once these parameters are identified, they are maintained consistently across all trials for a given set of results.

Additionally, the GUI displays signals from the sensors (coloured traces), the resulting error signal (black trace), and the integral error (white trace) for monitoring purposes. The top section shows the number of time-steps or iterations of learning, as well as the

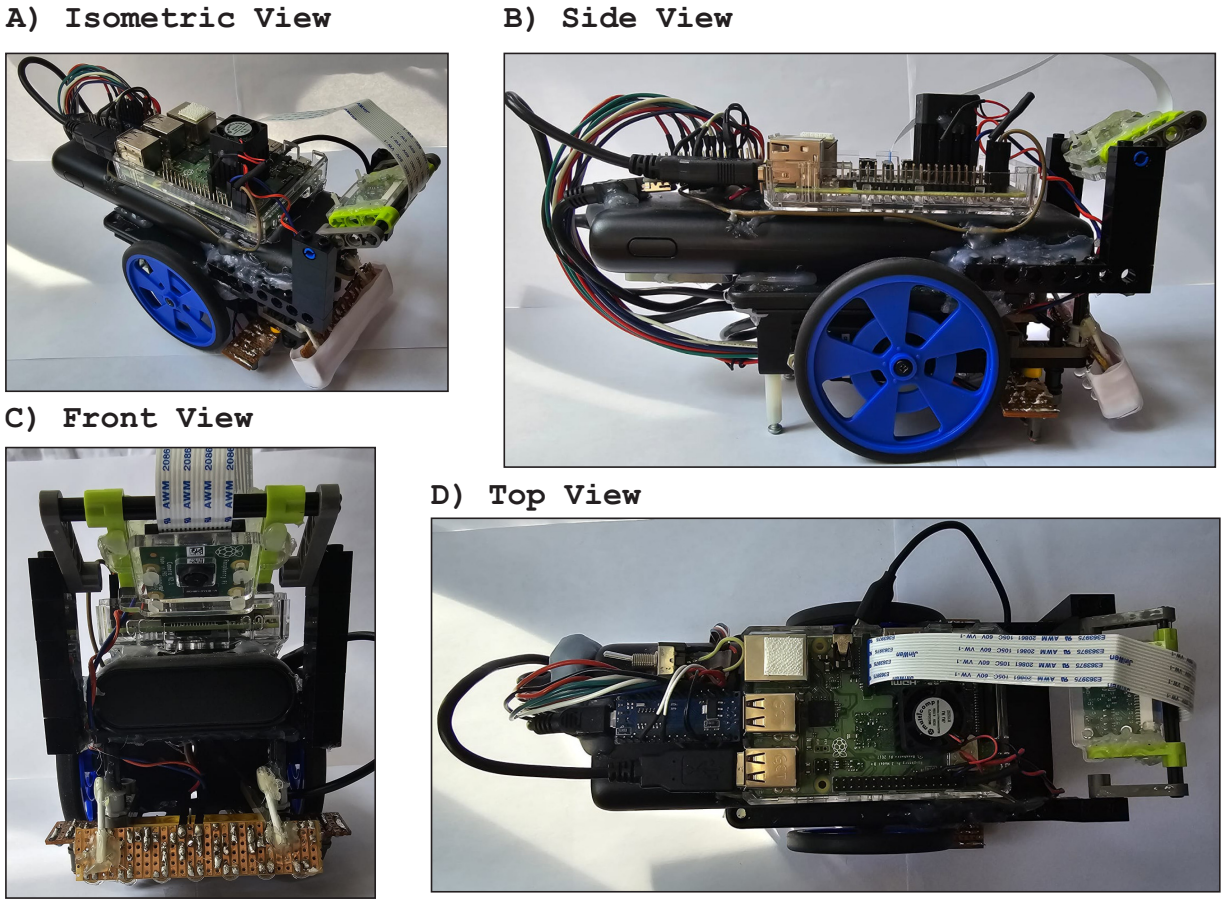

Figure 2.17: *Pictorial representation of the custom-made robot:* A) *Isometric view,* B) *Side view,* A) *Front view, and* A) *Top view.*

maximum and average values of the error integral, including the threshold below which successful learning is achieved. Furthermore, the raw and final values of the error and network output, after multiplication by their respective gains, are presented in the top banner.

2.7.3 Reflex Components of the Robot

2.7.3.1 Light-Dependent Resistor (LDR) Array

The reflex mechanism receives sensory inputs from an array of light sensors. This array comprises six light dependent resistor (LDR) symmetrically positioned underneath the robot's chassis in proximity to the canvas, as illustrated in Figure 2.16A. The LDR will be referred to as the 'sensor' in the following sections. On the left and right sides of the robot, individual sensors are labelled as $S_{1,2,3}$ and $S^*_{1,2,3}$, respectively. The star symbol indicates the sensor's symmetrical positioning relative to its counterpart on the opposite side of the robot, as depicted in Figure 2.16B.

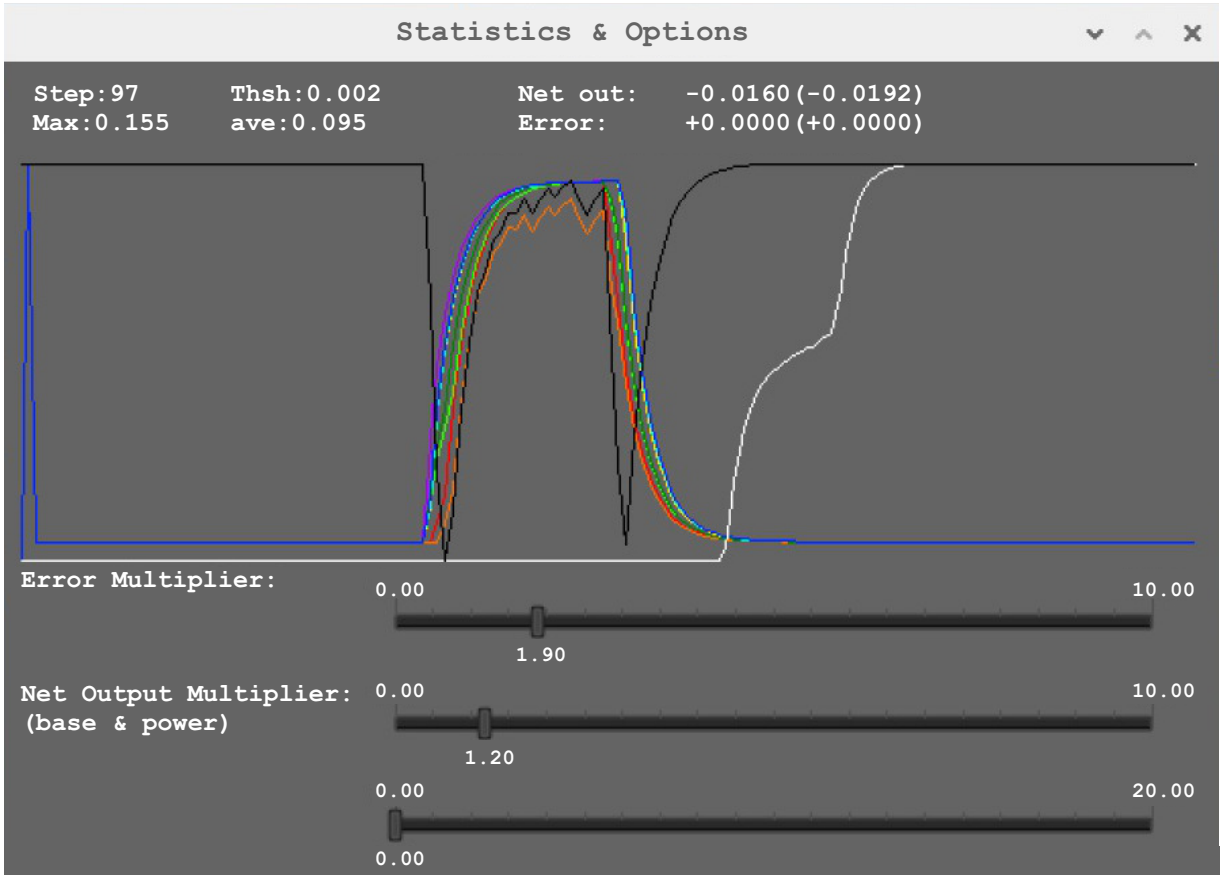

Figure 2.18: *Custom-designed GUI used for monitoring and manipulating physical parameters during experiments. The interface allows adjustment of the error multiplier and net output multiplier to achieve stable trials across various learning rates. Displayed signals include sensor outputs (coloured traces), error signal (black trace), and integral error (white trace). The top section shows the number of learning iterations, maximum and average values of the error integral, threshold for successful learning, and the raw and final values of error and network output after gain multiplication.*

2.7.3.2 Sensors' Field of View (FoV)

Each sensor receives reflected light from a small section of the canvas positioned directly beneath it. We term this area the FoV of the sensor, depicted by a dotted circle encompassing sensor S_1^* in Figure 2.16B.

2.7.3.3 Grey-Scale Value (GSV)

During navigation, these sensors translate changes in reflected light intensity into voltage fluctuations, which are sampled by the 8-bit analog to digital converters (ADCs) on the Arduino Nano ($2^8 = 256$). As a sensor's FoV transitions from fully capturing the black path to capturing only the white background, it generates a voltage potential (V_i) within the range of $\Delta V = (600 - 1500)[mV]$. We determine the grey value (G) of each sensor by

linearly mapping its voltage potential to a range of $[0, 256) \in \mathbb{N}$. This range signifies the gray-scale value (GSV) of the sensor's FoV.

$$G_i = (V_i - 600) \times \left(\frac{256}{\Delta V}\right) \tag{2.62}$$

2.7.3.4 Measure of Deviation

The G values of the sensors are directly proportional to the presence of the black path in their respective FoVs. This characteristic facilitates the determination of the vertical alignment of each individual sensor concerning the path, thereby serving as a measure of the robot's relative positioning. For example, in Figure 2.16B, sensor S_3^* is vertically aligned with the path, resulting in a low grey value value, closer to 0, while the other sensors exhibit high grey value values, closer to 256. The alignment of this sensor with the path indicates a significant overall deviation of the robot to the left. Conversely, the alignment of sensor S_1 would imply a slight overall deviation to the right.

2.7.3.5 The Error Signal (E)

From a technical perspective, the robot's deviation from the path is measured by a weighted sum of the discrepancies in G between symmetrical sensor pairs. Consequently, the experimentally obtained value of E, as defined earlier in Equation 2.20, can be represented as:

$$\mathbf{E} = \Sigma_{i=1}^{3} K_i (G_i - G_i^*) \quad [GSV] \tag{2.63}$$

The weighting factor, K_i, increases linearly with i to capture the extent of deviation. Thus, the further the active

sensor is from the centreline, the more pronounced the spike in E. This signifies a greater deviation, where a positive value implies deviation to the left, and a negative value implies deviation to the right.

2.7.4 Learning Components of the Robot

2.7.4.1 Camera Image

A camera provides predictive sensory inputs for the robot's learning process by capturing a 1280×720 pixel image of the surrounding environment. Unlike the light sensors, which are based on voltage readings, the camera directly provides the GSV of the pixels within

the range of $[0, 256) \in \mathbb{N}$. This information enables the robot to anticipate steering by providing insight into the path in the near distance.

To alleviate computational demands, the camera image is segmented into square regions, as depicted in Figure 2.16B. Each region is assigned the average GSV of the pixels it encompasses, thereby generating a sensory input from the camera in the form of an 8×12 matrix denoted as $[I]_{mn}$. Similar to the light array sensors, the differences between symmetrical entries in this matrix quantify the expected degree of deviation in the near distance:

$$C_{ij} = I_{ij} - I_{ij^*} \tag{2.64}$$
$$1 \leq i \leq m, \quad 1 \leq j \leq \lfloor \frac{n}{2} \rfloor \quad \textit{and} \quad j^* + j = n + 1$$

The 8×6 matrix formed by the camera signals C_{ij} contains information about forthcoming turns. A non-zero value of C_{ij} indicates a turn, with j representing the sharpness of the bend, i representing the distance of the turn from the current position, and the sign of C_{ij} indicating whether the turn is to the right or left. These matricised camera signals are then fed into the network.

2.7.4.2 Filter Bank (FB) & Predictors

To achieve an optimal correlation with the error signal during learning (Daryanavard and Porr, 2020a), each camera signal undergoes a delay via a FB of 5 finite impulse response (FIR) filters represented by F_h.

$$P_k = F_h * C_{ij} \tag{2.65}$$
$$\textit{where} \quad 1 \leq h \leq 5, \quad 1 \leq i \leq m, \quad 1 \leq j \leq \lfloor \frac{n}{2} \rfloor$$

This produces a sequence of 240 predictor signals denoted as P_k, which are then supplied to the learner. It is worth noting that the predictors do not convey any explicit information to the learner; they are essentially the camera view but grouped together and filtered. However, the learner deduces the relevant information from these signals, which is then used to turn right or left with the correct timing and intensity. This process results in the generation of the anticipatory action of the learner, which is integrated into the overall motor command sent to the robot, as described below.

2.7.5 Motor Command (MC)

The robot manoeuvres by modifying the speeds of the right and left wheels, labelled as V_R and V_L respectively. These wheels typically rotate at a constant speed of $V_0 = 5[\frac{\text{cm}}{\text{s}}]$ when idle, and their velocities are adjusted through a motor command (MC) sent to them for modulation. Precisely, the MC is defined as:

$$\begin{cases} V_R = V_0 + MC, & \text{for right wheel.} \\ V_L = V_0 - MC, & \text{for left wheel.} \end{cases} \tag{2.66}$$

The MC is generated collaboratively by both the reflex mechanism and the learner. Therefore, it constitutes the sum of the reflex and predictive actions outlined in Section 2.5:

$$\text{MC} = A + A' \tag{2.67}$$

The proportionality constant of A is linked to E, whereas A' is calculated as a weighted sum of the activations in the output layer:

$$\begin{cases} A \propto E, & \text{reflex action.} \\ A' = f([A]^L) = [M]_{1,n} \cdot [A]^L_{n,1}, & \text{predictive action.} \end{cases} \tag{2.68}$$

Where $[M]$ signifies a weighting matrix applied to the activations of the output layer. This is a one-dimensional matrix that, when multiplied by the outputs of the network, results in a single scalar value that is sent to the motors for steering action. The use of this matrix allows for varying degrees of steering control for the robot, contingent on the active neuron and its associated weight factor. In particular, the weight factor can be tuned to enable sharp, moderate, or gradual steering adjustments.

2.7.6 Experimental Procedures

Throughout the experiments, the robot was consistently positioned at the same location on the canvas, ensuring reproducibility of the results. The process of data collection and learning initiated when any of the robot's sensors initially detected the path. Starting from this juncture, the robot commenced learning and adapting to follow the path, with the learning process vigilantly monitored via the GUI. Once the robot successfully generated the model of its environment, it would automatically deactivate the wheels and come to a halt, signifying the completion of the learning process. This methodology guaranteed the

reliability and consistency of the robot's learning process across all experiments.

2.7.6.1 Data Collection

The sampling rate for the experiment was established at $33Hz$ on the Arduino Nano. The data collected from all experiments encompassed several crucial parameters for the analysis of the robot's learning process. These factors comprised the error signal E, error's moving average $\bar{E}$ over a fixed number of steps n as in Equation 2.69, absolute error integral $\langle E \rangle$ over the entire duration of the trial N, see Equation 2.70, predictors GSV readings I', predictive motor command A', network's outputs P_i, individual neuron's weight changes ω_{ij}^{ℓ}, final weight distribution of the first layer, and the (x, y) coordinates of the robot to plot its trajectory.

$$\text{Error average: } \bar{E} = \frac{1}{n} \Sigma_{t}^{t-n} E(t) \tag{2.69}$$

$$\text{Error integral: } \langle E \rangle = \Sigma_{t=0}^{N} |E(t)| \tag{2.70}$$

From the gathered data, we derived an additional primary factor: the time required to attain successful learning. These metrics were employed to compare the various learning modalities introduced in this study. Through this data analysis approach, we were able to gain valuable insights into the robot's learning process and the effectiveness of different learning strategies. The data collected and scrutinised in this investigation provided a comprehensive understanding of the learning process, aiding in the identification of the most effective avenues for achieving successful learning.

In this work, the Euclidean distance of weights in each layer is used as an indicator of weight convergence stability. This is calculated as the multidimensional distance of the weight matrix $[\omega]$ at time $t^{'}$ from its initialisation matrix at time t_0:

$$\begin{aligned} Ed(t^{'}) &= Euclidean([\omega]\Big|_{t'}, [\omega]\Big|_{init}) \\ &= \sqrt[2]{\Sigma_{i,j=0}^{I,J}(\omega_{ij}^{\ell}\Big|_{t'} - \omega_{ij}^{\ell}\Big|_{t_0})^2} \end{aligned} \tag{2.71}$$

This parameter is calculated within individual layers of the network where ℓ is constant.

2.7.6.2 Condition for Successful Learning

In this study, the criterion for the success of the robot's learning process was established as the point at which the error signal fell below 1 % of its maximum value and sustained that state for a minimum of 100 time steps.

$$t_S = T \quad \text{if} \quad \langle E \rangle < max(E)/100 \Big|_{T}^{T-100} \quad \text{and} \quad T > 100 \tag{2.72}$$

To avoid erroneous identification of the success condition, the program initiated the evaluation process after the initial 100 steps had lapsed, thereby allowing for the accumulation of errors. This methodology ensured precise detection of the success condition, guaranteeing that the learning process had fully concluded before being deemed successful. By defining the success condition in this manner, we achieved a precise assessment of the efficacy of diverse learning modalities and pinpointed the most effective approaches for attaining successful learning outcomes. This approach to setting the success condition upheld the reliability and accuracy of the results obtained throughout this thesis.

2.7.6.3 Network Architecture

In the following chapters, we present the results of comparing various learning algorithms. All experiments in this study utilised a fully connected feedforward neural network. In all experiments, the networks are initialised with random weights and do not use any bias terms. This design choice stems from the fact that the network is intended to receive predictive inputs and generate motor commands, both of which are DC-free difference signals. Additionally, the control error used to train the network is also DC-free by definition.

Different network architectures were explored. Figure 2.19 illustrates the architecture of the network used in this study. It depicts FIR filter banks before the input layer, consisting of five low-pass filters acting on each input. This feature is consistent across all experiments. Additionally, the output layer consists of three neurons, and their weighted sum of activations produces the predictive action. This facilitates slow, moderate, and fast steering.

However, the network's shape can vary from experiment to experiment, depending on the number of layers and the number of neurons in each layer. The number of predictor signals can also vary between experiments, influencing the number of neurons in the first layer. At the start of each results section, we will specify these varying parameters.

2.8 Results

2.8.1 Successful Replica of ICO Learner

As described in preceding sections, the replication of the ICO learner served as an initial endeavour within this project. The experiments pertaining to the ICO learner were conducted using the simulation program introduced above. The robot was designed with two

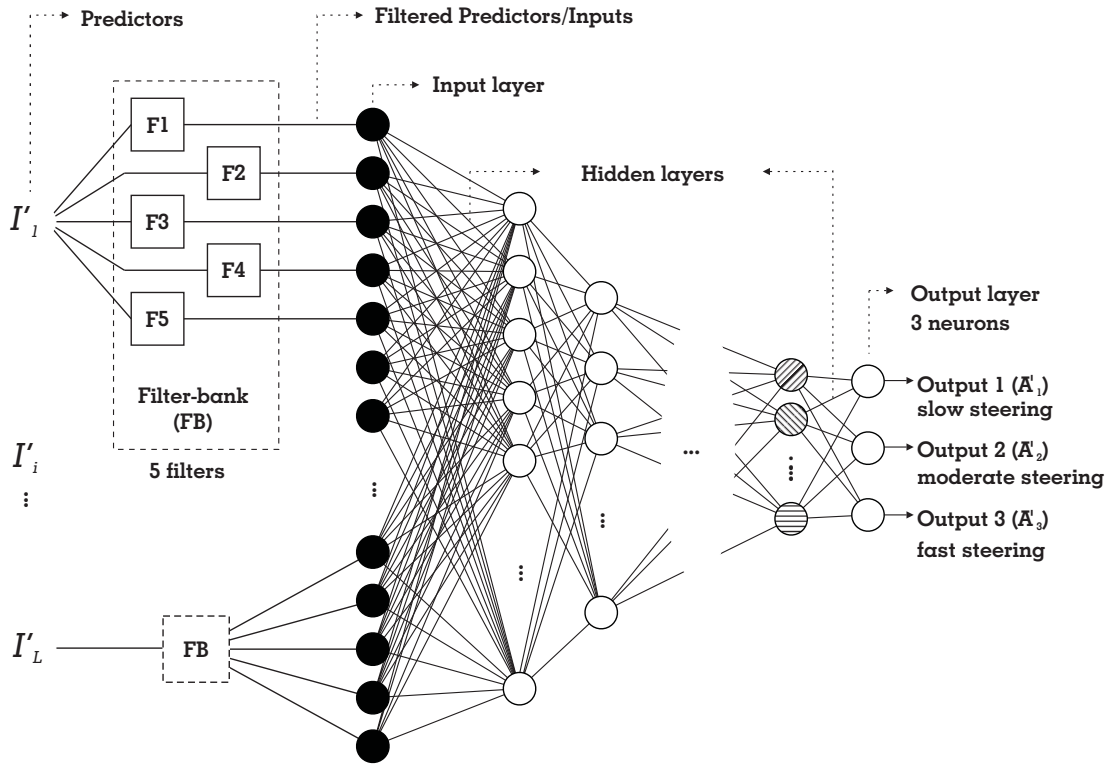

Figure 2.19: *Architecture of the neural network used across all experiments: a feedforward network composed of fully connected layers. This figure illustrates the filtering stage of the predictors with a filter-bank (FB), resulting in time-delayed inputs to the network. This aspect is consistent across all experiments. There are three neurons in the output layer, which remains constant across all experiments. Nonetheless, the number of predictors, hidden layers and the number of neurons in those layers can vary for each experiment.*

predictors, and, in alignment with the architecture of the ICO learner, one neuron was assigned to each predictor. The readings from each predictor were filtered by a low-pass FIR filter.

2.8.1.1 Error Minimisation

Figures 2.20A and B illustrate the GSV readings for the two predictors depicted as solid lines, accompanied by their filtered signals presented as dashed lines. Figure 2.20C showcases the error signal during a reflex trial as a grey solid line and the error signal during a learning trial involving the ICO learner as black solid lines.

These results clearly show that in the absence of learning, the reflex error persists throughout the trial duration. However, in another trial with the learning mechanism engaged, the error signal diminishes to zero at approximately $50[s]$, with only two minor pulses occurring later at $200[s]$ and $320[s]$. This decrease in error is due to the actions of the predictors shown in panels A and B, which supply input to the ICO learner, thus

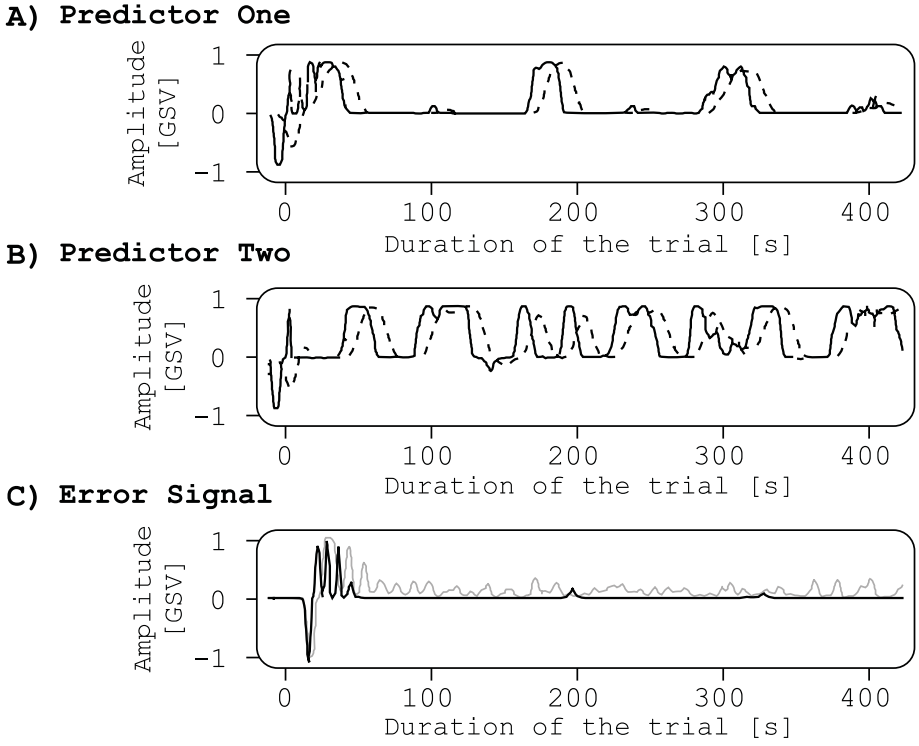

Figure 2.20: *Results from simulated experiments with the ICO learner.* A, B) *Displaying the activity of the two predictors along with their filtered signals using a low-pass FIR filter.* C) *Depicting the error signal during a reflex trial in grey, and during a learning trial in black traces. The y-axes represent the activities in GSV, while the x-axes represent time in seconds.*

generating the steering command. These predictor signals correlate with the error signal and drive the weight changes essential for learning.

2.8.1.2 Weight Changes

As reiterated earlier, the ICO learner was structured with two inputs. The weights associated with these inputs are visualised in Figure 2.21. Notably, the weight changes depicted in panels A and B mirror the behaviour of the corresponding predictor inputs showcased in Figures 2.20A and B, respectively.

This concludes the simulated experiments involving the ICO learner, which serves as a reference point for evaluating the CLDL algorithm, explored in subsequent sections.

2.8.2 Initial CLDL Algorithm (1.0.0) versus ICO Learner

The following findings in this section refer to result (a) in Figure 1.1 in the preface. Moving from the preliminary exercise, we proceeded to devise the initial CLDL algorithm. To facilitate a meaningful comparison, we executed a simulated trial employing an identical robot configuration as the ICO trial. This entailed having two predictors, each filtered by

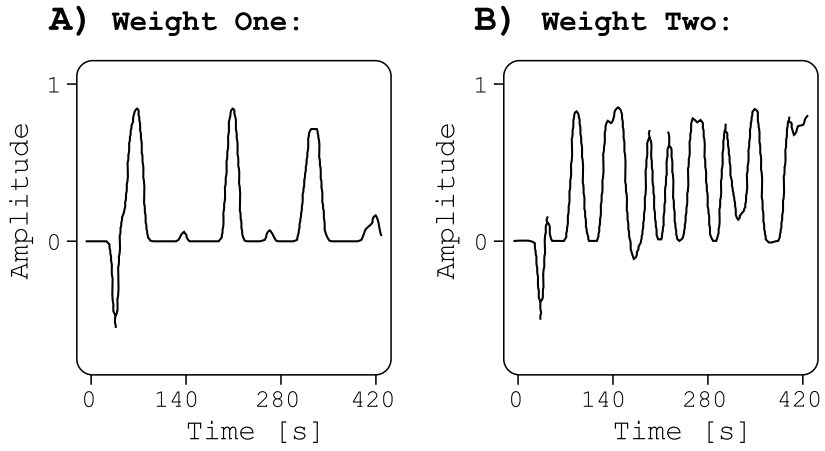

Figure 2.21: *Weight changes in a simulated trial with the ICO learner.* A) *Depicts the alterations in the weight linked to the first predictor. The y-axis denotes the relative amplitude, and the x-axis corresponds to time in seconds.* B) *Illustrates the results for the second weight associated with the second predictor.*

the FIR filters with the same characteristics. The neural network was structured with a simple architecture, comprising two layers: an input layer featuring two neurons to receive the predictors and an output layer encompassing a single neuron responsible for generating the motor command.

2.8.2.1 Error Minimisation

The behaviour of the two aforementioned predictors is illustrated in Figures 2.22A and B. Panel C further showcases the error signal during the reflex trial (grey line) overlaid with the error signal during the CLDL learning trial. It is evident that in the learning trial, the error has been significantly reduced in contrast to the reflex trial.

Furthermore, comparing these outcomes with the ICO learner's results underscores a distinct enhancement in the error profile. Primarily, the initial spikes within the first 50 seconds exhibit reduced magnitudes and occur less frequently. Secondly, the subsequent pulses seen at 200 seconds and 320 seconds are notably more negligible in comparison to those of the ICO learner.

2.8.2.2 Weight Changes

The weights of the neurons were initialised to 1 before the learning trial. Figure 2.23 illustrates the weight changes pertaining to the neurons linked with the predictors, as

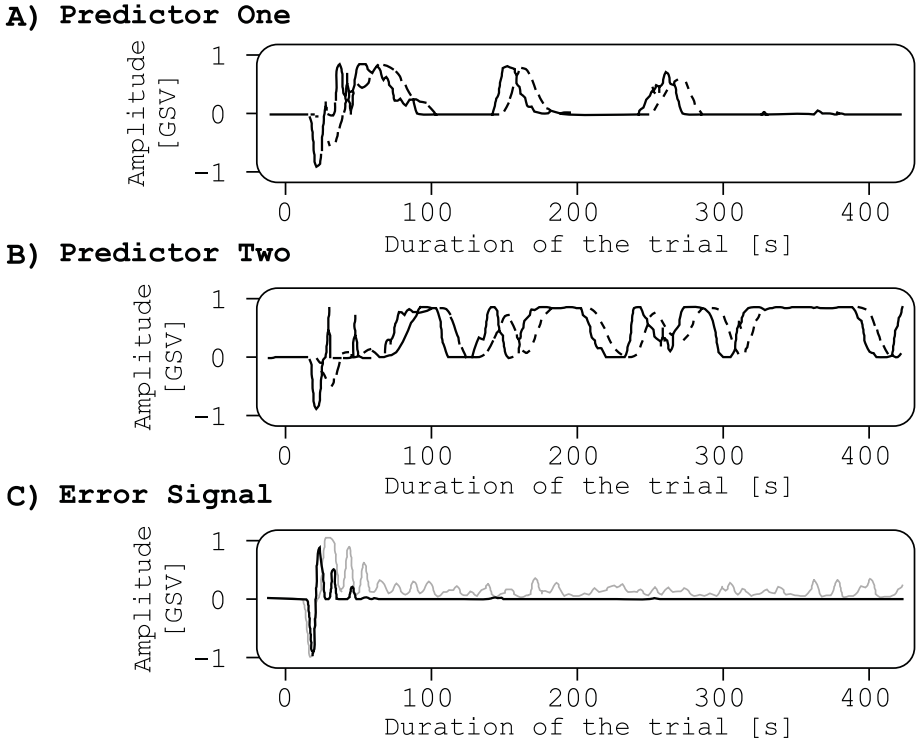

Figure 2.22: *Results from simulated experiments with CLDL.* A, B) *Displaying the activity of the two predictors along with their filtered signals using a low-pass FIR filter.* C) *Depicting the error signal during a reflex trial in grey, and during a learning trial in black traces. The y-axes represent the activities in GSV, while the x-axes represent time in seconds.*

observed in the learning trial. Notably, their alterations align with the behaviour of the predictors in Figures 2.22A and B, as anticipated.

This concludes the initial experiments involving CLDL, and its comparison with the ICO learner. The subsequent stages involve the expansion of CLDL to encompass additional layers and neurons, handle a greater number of predictors, and eventually extend to deployment on a physical robot. The forthcoming sections delineate the progressive evolution of CLDL into a robust and comprehensive algorithm.

2.8.3 Finalised CLDL Algorithm (2.0.0)

The following findings in this section refer to result (b) in Figure 1.1 in the preface. The results presented in this section are published in our journal paper Daryanavard and Porr (2020a). This section is divided into two segments. Initially, we present the outcomes derived from simulations. Subsequently, we unveil the results from experiments conducted on the physical robot.

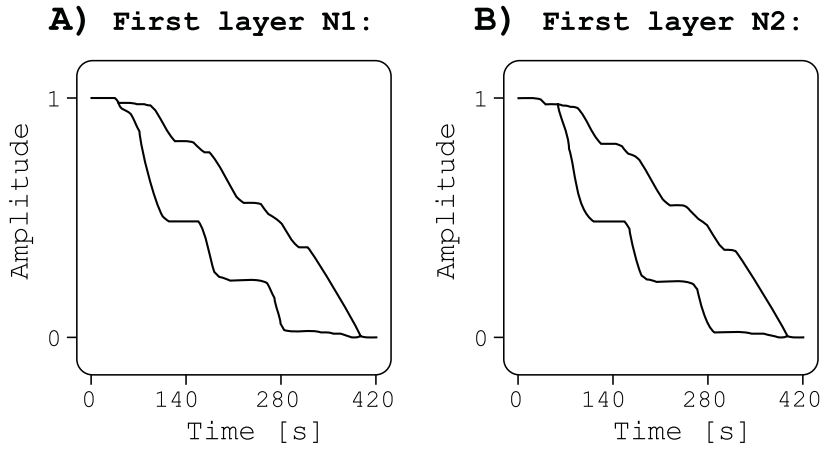

Figure 2.23: *Weight changes in a simulated trial with CLDL.* A) *Depicts the alterations in the weight linked to the first predictor. The y-axis denotes the relative amplitude, and the x-axis corresponds to time in seconds.* B) *Illustrates the results for the second weight associated with the second predictor.*

2.8.3.1 Experiments in Simulation

Compared to the prior findings, the simulation environment underwent enhancements, evolving into a more intricate configuration. In this advanced setup, the robot was equipped with multiple predictors, and the neural network was expanded to encompass four layers, comprising 40 neurons in the input layer, 12 neurons in the first hidden layer, 6 neurons in the second hidden layer, and a single neuron in the final layer. This leads to a collective count of 59 neurons.

2.8.3.1.1 Error Minimisation Figure 2.24A exhibits the error signal throughout a reflex trial. In this trial, the robot engages in navigation without the help of learning, essentially setting the learning rate to zero ($\eta = 0$). This trial serves as a reference point for evaluating the learner's performance, providing a visual basis for subsequent comparisons.

The reflex error signal is observed to persist continuously throughout the simulation, as depicted in Figure 2.24A. The error's sign changes every time the robot alters its direction, and the error signal only reaches zero when transitioning from negative to positive. This evidences that the reflex controller fails to maintain a constant zero error.

In contrast, Panel B visually represents the error signal observed during a learning simulation utilising a learning rate of $\eta = 10^{-2}$. Evidently, the error signal experiences a

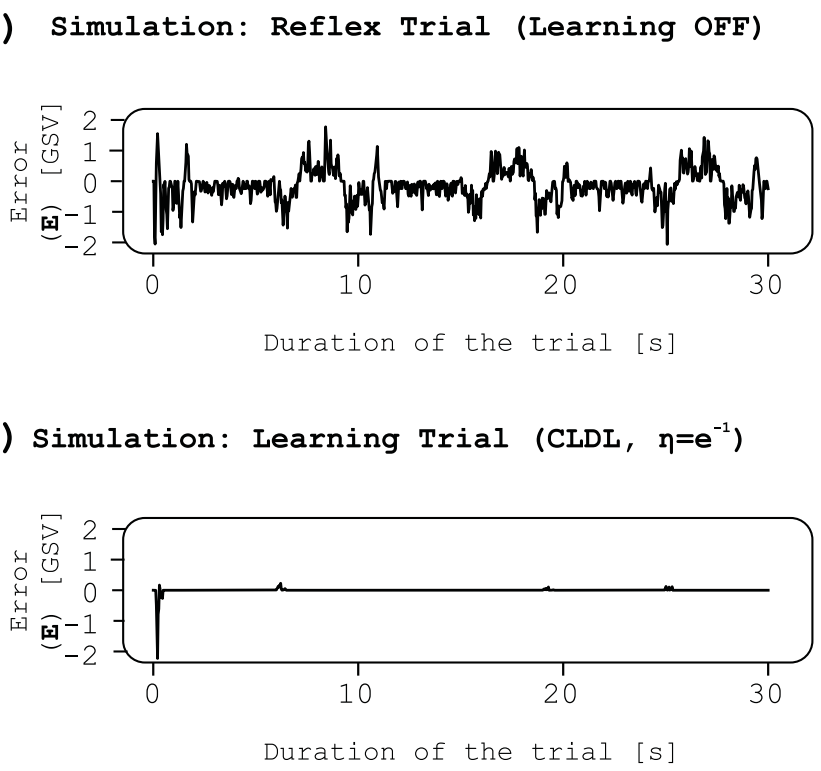

Figure 2.24: *Illustrates the error signal for CLDL experiments conducted within the simulation environment.* A) *Depicts the error during a trial with the reflex mechanism solely, wherein learning is inactive.* B) *Displays the error for a learning trial. The y-axis represents the error magnitude in GSV, while the x-axis signifies time in seconds.*

notable reduction. The learning process occurs rapidly, with only an initial spike and three subsequent pulses at cross-overs of the path, converging within approximately 2 seconds.

2.8.3.1.2 Euclidean Weight Distance & Convergence Figure 2.25 showcases the Euclidean distance between the weights and their initial random values during the learning trial. The graph visually depicts a progressive rise from zero, gradually converging to approximately 1. Given that the error signal is disseminated as a weighted sum of internal errors, all layers display similar behaviour in their weight adjustments. As predicted, the timing of weight modifications corresponds to instances when the error signal registers as non-zero.

2.8.3.1.3 Input Layer Weight Distribution Figure 2.26 presents the weight distribution within the first layer, accentuating the relative significance of each input as dictated by their respective weights, ranging from larger to smaller values. Each block relates to the cluster of filtered signals originating from each predictor.

This weight distribution exhibits a discernible pattern where the outermost predictor

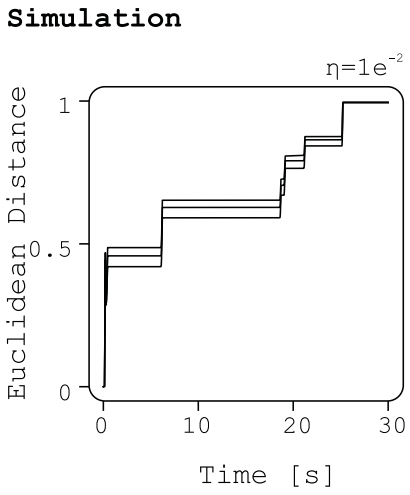

Figure 2.25: *Depicts the Euclidean weight distance within each layer of the neural network during a learning trial with CLDL in the simulation environment. The y-axis represents the relative amplitude, while the x-axis corresponds to time in seconds.*

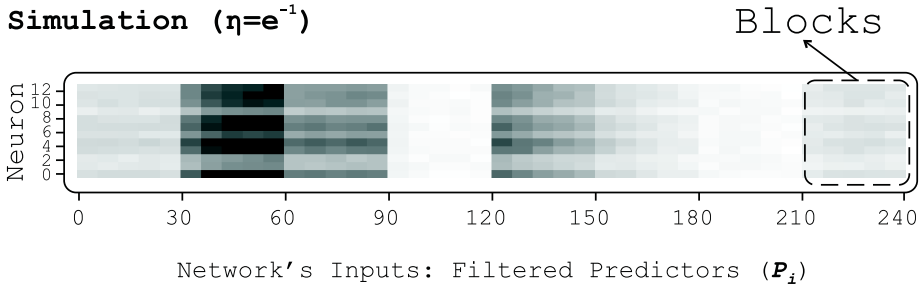

Figure 2.26: *Illustrates the weight distribution within the first layer of the neural network. The y-axis represents the index of neurons in this layer, while the x-axis signifies the index of inputs (the filtered predictors) to this layer. The final weight values are colour-mapped using a greyscale, where black signifies the highest value and white represents the lowest value. Each block corresponds to one predictor input.*

locations bear larger weights (darker shading), while the innermost locations are assigned smaller weights (lighter shading). The spatial configuration of each predictor in relation to the robot is shown in Figure 2.27. Another noteworthy observation relates to the predictor rows situated closer to the robot, manifesting a more pronounced gradient compared to those situated further ahead. Both observations align with expectations, as the robot generates a more robust output for sharper steering when the detected path veers farther from the centre but remains in proximity to the robot. Conversely, the robot generates a smaller output for milder steering if the path only slightly diverges from the centre with the bend appearing in the distance.

2.8.3.1.4 Robot Tracking To provide an enhanced visual contrast between the reflex and learning trials, the tracking data is graphically presented in Figure 2.28. These figures

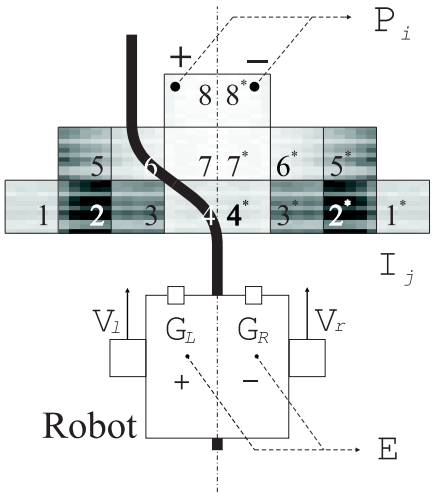

Figure 2.27: *Illustration of the virtual robot and its surrounding environment: The robot consists of a body equipped with two wheels, each with speeds denoted as V_r and V_l, and two ground sensors referred to as G_r and G_l. These ground sensors are responsible for generating the closed-loop error E. The robot is positioned on a track and possesses forward vision facilitated by 16 symmetrical ground light sensors labelled as I_j. These sensors provide data used to calculate the predictors denoted as P_i which are filtered and fed into the neural network.*

show the trajectory of the robot during both a reflex trial (Panel A) and a learning trial (Panel B). The visual representation underscores that the presence of learning imparts an anticipatory characteristic to the steering, resulting in a smoother trajectory. Conversely, in the absence of learning, the steering response is reactive and triggers abrupt changes.

2.8.3.1.5 Statistics & Reproducibility As demonstrated in Figure 2.24, the learning algorithm has the capacity to markedly diminish the error. An accurate gauge of this reduction is the average absolute error. To ascertain the reproducibility of the outcomes of the trial portrayed in that figure, supplementary simulations were conducted. The average absolute error magnitude was recorded across simulation runs employing a spectrum of learning rates $\eta : \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, with each instance repeated 10 times.

Figure 2.29 depicts the outcome of these simulation runs. It demonstrates that higher learning rates enable the robot to minimise the error more effectively.

2.8.3.2 Real-world Physical Experiments

This section provides the outcomes of experiments conducted using the physical robot. As before, we proceed to compare the performance of the CLDL algorithm with that of the reflex mechanism, which functions as a benchmark for assessing all learning paradigms.

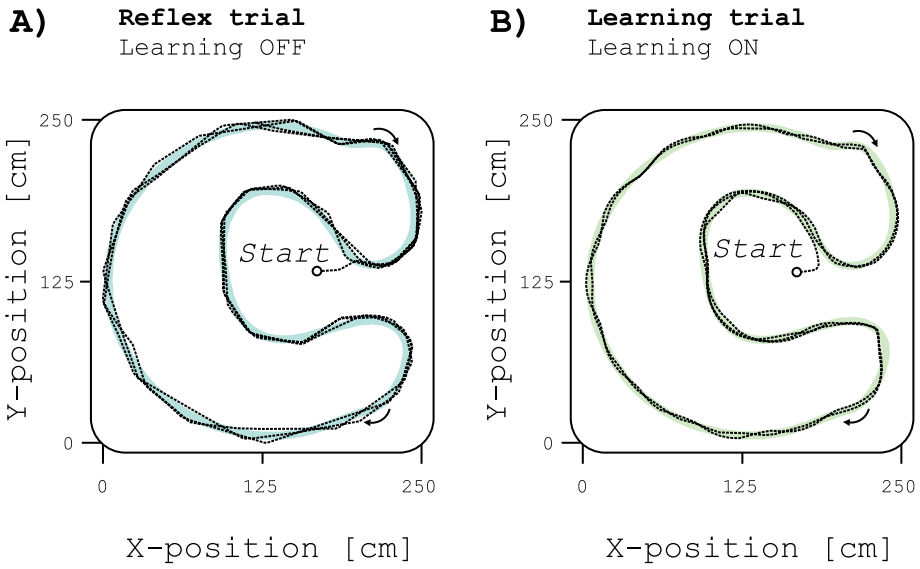

Figure 2.28: *Illustrates the trajectory of robot navigation during CLDL simulations.* A) *Depicts a trial with reflex, wherein learning is inactive.* B) *Displays a trial with active learning. The axes depict the x and y coordinates of the robot in centimetres. In both instances, the initial robot position is indicated, and the navigation direction is represented by arrows.*

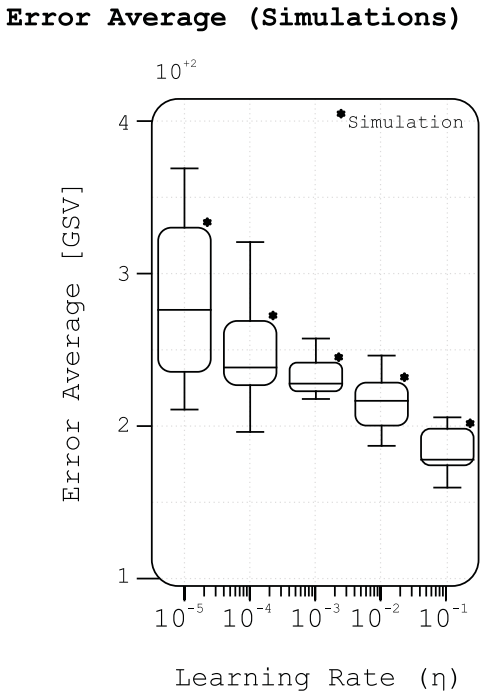

Figure 2.29: *Displays the reproducibility and statistical analysis of results for CLDL simulations. The graph showcases the average error for learning rates* $\eta : \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$*, each repeated 10 times.*

2.8.3.2.1 Experimental Setup & Network Architecture The experimental setup and common aspects of the network are detailed in Section 2.7. The network used for these experiments consisted of 10 hidden layers, each containing 11 neurons, except for the last layer which had 3 neurons. The purpose of the last layer was to generate outputs for fast, moderate, and slow steering commands. This design allowed the robot to have different levels of steering response depending on its state.

The input to this DNN consisted of 48 predictor signals extracted from the camera view, as described in the section referenced (Section 2.7). These predictor signals were processed using an array of 5 low-pass FIR filters to form a total of 240 inputs to the network.

2.8.3.2.2 Error Minimisation: Trial with $\eta = 10^{-1}$ Figure 2.30 contrasts a reflex trial (Panel A) with a learning trial (Panel B) employing a learning rate of $\eta = 10^{-1}$. The graph illustrates both the error signal (solid lines) and its moving average (dashed lines) across a quarter of the path. As explained earlier, the aim is to diminish the error signal, ideally converging it to zero.

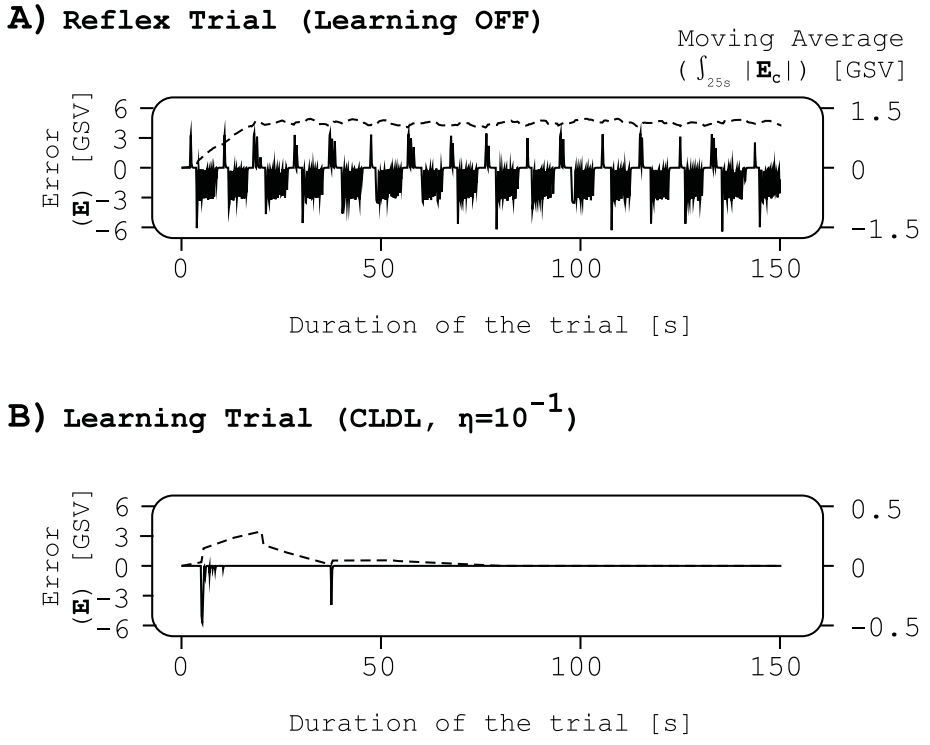

Figure 2.30: *Results of a trial conducted on a real robot using the CLDL algorithm with a learning rate of* 10^{-1}*. Both panels display the error signal as a solid line and its corresponding moving average as a dashed line. On the left side, the y-axis represents the magnitude of the error signals, while the right side corresponds to the moving average values. The x-axis represents time in seconds.* A) *shows these results for a reflex trial,* B) *shows these results for a learning trial.*

In Panel A, it is evident that the reflex error persists throughout the duration of the trial. The moving average of the error reaches its peak of approximately 1.5 GSV at around 20 seconds and maintains this peak until the trial's conclusion.

Moving to Panel B, the same trial is repeated, but with the CLDL learning mechanism activated. The error signal experiences a substantial reduction in both its persistence and magnitude. Initially, the error appears for a short period of less than 10 seconds before briefly re-emerging around 40 seconds. It then returns to zero and remains there for the remainder of the trial. The moving average of the error also illustrates this improvement, with its maximum remaining below 0.5 GSV.

These results showcase the rapid learning capabilities of the CLDL algorithm, as it successfully constructs a predictive model of the reflex mechanism in just 40 seconds.

2.8.3.2.3 Error Minimisation: Trial with $\eta = 10^{-3}$ For comparison, another trial with a slower learning rate is investigated.

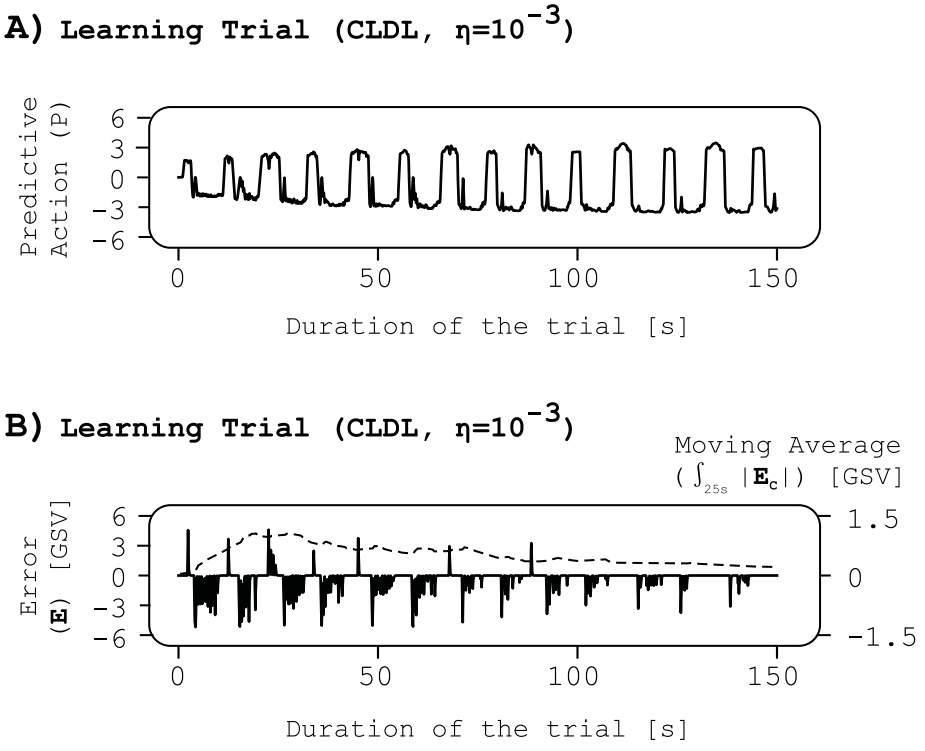

Figure 2.31: *Results of a trial conducted on a real robot using the CLDL algorithm with a learning rate of* 10^{-3}. A) *showcases the neural network's output (P),* B) *presents the error signal as a solid line and its corresponding moving average as a dashed line. On the left side, the y-axis pertains to the magnitude of the error signals, while the right side corresponds to the moving average values. The x-axis depicts time in seconds.*

In Panel A of Figure 2.31, the output of the network P is depicted for a trial at a learning rate of $\eta = 10^{-3}$. Panel B shows both the error signal and its moving average

for this trial. A comparison of these results with the reflex error signal presented in Figure 2.30A highlights a gradual reduction in the error signal.

The formation of the motor command involves a combination of the signal P and the error signal, as described in Equation 2.68. Notably, as the output of the network P increases in magnitude, the error signal decreases, thereby generating an appropriate motor command at each point during the trial. This interaction between the network's output and the error signal contributes to the improvement in the error reduction observed in this trial.

With increasing learning rates, the behaviour and convergence of the neural network can vary widely depending on several factors: 1) Architecture, which includes the number of layers, neurons per layer, inputs, and outputs, 2) Configuration, which involves aspects like weight initialisation and the activation functions used, and 3) External Factors which can include the robot's speed and the camera view, among others. Neural networks of this size are highly non-linear and unpredictable. Despite this, there are observable trends and patterns in the collective results of this work. Generally, a higher learning rate leads to faster learning. However, beyond a certain threshold, increasing the learning rate further can result in no additional improvements or even cause instability in the network. This phenomenon is explored in later experiments involving SaR learning, refer to Section 3.5.7.

2.8.3.2.4 Euclidean Weight Distance & Convergence In this section, we delve into the analysis of weight changes for the trials discussed earlier. Figure 2.32 provides insight into the Euclidean distance of the weights from their initial random values. This distance is a metric used to quantify how much the weights have changed over time. Panels A and B display this analysis for the two trials presented above, with learning rates of $\eta = 10^{-1}$ and $\eta = 10^{-3}$, respectively.

In both cases, the weight changes occur at moments where the error signal is present, consistent with the error fluctuations observed in Figures 2.30B and 2.31B. This correspondence aligns with the learning rule, which stipulates that weight adjustments are made when the error signal is non-zero, allowing the system to work towards reducing the error. If the error signal is zero, there is no need for further weight changes.

Figure 2.32A demonstrates the weight changes for a higher learning rate of 10^{-1}. This leads to more abrupt and significant weight changes, resulting in an overall greater weight change magnitude of 8×10^{-1}. Panel B depicts the weight changes for a slower learning rate of 10^{-3}, which results in an overall smaller weight change magnitude of 4×10^{-1}.

Comparing the two panels highlights a trade-off between weight stability and the time required for learning. Higher learning rates can lead to unstable behaviour and overfitting, while slower learning rates offer more stability at the cost of longer learning times

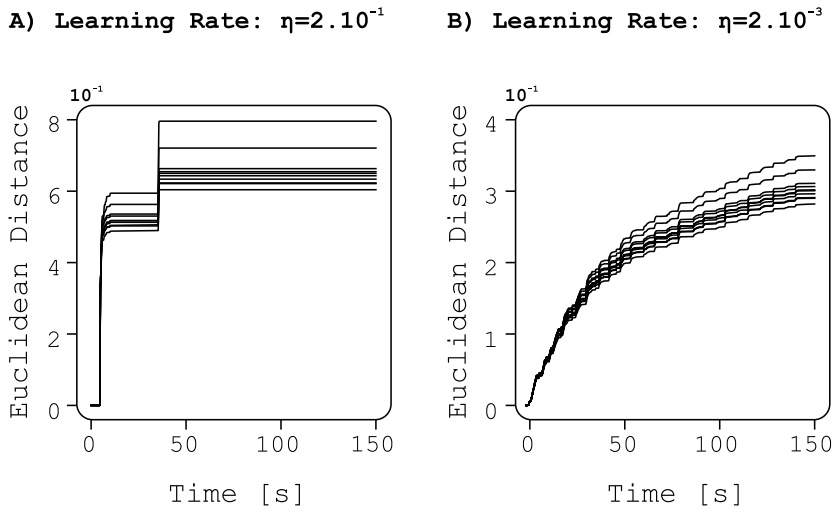

Figure 2.32: *Depicts the Euclidean weight distances for each layer of the network during trials with the CLDL algorithm on the real robot.* A) *displays these results for a trial with a high learning rate of* 10^{-1}, B) *shows these results for a trial with a slow learning rate of* 10^{-3}. *The y-axis represents the distance of the weights, and the x-axis represents time in seconds.*

Shalev-Shwartz and Ben-David (2014).

2.8.3.2.5 Input Layer Weight Distribution The first layer is of particular importance as it is the initial point of interaction with the sensory predictive inputs. This layer reveals the meaning and significance that the network learns to assign to each input through the assignment of different weights. The weight distributions in the first layer for the above trials are depicted in Figure 2.33.

The network assigns weights to different input-neuron connections based on the significance of each connection. In the weight distributions, the inputs generated from each row of predictors are filtered and organised into blocks, which are highlighted with dashed rectangles. The arrangement of these blocks closely follows the positioning of the predictors, as shown in Figure 2.16B.

In both cases, whether for a higher learning rate (Panel A) or a slower learning rate (Panel B), the weight distributions exhibit a pattern. The weights assigned to the filtered signals coming from the outermost column of predictors, such as $P_{30,60,\ldots,210,240}$, have higher values (appear darker in the greyscale representation). In contrast, weights assigned to the filtered signals from the innermost column of predictors, such as $P_{5,35,\ldots,185,215}$, have smaller values (appear lighter). This pattern aligns with the network's behaviour of generating more significant steering responses for greater deviations, which occur when the path is detected in the outer portion of the camera view (activating the outermost predictors). Conversely, the network generates subtler steering responses for smaller deviations, occurring when the path is closer to the middle of the camera view (activating the innermost

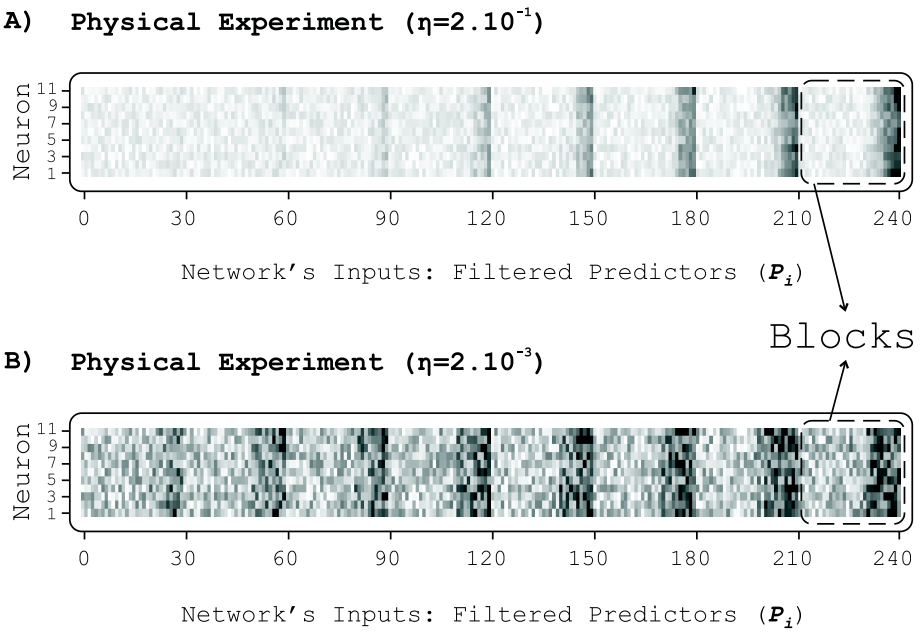

Figure 2.33: *Illustrates the weight distributions within the first layer of the neural network for trials with CLDL on a physical robot. The y-axis represents the index of neurons, while the x-axis signifies the index of inputs (the filtered predictors). The final weights are colour-mapped using a greyscale, where black signifies the highest value and white represents the lowest value. Each block corresponds to one predictor input.* A) *the weight distribution is shown for a trial with a higher learning rate of* 10^{-1} B) *the weight distribution is shown for a trial with a slower learning rate of* 10^{-3}.

predictors).

Comparing Panels A and B reveals that the weight distribution is more distinct for a higher learning rate (Panel A). This indicates that with a higher learning rate, the network quickly adapts its weight distribution to assign more importance to relevant predictors, resulting in a more efficient learning process.

2.8.3.2.6 Robot Tracking In this section, we compare the performance of the Reflex algorithm and the CLDL algorithm by visualising their trajectories while the robot navigates the map. To achieve real-time position tracking, the OptiTrack IR motion capture system was utilised, which comprises 18 IR cameras and provides tracking precision down to the millimetre level OptiTrack (2019).

Figure 2.34A displays the trajectory of the robot during a Reflex trial, while Figure 2.34B depicts the trajectory during a Learning trial. The blue line represents the path for the Reflex trial, and the green line represents the path for the Learning trial (they are identical). In both cases, the actual trajectory of the robot is illustrated with solid black lines.

In Panel A, it is evident that when learning is disabled (Reflex trial), the path followed

by the Reflex algorithm frequently diverges from the blue path. Multiple crossing points, denoted by star symbols, are observed (eight crossings corresponding to the eight turns in the path). The trajectory traced by the Reflex algorithm seems to be a skewed version of the true path.

Conversely, in Panel B, with CLDL learning enabled (using a learning rate of $\eta = 2 \cdot 10^{-1}$), the robot's trajectory aligns closely with the desired path. Slight deviations at the top portion of the path represent the initial phase of the trial, where the learner is still acquiring information, and the error has a more pronounced impact on determining the motor commands.

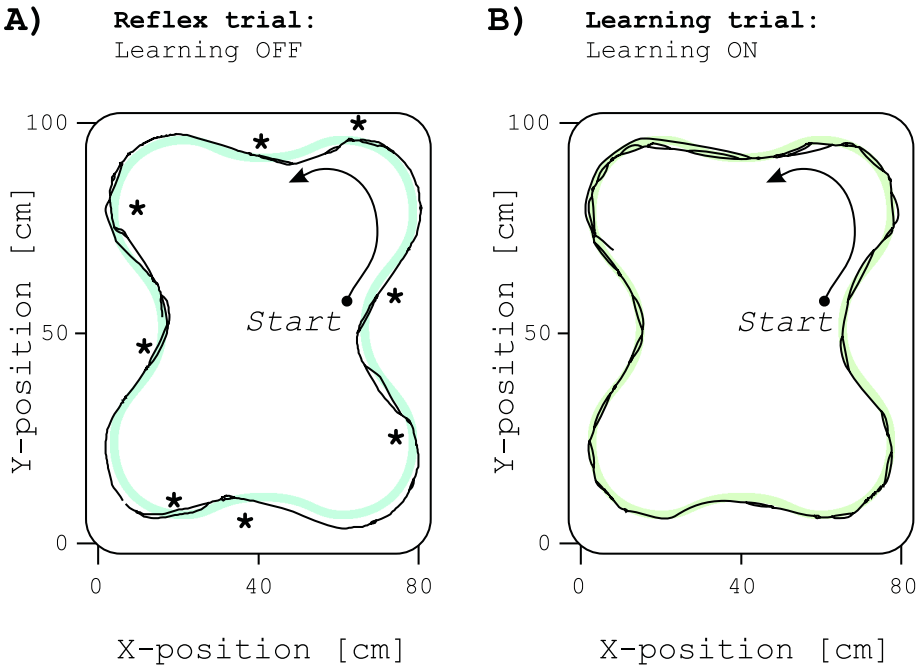

Figure 2.34: *Illustrates the trajectory of robot navigation during CLDL trials with the physical robot.* A) *Depicts a trial with reflex, wherein learning is inactive.* B) *Displays a trial with active learning with a learning rate of* $\eta = 2 \cdot 10^{-1}$. *The axes depict the x and y coordinates of the robot in centimetres. In both instances, the initial robot position is indicated, and the navigation direction is represented by the arrow.*

2.8.3.2.7 Statistics & Reproducibility Up to this point, we have presented results from three types of trials: a Reflex trial and two Learning trials conducted with high and low learning rates. To ensure the reliability of these outcomes, each trial was repeated five times.

For a comprehensive understanding of the impact of the learning rate on the robot's performance, we incorporated three additional learning rates into the analysis: $\eta = \{2 \cdot 10^{-3}, 2 \cdot 10^{-2.5}, 2 \cdot 10^{-2}, 2 \cdot 10^{-1.5}, 2 \cdot 10^{-1}\}$. In Figure 2.35A, we showcase the time required to achieve success in these various trials. The graph reveals an exponential decrease in

the time needed for success as the learning rate increases.

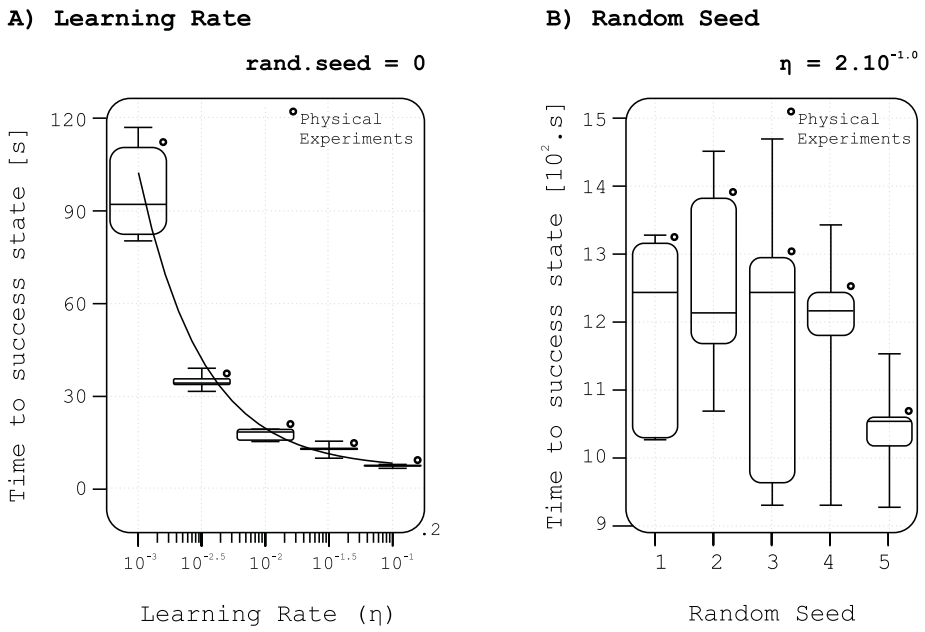

Figure 2.35: *Statistics and Reproducibility of CLDL results:* A) *shows this metric against learning rates of:* $\eta = \{2 \cdot 10^{-3}, 2 \cdot 10^{-2.5}, 2 \cdot 10^{-2}, 2 \cdot 10^{-1.5}, 2 \cdot 10^{-1}\}$. B) *Shows the effect of random seed in weight initialisation on the time taken to achieve successful convergence.*

The earlier sections demonstrated that weight changes are a robust indicator of internal stability and convergence within the network. Therefore, it is essential to explore the potential impact of random weight initialisation on the experimental outcomes. Figure 2.35B shows the influence of five different random seeds for weight initialisation on the time required to achieve success in trials with a fixed learning rate of $\eta = 2 \cdot 10^{-1}$. The depicted results indicate that there is no notable disparity in the data across different random seeds. This reinforces the credibility of the findings, as the time necessary to achieve success does not appear to be contingent on the specific random seed used for weight initialisation.

To conclude, the experimentation with the CLDL algorithm has demonstrated its capacity to minimise the error signal through its predictive capabilities. These findings have been published Daryanavard and Porr (2020a). By establishing CLDL as a benchmark, it can provide a foundation for evaluating forthcoming algorithms that will be introduced in subsequent chapters.

2.8.4 GPU Implementation with CUDA

Both the forward propagation of input for prediction and the backward propagation for training a neural network involve extensive calculations. Forward propagation necessitates activation calculations for each neuron, while back propagation demands initial error computation before updating each neuron's weights. Currently, the use of a single execution thread mandates sequential processing of these calculations. This causes longer processing times and notable delays, particularly in larger networks, which is problematic in closed-loop systems. However, transitioning to GPU implementation can mitigate these limitations.

Therefore, the CLDL algorithm was also developed in compute unified device architecture (CUDA) language to enable parallel programming and seamless implementation on a GPU (Porr, 2021). The CUDA, developed by NVIDIA, is a software framework designed to harness the capabilities of their graphics cards. Based on C++, CUDA enhances functionality with features for memory allocation, multi-threading, and executing code kernels on the GPU. The following results were achieved in collaboration with an undergraduate team.

2.8.4.1 Memory Allocation & Initialisation Time

The GPU, connected to the host system via the PCI bus, lacks direct access to the main computer's memory. For GPU execution, data must first be transferred from the CPU to the GPU space. Once processing is complete, the data typically needs to be moved back to the CPU for further processing. However, if multiple tasks on the same data are required, memory can be allocated within the GPU, similar to CPU memory allocation. This GPU memory space is referenced by a pointer, which can be passed to GPU kernels when invoked by the CPU (Sanders and Kandrot, 2010).

This process incurs a memory allocation overhead on the GPU. Figure 2.36 illustrates the relationship between the increasing number of layers (bottom axis) and neurons (top axis) in the network, and the initialisation time required. The graph's green and blue traces represent the initialisation times on the GPU and CPU, respectively. It is observable that the CUDA memory allocation initialisation on the GPU takes longer with an increasing number of layers and neurons, compared to C++ memory allocation on a CPU.

2.8.4.2 Propagation in Runtime

Once the network is initialised with the required memory allocations, the duration of one iteration of forward and backward propagation is examined. Figure 2.37 displays this propagation time, with the GPU's performance in green and the CPU's in blue. As the number of layers and neurons increases, the CPU's time to complete an iteration rises

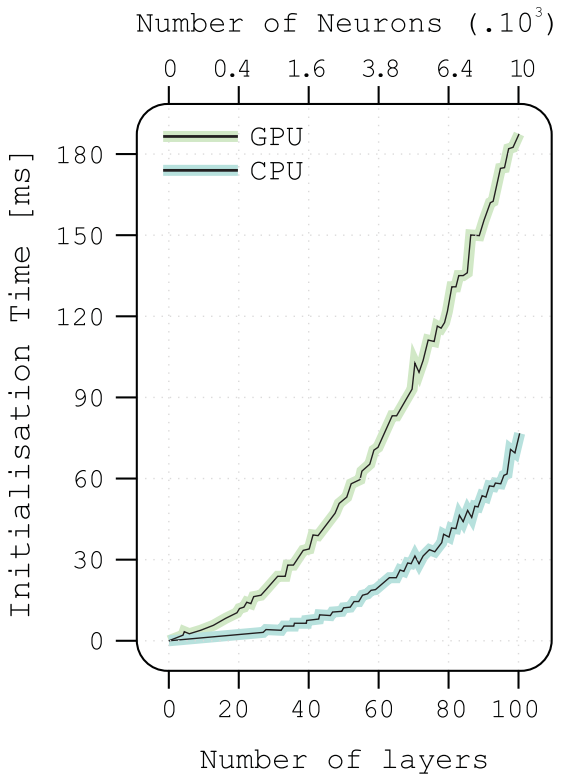

Figure 2.36: *Displays the initialisation time needed for memory allocations: The green trace indicates an increased overhead for GPU memory initialisation as the number of layers (bottom axis) and neurons (top axis) grows. The blue trace illustrates that memory allocation on a CPU is less time-consuming.*

exponentially. In contrast, the GPU's propagation time remains relatively stable, even with an expanding network. This highlights the efficiency of parallel programming and the effectiveness of harnessing GPU power.

2.8.4.3 Error Minimisation

Figure 2.38 illustrates the error minimisation performance of CLDL on both CPU and GPU. Panel A presents the benchmark error signal. Panel B demonstrates a learning trial on the CPU, while Panel C depicts a corresponding trial on the GPU, with identical specifications. As anticipated, both trials achieve equivalent results. The slight variations observed between the trials can be attributed to the different random initialisations of weights.

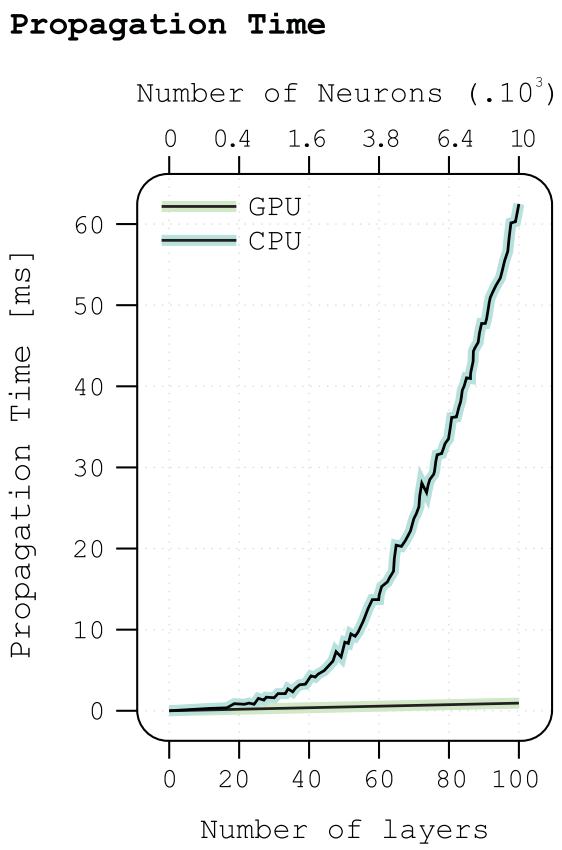

Figure 2.37: *Illustrates the time required for one iteration of network propagation: The blue trace represents CPU implementation, revealing that with an increasing number of layers (bottom axis) and neurons (top axis), the time for the CPU to complete one iteration grows exponentially. The green trace demonstrates that the GPU implementation remains largely unaffected by significant increases in network size.*

2.9 Discussion

Closed-loop deep learning (CLDL) utilises a multi-layered neural network, in contrast to earlier studies that often employed shallow neural networks with only a single layer. Shallow networks may not capture complex relationships or accurately predict reflex responses. Model-free RL, a different approach, involves agents learning to make decisions based on their interactions with the environment without explicitly modelling it, using deep learning to estimate the expected rewards associated with actions.

Our approach creates forward models, while model-free RL estimates expected rewards. Although both approaches use deep neural networks and back-propagation, they serve different purposes (Dolan and Dayan, 2013; Botvinick and Weinstein, 2014). Model-free RL focuses on learning optimal decision-making policies based on rewards, whereas our approach builds predictive models of reflexive responses. Integrating both approaches can lead to hierarchical RL, where the model-free RL component provides a prediction error signal for an actor, aiding in the development of forward models. This integration allows

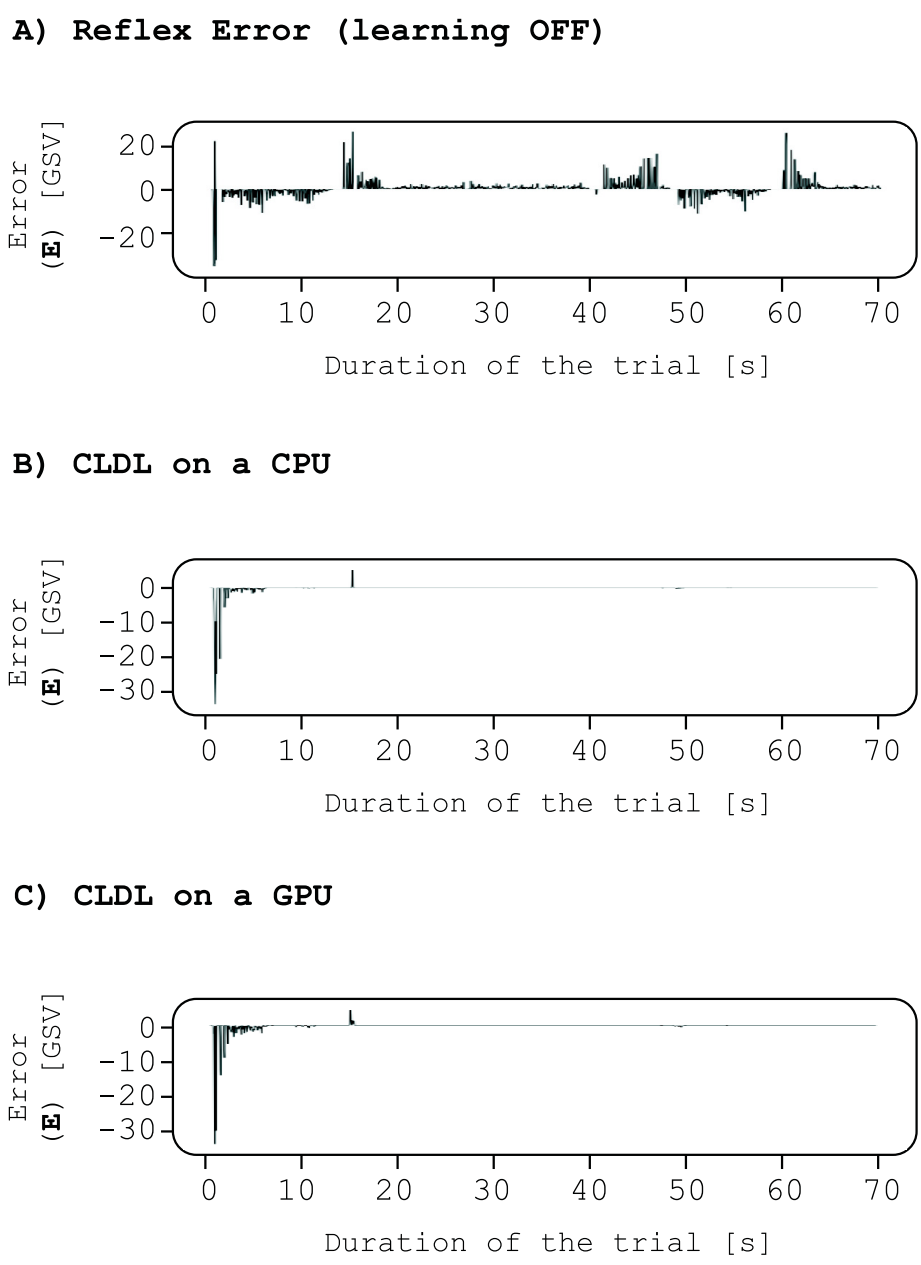

Figure 2.38: *Presents the error minimisation in the CLDL algorithm:* A) *displays the error signal during a reflex trial as a benchmark.* B) *illustrates error minimisation in a learning trial on a CPU,* C) *demonstrates error minimisation in a learning trial on a GPU, which shows the same learning outcome.*

model-free RL to guide the creation and refinement of forward models, enhancing learning.

Error back-propagation, used in training neural networks, is well-suited for open-loop scenarios where the system's actions do not depend on previous states or actions. In closed-loop systems, where actions affect subsequent states, recursion complicates back-propagation. We address this by adopting the z-domain mathematical framework, transforming recursive processes into simpler algebraic equations. This transformation allows the algorithm to handle closed-loop scenarios more effectively, making it suitable for sys-

tems with feedback loops.

Long short-term memory (LSTM) networks, a type of recurrent neural network (RNN) designed to handle sequences of data, can capture long-term dependencies (Schmidhuber et al., 1997). In closed-loop scenarios, LSTM networks unroll recursion and use back-propagation through time (BPTT) for weight calculation. Unlike typical offline training, our algorithm calculates network weights in real-time as the agent interacts with its environment, enabling online learning and adaptation.

Deep learning is often considered slow due to its iterative nature, adjusting parameters through back-propagation and gradient descent. Deep reinforcement learning (DRL), combining deep learning with reinforcement learning, can be even slower due to sparse discrete rewards, where feedback is occasional. Systems with continuous error feedback can learn more quickly, achieving nearly one-shot learning with rapid behavioural adjustments after few interactions. However, purely continuous systems may limit sophisticated planning and decision-making due to their reliance on simple reflex behaviours. Combining model-free DRL for complex tasks with model-based learning for efficient planning strikes a balance between slow and fast learning, allowing dual-system approaches for complex tasks and rapid adaptation.

Our novel approach utilises DNN architectures, designed to inherit the advantages of standard deep learning techniques, such as convolutional layers and high-level feature development. These features are crucial for extracting meaningful information from raw sensory data. Deep architectures automatically learn hierarchical data representations, enabling more sophisticated anticipatory actions in applications like line-following robots.

Forward models are essential in both robotic and biological motor control (Wolpert and Kawato, 1998; Kawato, 1999). They predict action consequences and ensure optimal trajectories. Our approach offers opportunities to learn complex forward models using deep networks, which can be combined with traditional Q-learning to improve movement planning. This combination enhances decision-making and action execution.

While our model-based approach is tailored to specific situations and may not generalise to different forward models, such as manipulating various objects, the modular selection and identification for control (MOSAIC) Model (Haruno et al., 2001) addresses this by learning multiple pairs of forward and inverse controllers, although this aspect is beyond our scope.

Chapter 3

Sign & Relevance (SaR) Learning

3.1 Introduction

In this section, we introduce the SaR learning paradigm. Building on the foundation of CLDL, a mathematically sound algorithm, we integrate concepts from neuroscience and biology to create a novel algorithm that better aligns with biological observations of brain learning. Following a brief motivation emphasising these neuroscientific concepts, we design and derive the SaR learning platform and its associated learning rule. Subsequently, we present the results and engage in a discussion.

3.2 Motivation

The development and learning processes of an organism are intimately tied to its interaction dynamics with its surroundings. Central to this understanding is the mechanism by which organisms gauge their environment. This is achieved through the cyclical process of receiving sensory inputs, initiating motor outputs, and consequently obtaining new sensory inputs (Maffei et al., 2017). This perpetual cycle of interaction is rooted in the concept of closed-loop learning (von Uexküll, 1926; Daryanavard and Porr, 2020a), where each action, based on its outcome, is categorised as either beneficial or detrimental. It is within this framework that the principles of reinforcement learning emerge (Dayan and Balleine, 2002).

The essence of reinforcement learning, particularly in a framework that mirrors biological systems, is the computation of the reward prediction error. In simple terms, it is an assessment of the discrepancy between expected and received rewards. During the 1990s, a hypothesis by Schultz et al. (1997) proposed that the neurotransmitter dopamine encodes this reward prediction error (Bromberg-Martin et al., 2010; Wood et al., 2017; Takahashi et al., 2017). This proposition drew intriguing parallels between the concept of temporal difference error, a cornerstone in machine learning algorithms, and the biological

mechanism of dopamine secretion (Sutton, 1988a).

From this perspective, an assumption emerged, suggesting that the brain's operational architecture could be likened to the actor-critic model. In this model, dopamine serves a dual function. Firstly, it acts as a messenger for the reward prediction error. Secondly, it instigates adaptive synaptic modifications in brain regions such as the striatum (Humphries et al., 2006). To delve deeper into the actor-critic paradigm, one could conceptualise it as a dual-layered closed-loop system. The innermost layer, or the reflex loop, generates an error signal. This signal then modulates the outer layer, or the actor, conditioning it to anticipate future actions. Essentially, the actor is sculpted into generating a predictive model of the reflex, ensuring a more calibrated response to future stimuli (Porr and Wörgötter, 2002b). As a result, the actor-critic structure is versatile, forming the foundation for both model-based (Verschure and Coolen, 1991) and model-free learning systems.

Utilising a global error signal, like the dopamine-based reward prediction error, certainly possesses intrinsic limitations. One of the key challenges with such an overarching error signal is its universal impact on all neurons. This broad application of an error signal can reduce the effectiveness and specificity of multi-layer neural networks, as the collective modulation could inadvertently constrain the differentiation and function that separate layers provide (Humphries et al., 2006; O'Reilly and Frank, 2006).

This limitation perhaps provides insight into the distinct architectural differences found within the brain. The striatum, for instance, is perceived as a single-layered structure that obtains its dopamine supply mainly from the substantia nigra pars compacta (SNc). Such a simplistic architecture might be well-suited to the overarching influence of a global error signal like dopamine.

Cortical networks, notably the orbitofrontal cortex (OFC) and the medial prefrontal cortex (mPFC), play pivotal roles in the intricate processes of reinforcement learning and decision-making. Their involvement is not merely a by-product of their cortical status but is underscored by their unique dopaminergic inputs. Unlike the striatum, these cortices receive their dopamine from the ventral tegmental area (VTA) (Haber et al., 1995; Berthoud, 2004; Rolls and Grabenhorst, 2008; Dela Cruz et al., 2016). This distinctive source and the distinct pathways underline the nuanced and multifaceted nature of the brain's learning and decision-making mechanisms, suggesting a need for more specific and tailored error signals for efficient neural function.

When delving into the intricacies of cortical pyramidal neurons, a central aspect to highlight is the primary mechanisms responsible for inducing neural plasticity. These mechanisms, which are essentially the brain's means of adapting and learning from experiences, are primarily governed by distinct learning rules. One of the earliest and most recognised of these rules is Hebbian learning, where the synaptic strength between neu-

rons is increased if both neurons are active simultaneously (Hebb, 1949b; Bliss and Lomo, 1973). Later, another form of synaptic plasticity, known as spike-timing-dependent plasticity (STDP), was introduced. In STDP, the change in synaptic strength is determined by the relative timing of spikes between the presynaptic and postsynaptic neurons (Markram et al., 1997).

Central to these processes is calcium. The dynamics of postsynaptic calcium concentrations play a decisive role in determining the nature of synaptic changes. Specifically, when the concentration of calcium is high, long-term potentiation (LTP) is induced, leading to increased synaptic strength. Conversely, when calcium levels are low, long-term depression (LTD) is triggered, decreasing synaptic strength (Lu et al., 2001; Castellani et al., 2001). This observation is crucial because calcium essentially dictates the direction, or the *sign*, of plasticity, determining whether the neuron undergoes LTP or LTD.

However, these learning rules do not operate in isolation. They are deeply intertwined with neuromodulators (Mattson et al., 2004; Lovinger, 2010), with serotonin being particularly influential (Roberts, 2011; Linley et al., 2013; Luo et al., 2015; Li et al., 2016; Crockett et al., 2009). Dopamine, another potent neuromodulator, also plays a role in synaptic plasticity (Dela Cruz et al., 2016). In a simplified conceptualisation, while calcium and the associated learning rules dictate the direction of synaptic change (potentiation or depression), neuromodulators like dopamine and serotonin can be viewed as adjusting the magnitude or the *learning rate* of this change.

Drawing from these biological insights, there have been theoretical propositions such as ISO3 learning. This learning framework is intriguing as it combines the concept of differential Hebbian learning with a rectified error signal termed as "relevance". However, similar to some of the brain's biological networks, the ISO3 architecture remains relatively simplistic, encompassing just a single layer.

This section presents a novel learning mechanism called SaR learning that expands upon the single-layer approach of ISO3 learning (Porr and Wörgötter, 2003) by incorporating multiple layers. SaR network utilises a top-down pass of the *sign* of an error signal to determine whether to implement LTP or LTD, while a global neuromodulator controls the learning speed.

3.3 Design & Derivation of SaR algorithm

3.3.1 Learning Platform

The learning platform for SaR is depicted in Figure 3.1. Both the reflex and learning loops, which were elaborated upon in Section 2.4, remain unchanged in this context.

Similar to the CLDL, the SaR algorithm constructs the forward model of the reflex by mapping its predictive actions to a set of sensory consequences perceived by the agent,

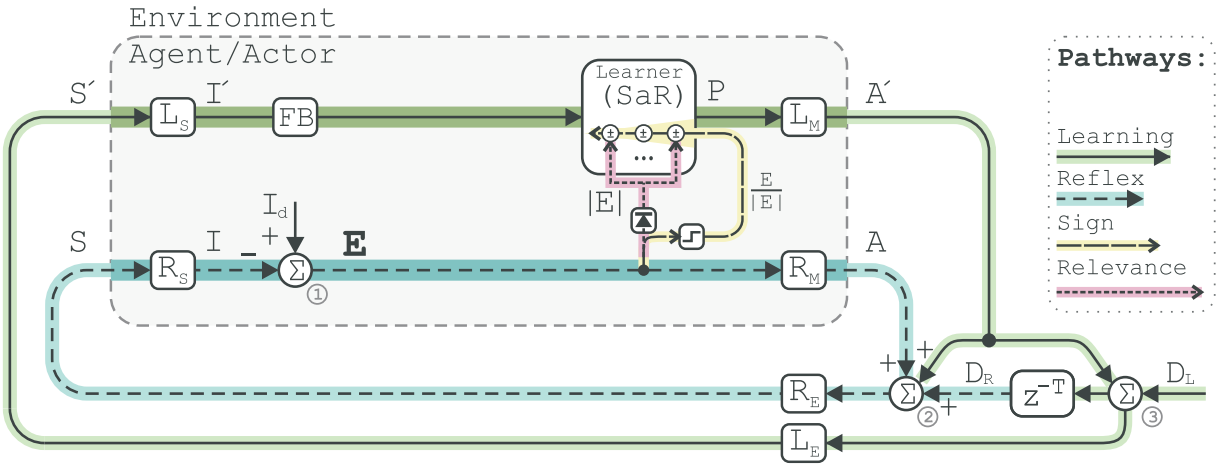

Figure 3.1: *Illustrates the integration of the SaR learner with the learning platform. The reflex and the learning loops operate as explained before. The yellow pathway shows the Sign signal that is fed into the network at the output. The pink pathways show the Relevance signal that enters every layer of the network for local propagation.*

which form the error signal.

A distinctive feature of the SaR paradigm is its provision of instructive feedback on the error through two separate pathways: 1) the bottom-up transmission of the *sign* of the error (highlighted in yellow), and 2) the global intervention involving the rectified error (highlighted in pink). Please see Figure 3.3 for a detailed illustration.

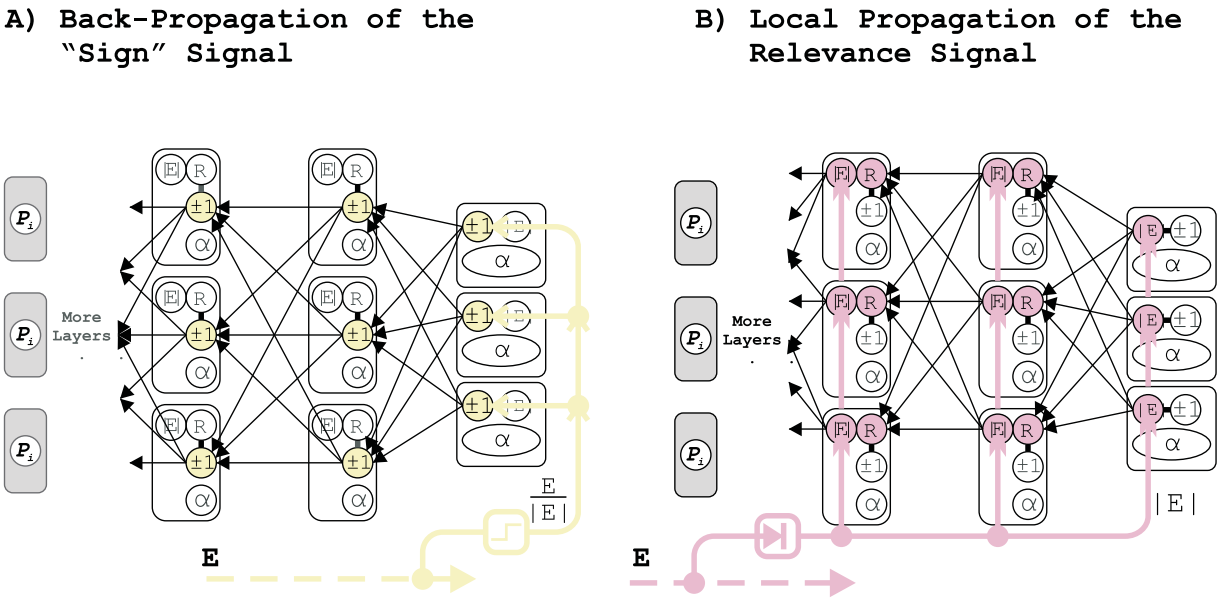

Figure 3.2: *Signal pathways within SaR network:* A) *Shows the back-propagation of the Sign signal through the entire network.* B) *Illustrates the local propagation of the closed-loop error from each layer to the adjacent layer, resulting in the Relevance signal. P_i represent the predictive inputs in the input layer and α represents the activation of each neuron.*

Figure 3.2 provides a detailed illustration of these pathways. In Panel A, we observe the bottom-up pass of the error's sign, highlighted in yellow. These traces signify the transmission of error signals from the reflex loop to the output neurons. The error's sign cascades from the final layer down to the deeper layers. Within each layer, the sign of the resulting value is further conveyed to the deeper layers, resulting in an error value of either ± 1 or 0 within each neuron. This, in turn, primes their connections for strengthening $(+1)$ or weakening (-1), analogous to the processes of LTP and LTD in the context of neurophysiology.

In Panel B of this figure, we observe the local pass of the error's modulus, highlighted in pink. These traces represent the entry of this signal from the closed-loop platform into every layer. Each neuron receives this value and transmits it to only one adjacent deeper layer. The absolute value of the resulting sum determines the extent to which the previously primed connections are strengthened or weakened. This mechanism is akin to the impact of neuromodulators on plasticity, particularly the role of serotonin.

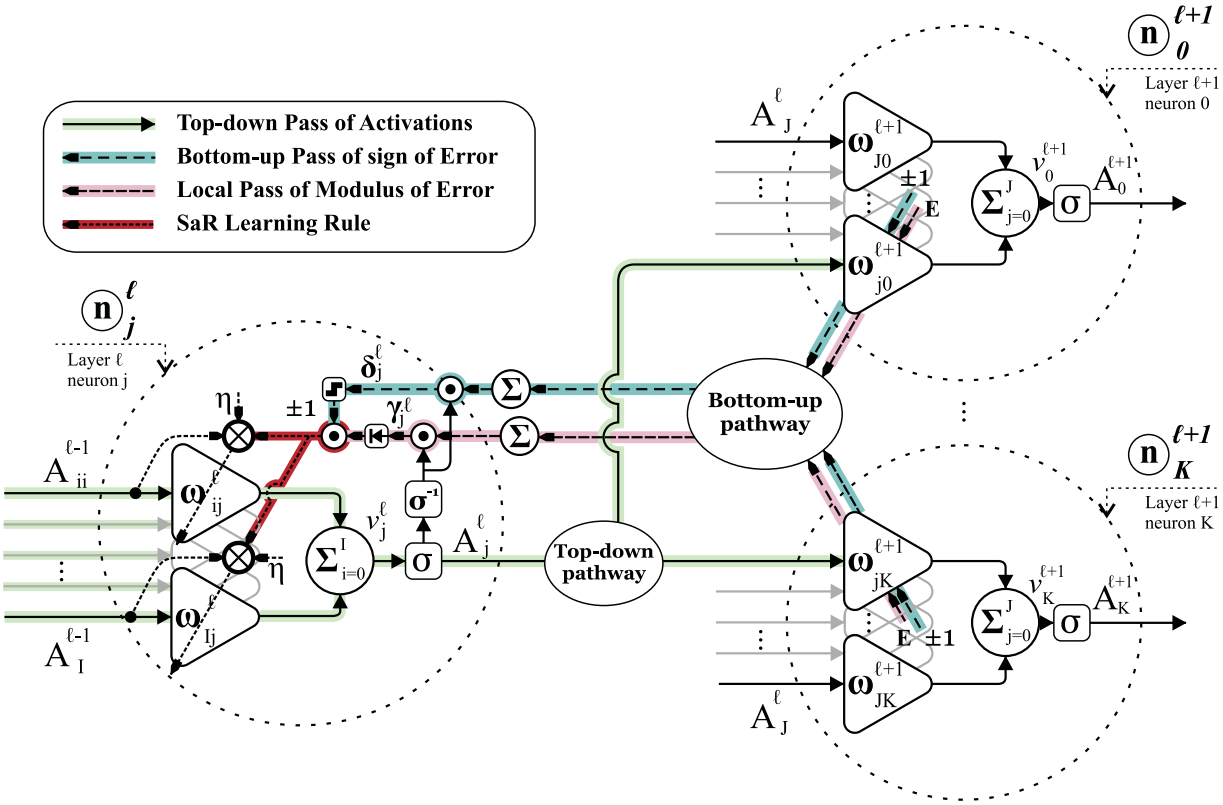

Figure 3.3: *Detailed signal propagation in SaR network.*

3.3.2 Inner Working of Neurons

In this section, we derive the learning rule for the SaR paradigm. The top-down passage of predictive inputs follows the conventional signal flow in a fully-connected feed-forward

neural network, as established in Section 2.3. This pipeline is represented by the green lines in Figure 3.4. This process results in the activation of the output neurons, leading to the generation of both the predictive output P and the predictive action A'.

As shown in Section 2.4, the execution of this action initiates a sequence of events within the closed-loop platform, culminating in the generation of the error signal defined in Equation 2.19. This error signal governs the learning process of the SaR network.

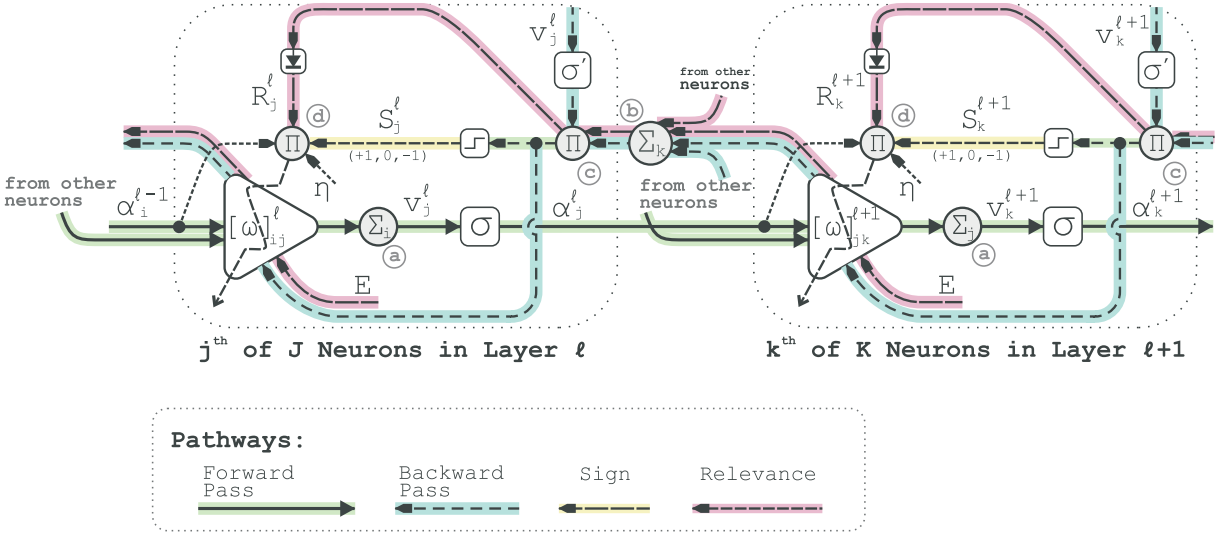

Figure 3.4: *Displays the internal connections between two neighbouring neurons within the SaR network. Forward propagation of inputs is shown with the left-to-right solid lines highlighted in green. σ is the sigmoid activation function and A_j^ℓ denotes the activation of the j^{th} neuron in layer ℓ. $[\omega]_{ij}^\ell$ is the weight matrix associated with inputs I inputs to this layer. The summation node ⓐ corresponds to Equation 2.4. Backpropagation pathway is shown with right-to-left dashed lines highlighted in blue. The summation at node ⓑ and product at node ⓒ correspond to Equation 2.16. The short yellow dashed lines carry the sign of the internal errors as calculated in Equation 3.2, and pink long-dashed lines show the pathways for the relevance signal, the summation at node ⓑ and the multiplication at node ⓒ correspond to Equation 3.3 as it relates to the relevance signal. The multiplication node ⓓ highlights the production in the learning rule as in Equation 3.4.*

3.3.2.1 Sign Signal

Recall the derivation of the internal error in an open-loop network, as described in Equation 2.17, and for the CLDL algorithm, as shown in Equation 2.52. In Figure 3.4, the green pipeline serves to carry and propagate the internal error of the neurons. However, in this work, only the *sign* of the internal error is utilised for the bottom-up propagation into deeper layers. The result can be either $+1$, -1, or 0. This signal, known as the sign signal and denoted as S_j^ℓ, is calculated as follows:

$$\delta_j^\ell = \sigma^{-1}(v_j^\ell) \cdot \Sigma_{k=0}^{K}(\omega_{jk}^{\ell+1} \cdot \frac{\delta_k^{\ell+1}}{|\delta_k^{\ell+1}|}) \tag{3.1}$$

$$S_j^\ell = \frac{\delta_j^\ell}{|\delta_j^\ell|} = \{+1, -1, 0\} \tag{3.2}$$

The yellow pathways illustrated in Figure 3.4 depict the calculation of the sign signal. It is important to note that the derivative of the sigmoid function is strictly positive and, as a result, has no influence on the resulting sign signal.

3.3.2.2 Relevance Signal

The magnitude of excitation or depression of neurons is, however, determined through a local pass of the error signal, which is formulated as:

$$R_j^\ell = |\sigma^{-1}(v_j^\ell) \cdot \Sigma_{k=0}^{K}(\omega_{jk}^{\ell+1} \cdot E)| \tag{3.3}$$

In this context, R_j^ℓ represents the relevance signal, which is emphasised by the pink pathways in Figure 3.4. It is worth noting that, unlike the internal error carried by the green lines, the relevance signal is not transmitted to deeper layers. Consequently, R_j^ℓ is independent of $R_k^{\ell+1}$, and the closed-loop error reinstigates this process in each layer. As a result, this signal can be simultaneously and globally generated across all layers.

3.3.2.3 Learning Rule

Having clarified the sign and relevance signals, we can now define the learning rule for the SaR network as follows:

$$\underset{\text{SaR}}{\Delta\omega_{ij}^\ell} = \eta \cdot S_j^\ell \cdot R_j^\ell(z) \cdot \alpha_i^{\ell-1}(-z) \tag{3.4}$$

This learning process occurs at node ⓓ in Figure 3.4, where all the terms are correlated to calculate the weight change, with η representing the learning rate.

3.4 Experimental Setup & Network Architecture

The experimental setup and common aspects of the network are detailed in section 2.7. In this section, we use 48 predictors, meaning the camera captures 240 inputs, thus the

input layer of the network is initialised with 240 neurons. The network features an encoder topology, with 10 hidden layers that linearly decrease in the number of neurons: $\{\overbrace{240}^{\text{Input}}, \overbrace{13, 12, ..., 5, 4,}^{\text{Hidden}} \overbrace{3}^{\text{Output}}\}$.

3.5 Results

In this section, we present a comparison between the CLDL algorithm and the novel SaR learning approach. The following findings in this section refer to result (c) in Figure 1.1 in the preface. To gain a deeper insight, additional experiments were conducted by focusing on the local propagation of the relevance signal exclusively. This further underscores the significance of the synergy between the sign and the relevance signal.

3.5.1 Error Minimisation: Trial with $\eta = e^{-5}$

Figure 3.5 presents a series of trials conducted with a learning rate of $\eta = e^{-5}$. In Panel A of this figure, we observe the error signal and its moving average in a trial utilising CLDL. The error signal exhibits persistence for approximately 200 seconds before gradually converging to zero at $t_1 = 333$ seconds.

Panel B displays the same information for a trial with local propagation only, where the error signal is injected into each layer and propagated for just one layer to drive the learning process. In this case, learning is notably faster than in the CLDL trial, with the error signal persisting for only 20 seconds before converging to zero at $t_2 = 93$ seconds.

Finally, Panel C illustrates the results of a trial employing SaR learning, where the combination of sign and relevance signal propagation drives the weight change as defined in Equation 3.4. Notably, in this scenario, learning is significantly improved compared to the two previous trials. The error signal does not persist; instead, it immediately begins to converge, and learning is achieved at $t_3 = 42$ seconds.

3.5.2 Error Minimisation: Trial with $\eta = e^{-1}$

Figure 3.6 presents a similar set of trials to those in Figure 3.5, but with a different learning rate, $\eta = e^{-1}$. In Panel A, during a trial with CLDL, we observe that the error signal spikes over a period of 25 seconds before fully converging at $t_4 = 32$ seconds.

Panel B shows the results of a trial with local propagation, where the error signal only spikes twice before fully converging at $t_5 = 13$ seconds.

In Panel C, during a trial with SaR, one-shot learning is achieved. The error signal spikes once at $t = 5$ seconds, and learning is completed at $t_6 = 12$ seconds. Further details regarding this one-shot learning are provided in Figure 3.7.

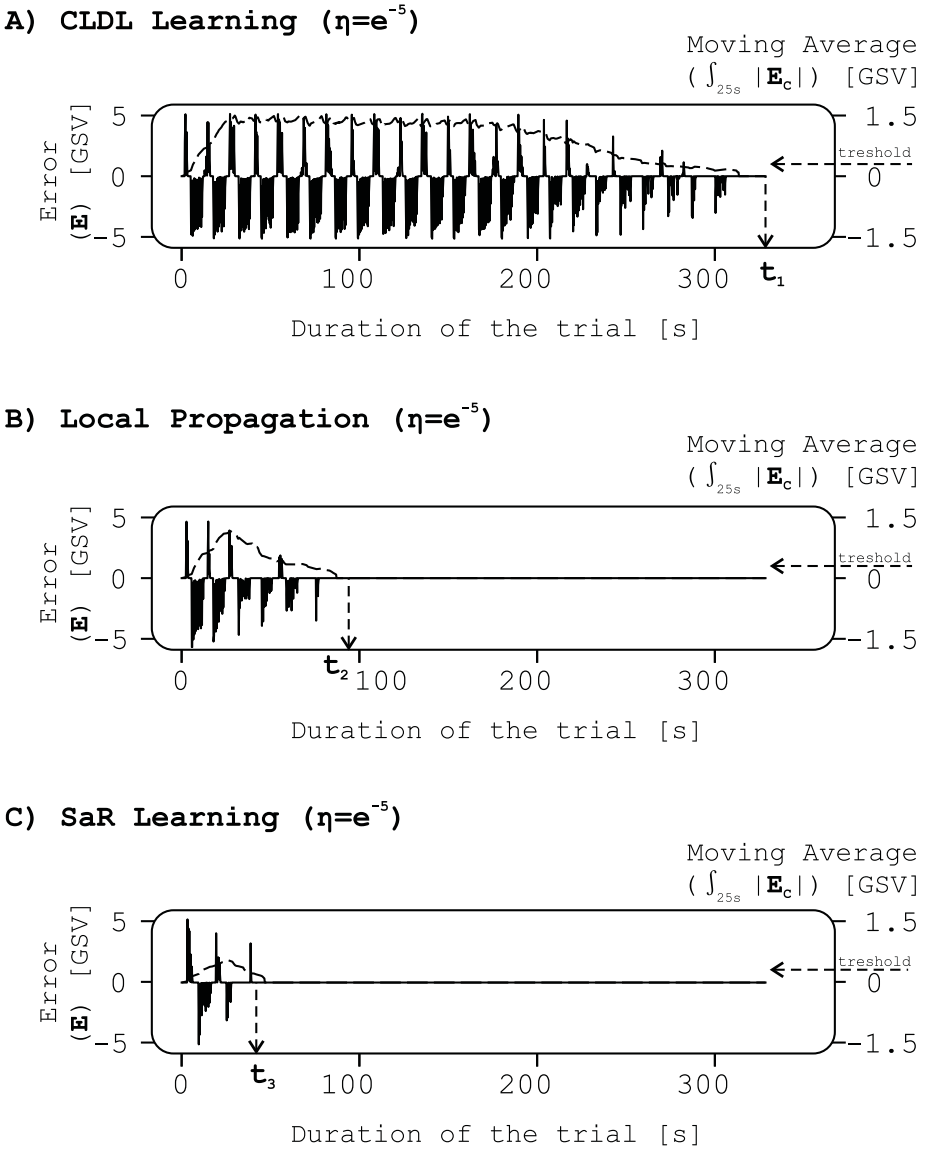

Figure 3.5: *Comparison of CLDL, the local propagation of the relevance signal, and SaR, with learning rate of* $\eta = e^{-5}$

3.5.3 Predictors & Motor Command: Trial with $\eta = e^{-1}$

Panel A displays the error signal, Panel B exhibits the activity of one predictor from each row of the camera view, and Panel C shows the network's output sent to the motors. The robot initially encounters the line at time $t_7 = 8$ seconds. Prior to this point, predictive signals are active, but there is no steering command at the network's output. Consequently, the robot encounters the path, and the error signal spikes at $t_8 = 9$ seconds. This error signal triggers the learning process, which is completed between times t_7 and t_8.

From this point onward, the network generates an appropriate steering command based on the activity of the predictors at the input. This ensures that the error signal remains at zero until the success condition is met at time $t = 37$ seconds.

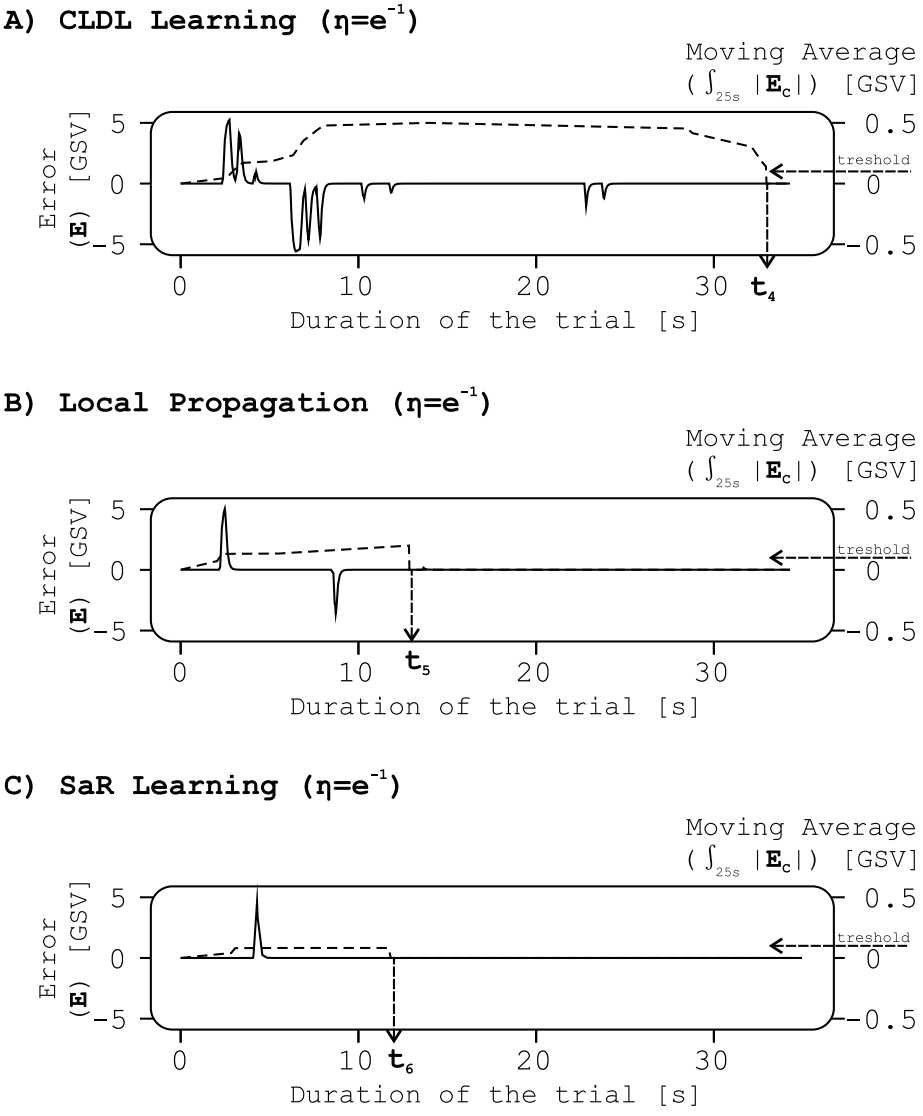

Figure 3.6: *Comparison of CLDL, the local propagation of the relevance signal, and SaR, with learning rate of* $\eta = e^{-1}$

3.5.4 Euclidean Weight Distance & Convergence

The rapid one-shot learning exhibited in the previous trials raises questions about potential issues such as over-fitting or unstable learning. To assess the stability and behaviour of the network, a comparison is made between the nature of the weight changes and the convergence of the trials as previously presented in Figure 3.6.

Figure 3.8 provides a comparison of weight changes in trials using the various learning paradigms.

In Panel A, we observe that in the CLDL trial, the total weight changes in all layers fall within a range of 1 to 4 (highlighted in the grey area) from their initial random values. The weight change in the first layer, as highlighted by the black line, is approximately 2 for this trial, indicating its greater significance, as previously described by Porr and Miller

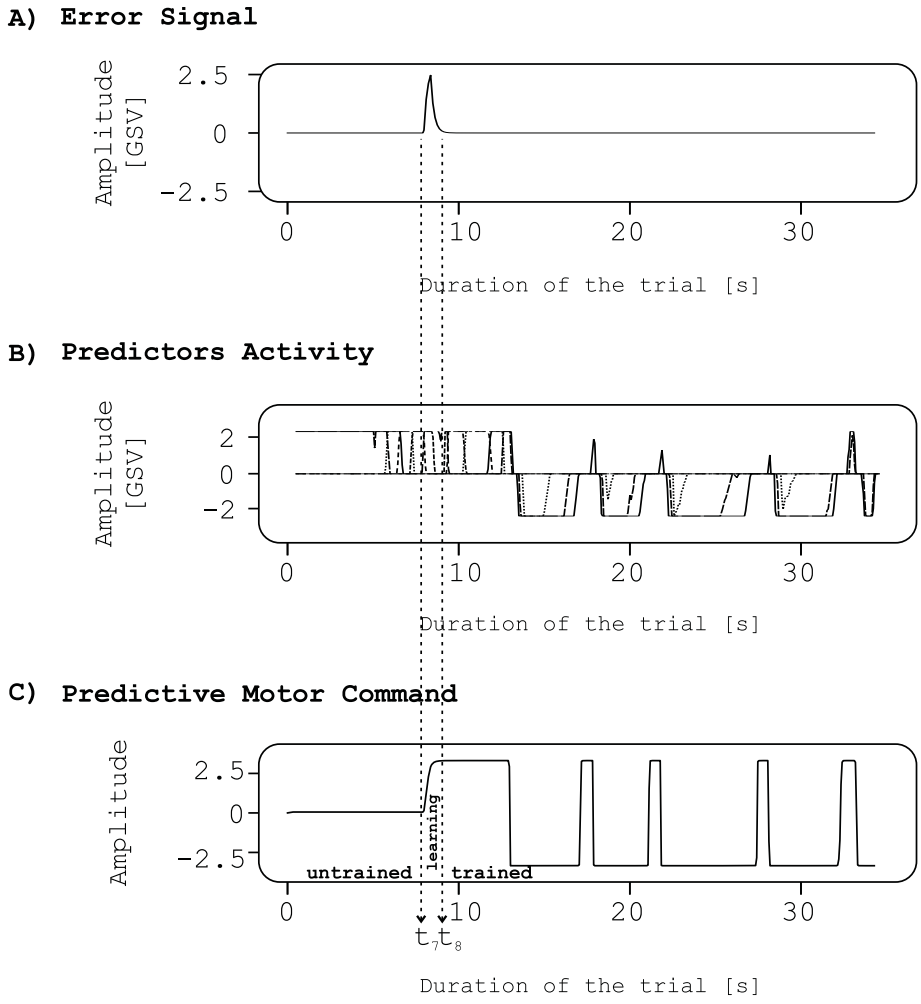

Figure 3.7: *A trial with SaR: showing the error signal, predictors activity and the motor command*

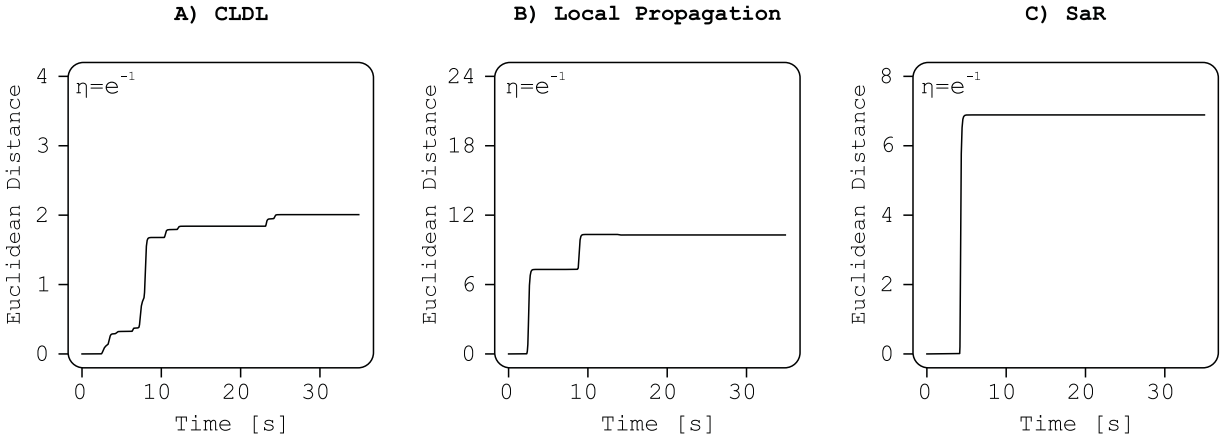

Figure 3.8: *Weight distances for trials with CLDL, local propagation of relevance signal, and the SaR*

(2020).

Panel B presents the results for the trial with local propagation, where the range of weight changes for all layers is between 3 and 24, with the first layer showing a weight change of approximately 10. This significant increase in weight changes raises questions about the nature and efficiency of learning.

In Panel C, the results for the trial with SaR in presented, where the weight change is not significantly greater than that of the CLDL trial (almost doubled), with weight changes ranging from 3 to 8 for individual layers and a total weight change of approximately 7 for the first layer.

From the observations in this figure, combined with those in Figure 3.6 and Figure 3.5, it can be concluded that SaR offers much faster learning without the risk of unstable weight changes or over-fitting.

3.5.5 Input Layer Weight Distribution

As mentioned earlier, the significance of learning at the input layer for a closed-loop organism is emphasised by Porr and Miller (2020). Figure 3.9 presents the weight distribution in the first layer of the network after learning is achieved, for the three paradigms mentioned above.

In Panels A, B, and C, we can see the distribution of weights after trials with CLDL, local propagation, and SaR, respectively. It is evident that all learning paradigms assign importance to the predictive signals in a similar pattern. This weight distribution follows the same trend as explained in-depth in Section 2.8.3.2.4. However, SaR provides a more distinct pattern of distinction between the predictive signals compared to local propagation and BP.

3.5.6 Statistics & Reproducibility

Figure 3.10 illustrates the reproducibility of these outcomes.

In Panel A, we observe the total error integrals across 10 trials with each of the learning paradigms, and in Panel B, we see the time taken to reach the success condition. It is evident that SaR consistently offers both faster learning and smaller error accumulation when compared to the other paradigms.

To further validate the results, additional trials were conducted using different learning rates $\eta = \{e^{-5}, e^{-4}, e^{-3}, e^{-2}, e^{-1}\}$. Figure 3.11 presents this data.

In Section A and B, we observe the error integral and the time taken to reach success for trials with local propagation and SaR. It is clear that SaR consistently outperforms local propagation, achieving faster learning with smaller error accumulation. Additionally, it is noteworthy that as the learning rate increases, both paradigms exhibit improvements

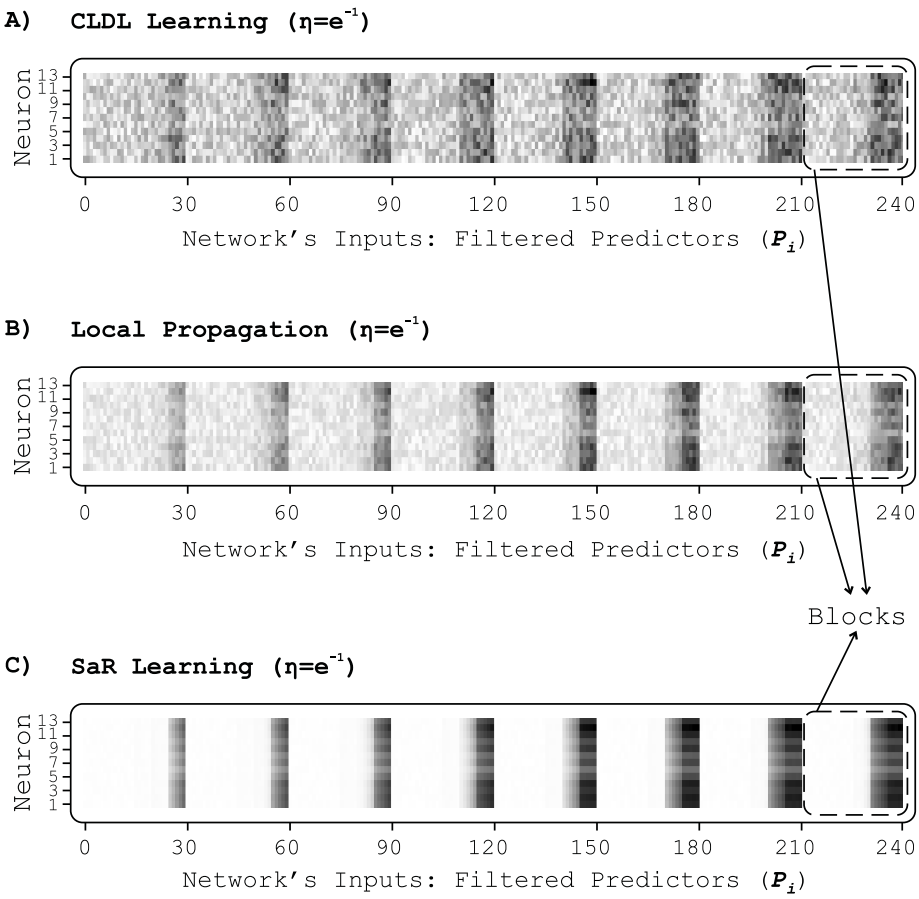

Figure 3.9: *Displays the distribution of weights in the initial layer of the neural network during the trials with CLDL, local propagation, and SaR. The vertical axis indicates the neuron indices, whereas the horizontal axis denotes the indices of inputs, namely the filtered predictors. The greyscale is used to represent the final weights, where black indicates the highest weight value and white the lowest. Each distinct segment corresponds to a row of predictors.* A) *Presents the distribution of weights for a trial using CLDL.* B) *Illustrates the weight distribution in a trial involving local propagation.* C) *Shows this result for a trial with SaR. All trials were conducted with a learning rate of* e^{-1}.

in terms of error integral and time to success.

3.5.7 Optimal Learning Rate

For a more equitable comparison between SaR and CLDL, we conducted additional simulation experiments using higher learning rates to identify the optimal settings for each of these paradigms. Both paradigms failed to converge under the specified success criteria when using a learning rate of e^2. For completeness, we considered learning rates in the set $\eta = \{e^{-2}, e^{-1}, e^0, e^1, e^2\}$, and repeated the experiments 10 times. The results of these experiments are presented in Figure 3.12. In Panel A, the results are shown for a deep neural network with an encoder topology. CLDL performs best at a learning rate of e^1,

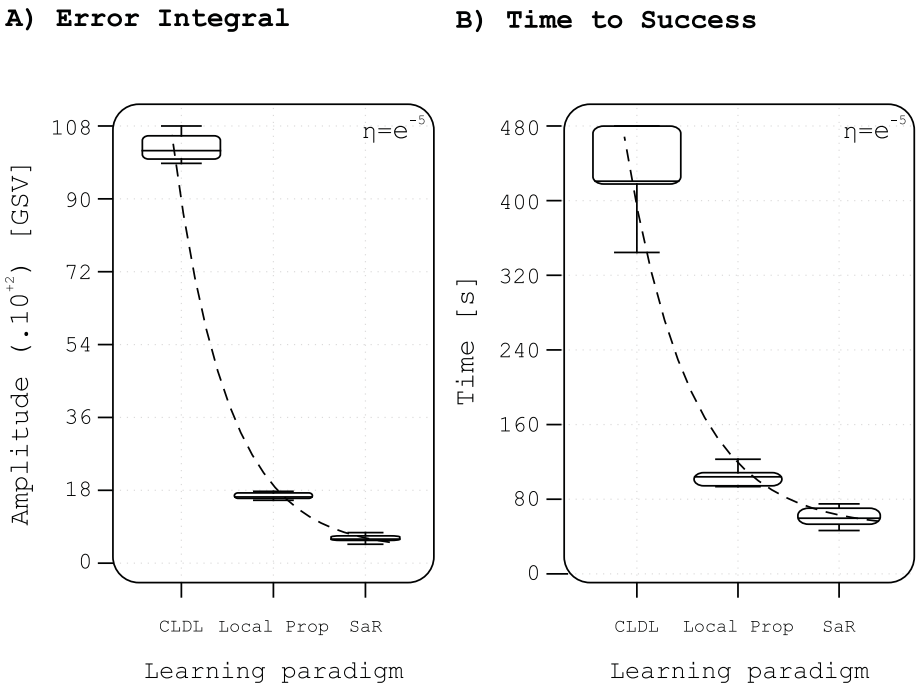

Figure 3.10: *Reproducibility of the results: a comparison of SaR with CLDL and local propagation with a learning rate of* e^{-5}. A) *shows the total error integral during these trials, and* B) *shows the time taken to reach success.*

reaching an average of 288 steps before convergence, while SaR performs optimally at e^0, requiring an average of 37 steps for convergence. Panel B illustrates a similar outcome for a network with a square topology. Here, both paradigms achieve their best performance at a learning rate of e^0. CLDL converges in an average of 64 steps, while SaR achieves success in an average of 15 steps.

3.5.8 Encoder Network Topology

Figure 3.13 illustrates the effect of the number of layers in a network with a typical triangle-shaped encoder topology. The solid line represents SaR and the dashed line represents CLDL. The number of hidden layers increases from 0 to 20, with the number of neurons increasing linearly for each added layer: $\{\overbrace{240}^{\text{Input}}, \overbrace{23, 22, ..., 5, 4,}^{\text{Hidden}} \overbrace{3}^{\text{Output}}\}$. Zero hidden layers indicate that the network is initialised with one input layer and one output layer, where inputs are directly fed into the output neurons. Mathematically speaking, there is no difference between the two algorithms when no hidden layers are present. This has been experimentally demonstrated, as both algorithms produce identical results. Interestingly, the SaR algorithm is less affected by variations in the number of layers, as expected from its single-layer propagation. On the other hand, the CLDL algorithm shows a decrease in performance as the number of layers increases. It is worth noting that there are no

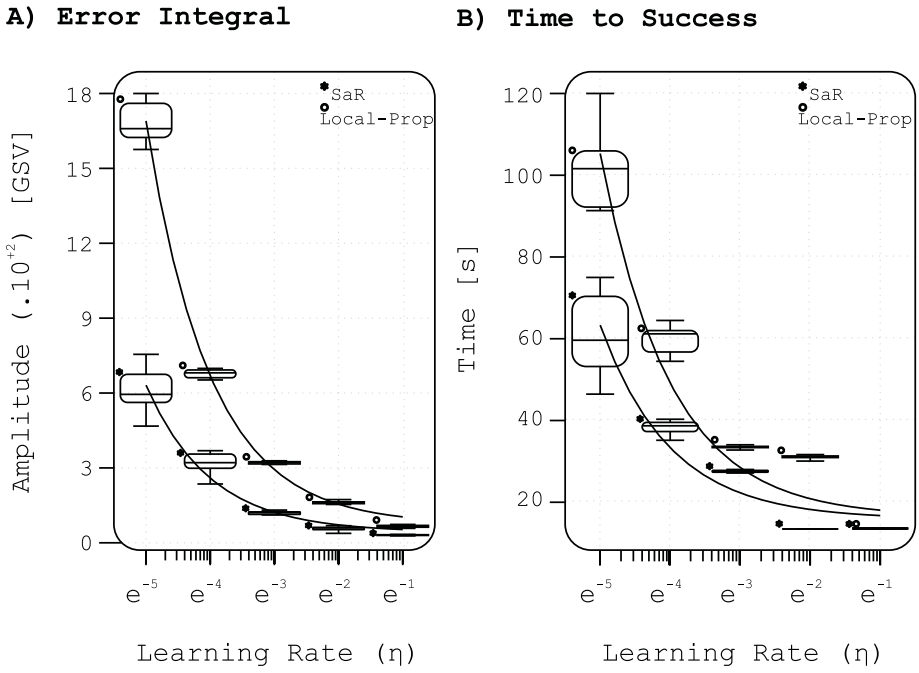

Figure 3.11: *Reproducibility of the SaR paradigm with different learning rates* $\eta = \{e^{-5}, e^{-4}, e^{-3}, e^{-2}, e^{-1}\}$. A) *shows the error signal during these trials, and* B) *shows the time taken to reach the success condition. The star-marked boxes represent the results for SaR learning, while the circle-marked boxes represent the results for local propagation of the error.*

data points for CLDL with more than 12 hidden layers since the success condition was not met (convergence was not achieved), which is due to the vanishing gradient problem. It is evident that the SaR network continues to perform well with as many as 20 hidden layers without significant changes in the results shown.

3.5.9 Square Network Topology

In a triangle-shaped encoder network, the total number of neurons increases drastically with each added layer. To make a more accurate comparison between the two paradigms, another set of simulations was conducted using a square-shaped network where the number of neurons was fixed at 10 per layer for each added layer: $\{\overbrace{240}^{\text{Input}}, \overbrace{10, 10, ..., 10, 10,}^{\text{Hidden}} \overbrace{3}^{\text{Output}}\}$. Figure 3.14 illustrates the results of these experiments. As the number of hidden layers increases, both algorithms show some improvement. However, as the number of layers continues to grow, the SaR algorithm is less affected by the depth of the network, whereas the CLDL algorithm progressively declines in performance. For CLDL, the time taken to achieve success grows exponentially with an increasing number of hidden layers, and the error integral shows a linear growth. This indicates that CLDL suffers from the vanishing gradient problem in deep networks, unlike its SaR counterpart.

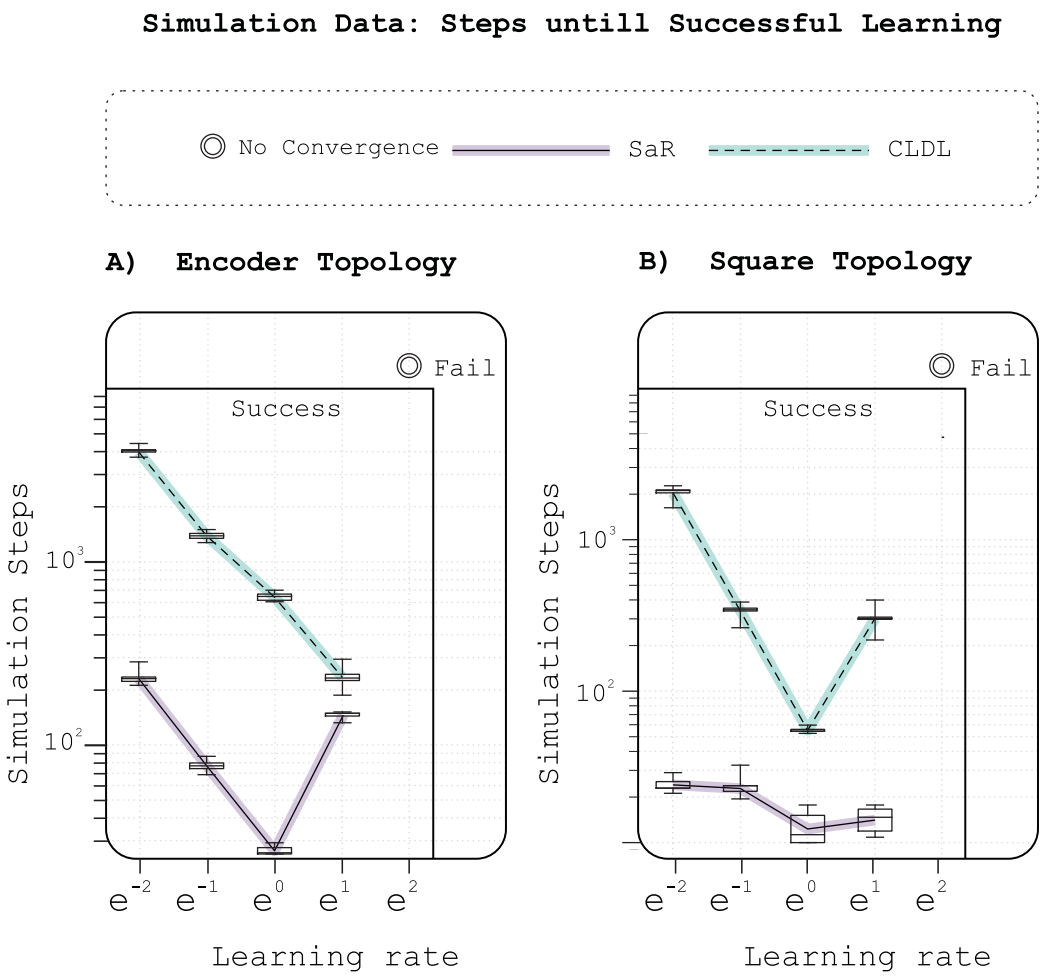

Figure 3.12: *Simulation results with higher learning rates include the time taken to meet the success condition. Runs exceeding 10,000 steps are considered failures, indicating that the network did not converge.* A) *Results with a deep network with an encoder topology.* B) *Similar results with a square topology.*

A comparison of the two topologies reveals that both algorithms perform better with a square-shaped network where the number of neurons is the same across all layers. This is expected because any increase in the number of neurons in triangle-shaped encoder networks directly affects the weighted sums of inputs and the internal errors in the forward and backward passes, respectively. As a result, the propagation is more susceptible to unstable changes, which can negatively impact performance.

The results presented in this work are unaffected by the shape of the path that the robot follows. The primary reason for this independence is the nature of the learning process employed. In this application, each time step represents one iteration of learning. Unlike typical Q-learning applications that aim to achieve a specific goal, such as finding the optimal route or uncovering a hidden treasure, there is no ultimate objective here. Each time step constitutes one complete iteration, ensuring that the results and conclusions are not influenced by the path's shape. This independence is further highlighted by the fact

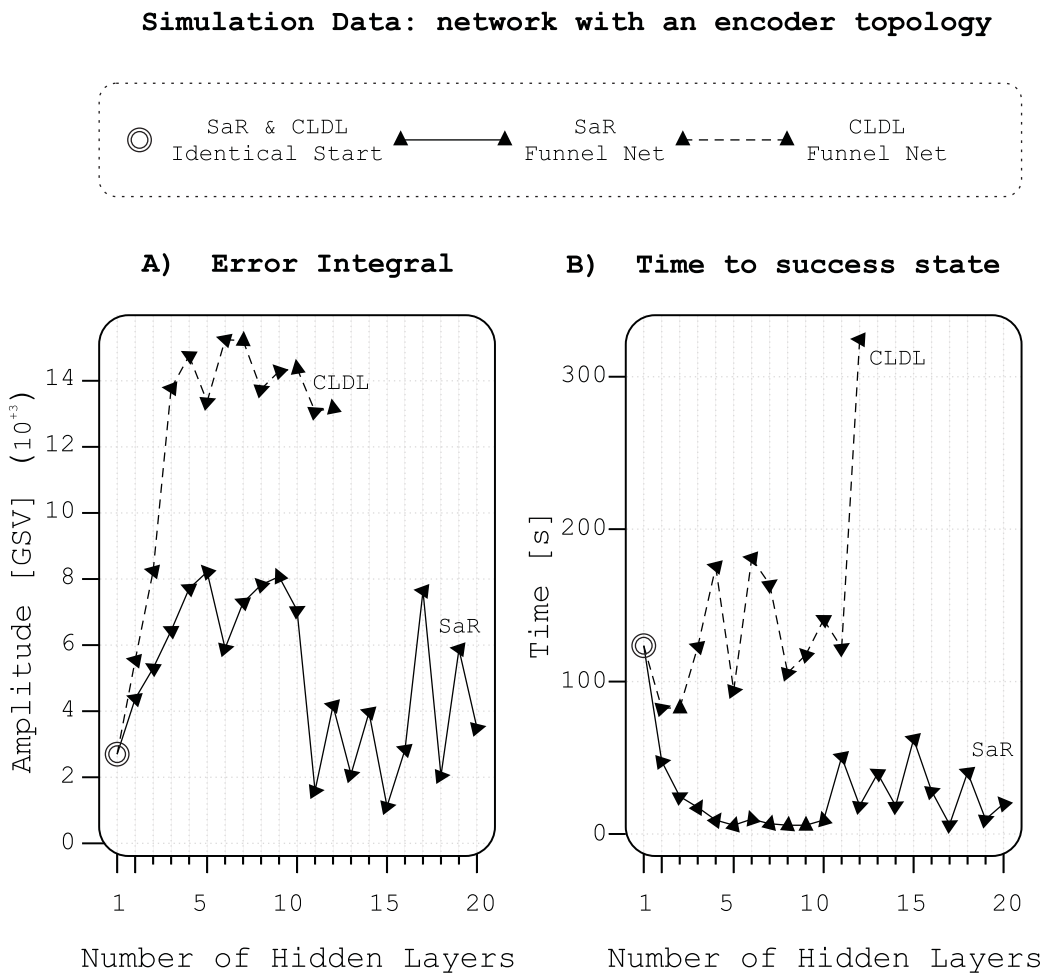

Figure 3.13: *The effect of the depth of the network, with an encoder topology, on the total error integral* A)*, and the time taken to reach the success state* B)*.*

that these learning trials occur within seconds, during which the robot traverses only a small fraction of the path.

3.6 Discussion

SaR offers faster convergence in closed-loop learning tasks, integrating neuromodulators and calcium-induced LTD or LTP, mirroring neurophysiological functions. This method may also support multi-modal processing, using any input to understand a forward model.

Lillicrap et al. (2016a) relates deep learning to the cortex but overlooked global neuromodulation like serotonin or dopamine. Classic biologically grounded RL models (Schultz and Suri, 2001; Wörgötter and Porr, 2005; Prescott et al., 2006) use reward prediction error, similar to the dopaminergic signal in the striatum (Schultz et al., 1997). While aligning with biological processes, these models face challenges tied to global error signals in deep structures, where varied layers undergo similar alterations, limiting deeper structures' efficacy.

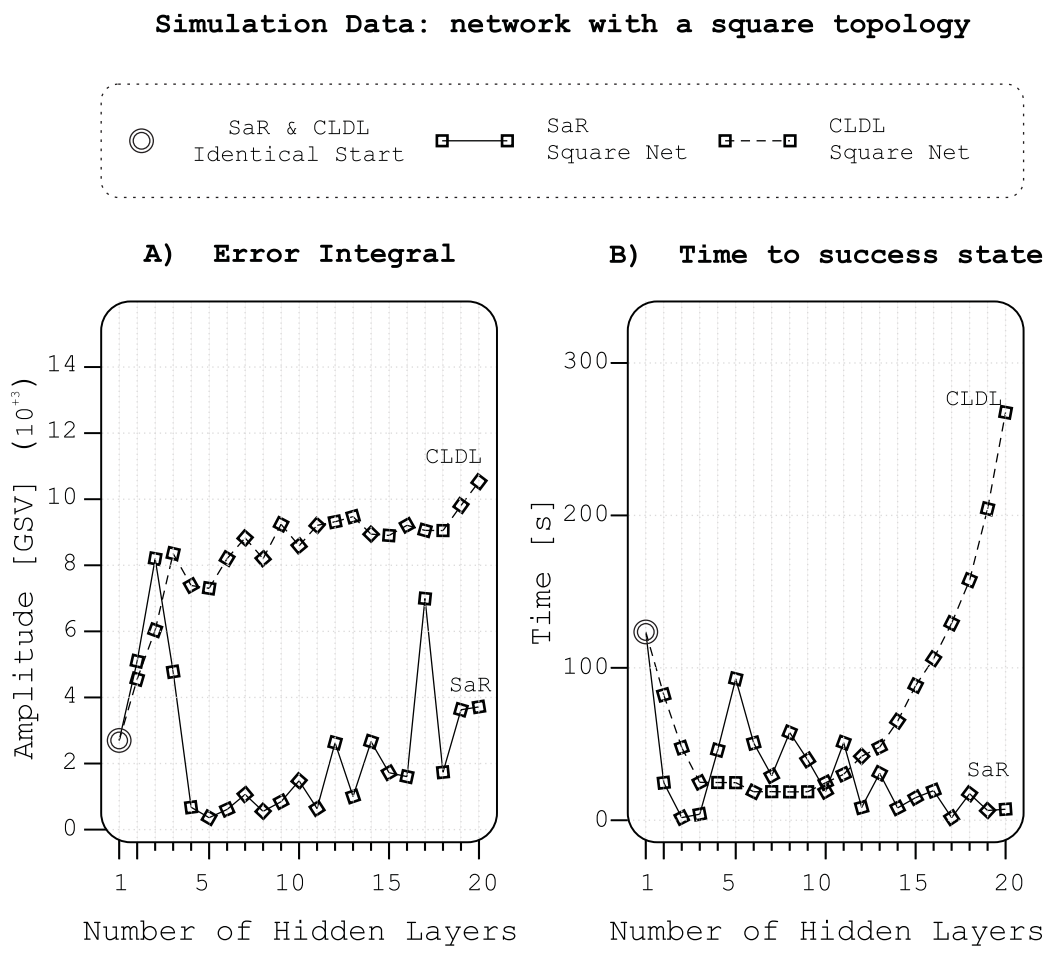

Figure 3.14: *The effect of the depth of the network, with a square topology, on the total error integral* A) *and the time taken to reach the success state* B).

Over the past two decades, reward-associated neuromodulators have been viewed as relaying error signals, but this perspective remains debated (Schultz et al., 1997). For instance, serotonin represents both reward and penalty anticipation (Li et al., 2016; Crockett et al., 2009; Cohen et al., 2015). We mathematically encapsulated this as a "modulus," but neurophysiologically, it might be more fitting as a "relevance signal," triggering plasticity (Porr and Wörgötter, 2007). Similarly, the adverse reaction of dopamine neurons to unfavourable reward predictions has been questioned due to its minimal baseline firing rate (Schultz, 2004). An alternative reading of serotonin and dopamine cues is as relevance signals (Porr and Wörgötter, 2007), intensifying plasticity (Lovinger, 2010; Iigaya et al., 2018), with inherent plasticity learning protocols determining LTP or LTD (Castellani et al., 2001; Inglebert et al., 2020).

Using global neuromodulation and local plasticity, we illustrate a biologically realistic processing method in Figure 3.15. This circuit, influenced by Larkum (2013); Rolls (2016) and enhanced with neuromodulatory innervation (Lovinger, 2010; Iigaya et al., 2018), features two pathways. The bottom-up route transmits sensor signals from input ("In") to output ("Out") through layers L0 to L2. The top-down path determines the sign of

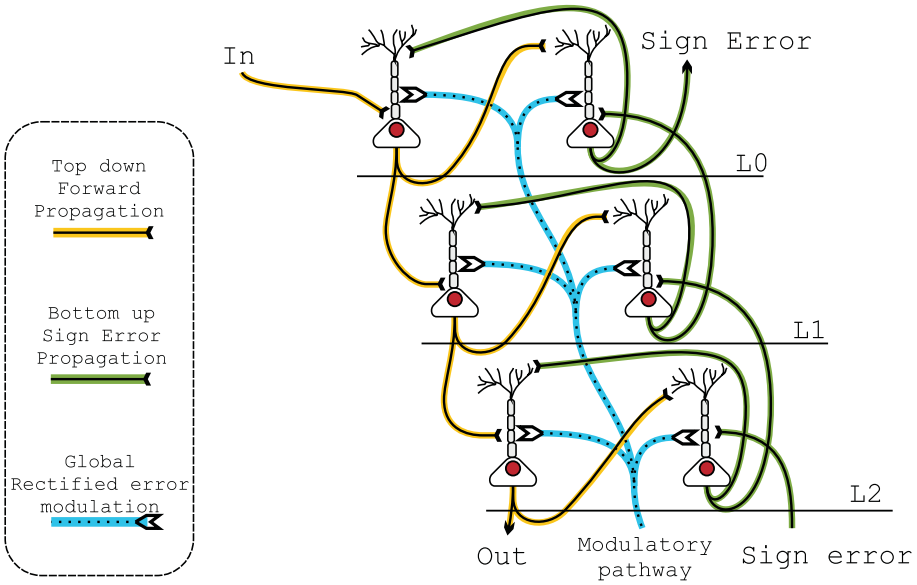

Figure 3.15: *Proposal of a neurophysiologically realistic model of SaR (Sensation and Action Replay) learning is depicted in the figure. The model comprises three network layers, denoted as L0, L1, and L2. Signal processing occurs through three pathways:* A) *The "bottom-up" pathway, responsible for transmitting a signal from "In" to "Out"* B) *The "top-down" pathway, tasked with conveying the error signal labelled as "Sign"* C) *The "modulatory" pathway, which imparts a global signal to' all neurons within the network. In the bottom-up and top-down pathways, signals traverse synapses located proximate to the respective somas. Conversely, reciprocal connections between neurons within a layer link to the dendrites, thereby influencing plasticity.*

learning and decides which neurons experience LTP or LTD.

The bottom-up pathway conveys the error signal's sign from layer two to layer zero via three synapses. Global neuromodulation E governs neuron plasticity (Eq. 3.3), related to neurophysiology through postsynaptic calcium concentration. Our binary distinction between LTP and LTD is based on Inglebert et al. (2020). Only significant calcium influx from both somatic burst spiking and dendritic calcium spikes results in LTP; less leads to LTD (Tamosiunaite et al., 2007).

Projections to dendrites alone cannot induce neuronal spiking. However, when aligned with somatic inputs, they prompt long-lasting bursts, causing LTP due to significant calcium influx (Larkum, 2013). Limited dendritic activation can induce LTD from minor calcium influx (Inglebert et al., 2020; Shouval et al., 2002). Considering reciprocal neural connections shown in Figure 3.15, strong spiking in the top-down neuron, due to high L2 synaptic weight, can stimulate the bottom-up neuron, affecting calcium influx. Intense influx initiates LTP; minor influx leads to LTD. Enhanced bottom-up synaptic weights intensify activity, influencing the top-down pathway and calcium influx. Thus, weight development signs between reciprocal neurons in both pathways are mirrored, affected by mutual connections and calcium concentrations. The neuromodulator adjusts the learning

rate, modulating LTP or LTD extents. Comprehensive biophysical modelling is needed to deeply explore this model, potentially benefiting mental illness models (Rolls, 2016).

Both global neuromodulation and local calcium-driven plasticity are essential for effective behavioural adjustment in learning experiments. Successful cortical learning demands both local calcium plasticity and neuromodulation. Inhibiting the N-methyl-D-aspartate (NMDA) receptor would interfere with calcium-driven plasticity, potentially affecting reward- or punishment-based learning. If cortical circuitry determines learning direction (LTP or LTD) and broadcasts it, disrupting this error transmission would maintain generic learning via the neuromodulator but strip its goal-directed nature, making learning aimless and less valuable.

In the context of optimal control, a similarity between Sign and Relevance learning and sliding mode control is the decomposition of a signal into its sign and magnitude. In sliding mode control, this decomposition enhances the stability of the system. Similarly, in Sign and Relevance learning, it is utilised to improve the learning process and convergence of the network.

Chapter 4

Prime & Modulate (PaM) Learning

4.1 Introduction

In this section, we introduce the PaM algorithm, building on the concepts previously introduced for the SaR paradigm. We begin by providing a concise motivation, followed by the design and derivation of the platform and its associated learning rule. Subsequently, we present the results section and conclude with a discussion.

4.2 Motivation

Deep learning faces challenges such as the exploding and vanishing gradient problem (EVGP). This issue arises when the error signal in neural networks, using activation functions like logistic and hyperbolic tangent (tanh), becomes too small or too large, disrupting learning. To address EVGP, researchers have explored new network architectures such as long short-term memory (LSTM), better weight initialisation methods, and alternative activation functions. Notably, linear rectifying units have been adopted to mitigate EVGP by transforming network dynamics and learning characteristics (Hanin, 2018).

From a neurophysiological perspective, learning involves both local and global mechanisms affecting synaptic plasticity (Reynolds and Wickens, 2002). This interaction has shown benefits in simple networks but has been less applied to complex architectures (Porr and Wörgötter, 2007). This study introduces a new learning approach combining local error back-propagation with global modulation to create a resilient forward model. This method uses the error signal's sign for local weight adjustments and a global "relevance" signal to enhance these adjustments, preventing EVGP and aligning more closely with neuroscience models than traditional back-propagation.

The research builds on traditional back-propagation by dividing the error signal into sign and magnitude components, leading to the development of the SaR algorithm presented in the previous chapter. The novel approach incorporates environmental cues into

the learning process, allowing systems to adapt dynamically. For example, self-driving vehicles can adjust their learning rate based on traffic conditions, enhancing safety and decision-making. This adaptive learning balances quick adaptation with thorough exploration, improving the system's effectiveness in varied situations.

4.3 Design & Derivation of PaM Algorithm

4.3.1 The Platform

Figure 4.1 illustrates the prime and modulate (PaM) learning paradigm. The reflex and learning loops function as previously described in Section 2.4. Similar to the CLDL and SaR algorithms, the PaM learning also constructs a forward model of the reflex using predictive learning.

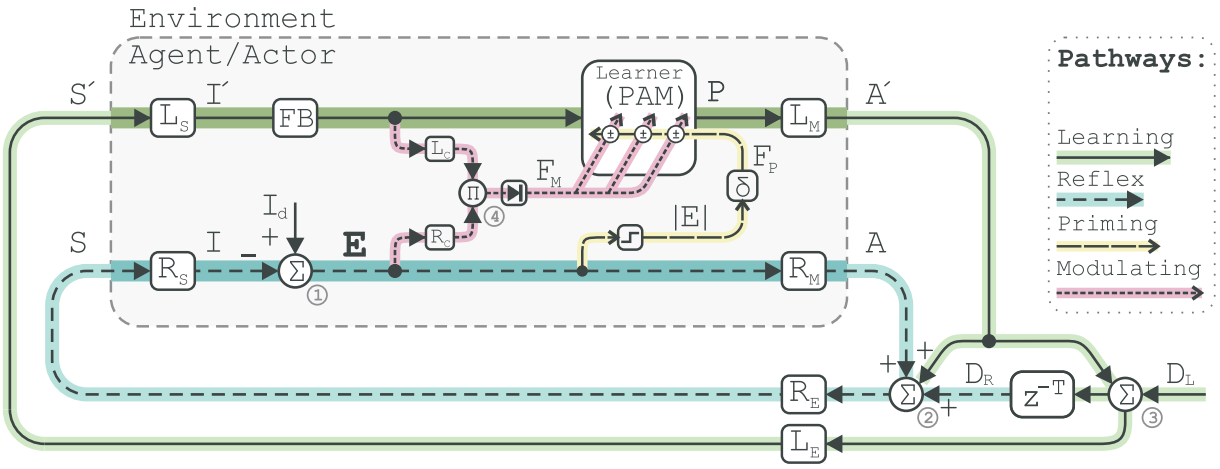

Figure 4.1: *Depicts the incorporation of the PaM learner in the closed-loop learning platform. The blue loop illustrates the reflex pathway, while the green loop represents the learning pathway. The yellow pathway demonstrates the extraction of the priming factor F_P from the error signal, which is then incorporated into the network. The red pathway showcases the extraction of environmental cues, or the modulating factor F_M from both the reflex and the learner, subsequently employed in the learning process.*

The distinctive feature of the PaM algorithm is its incorporation of environmental cues in the agent's learning process. This sets it apart from the other paradigms outlined in this study, where the sole driving force for weight adjustments is the error signal. This applies whether through a single back-propagation pathway, as in CLDL; a solitary forward-propagation pathway, as in FCL; a fusion of both, as demonstrated in Echo learning (to be introduced in the subsequent chapter); or via two distinct pathways, as observed in SaR.

Analogous to the SaR learning approach, the error's sign and magnitude are treated separately and employed for distinct purposes. The error's sign is routed through the

yellow pathway, depicted in Figure 4.1, referred to as the priming factor F_P. Conversely, the pink pathway illustrates how the error signal collaborates with other input from the learner's environment, culminating in the modulating factor F_M that propels the agent's learning process.

4.3.2 Inner Workings of Neurons

The PaM network employed a fully connected feed-forward neural network, as detailed in Section 2.3. In Figure 4.2, you can observe the internal connections between two neurons within this network. The forward propagation of inputs is indicated by the solid left-to-right blue lines.

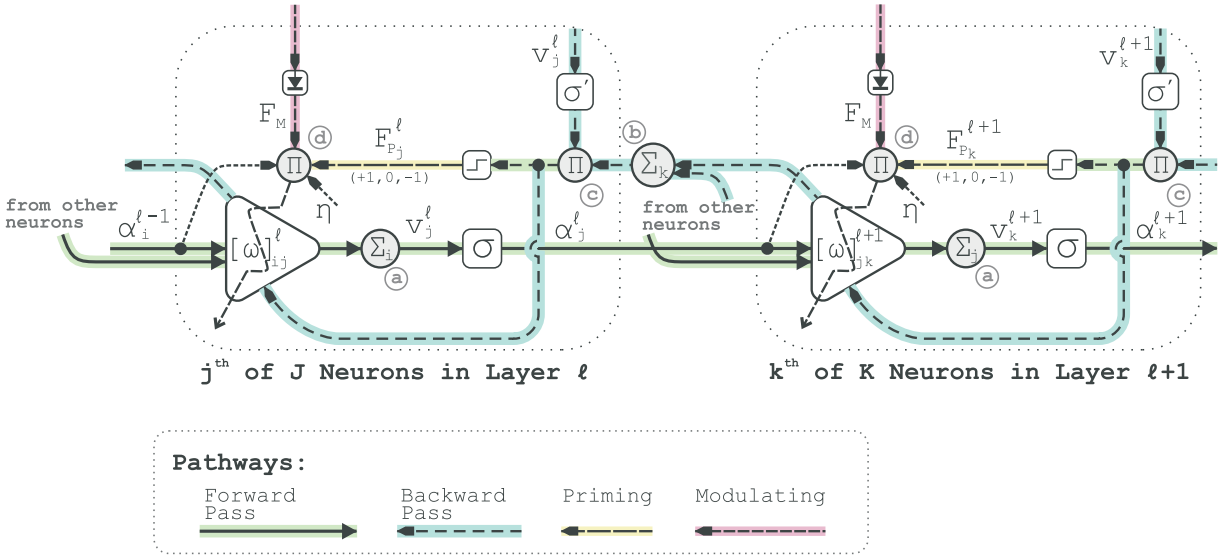

Figure 4.2: *Displays the internal connections between two neighbouring neurons within PaM network. Forward propagation of inputs is shown with the left-to-right solid lines highlighted in green. σ is the sigmoid activation function and A_j^ℓ denotes the activation of the j^{th} neuron in layer ℓ. $[\omega]_{ij}^\ell$ is the weight matrix associated with inputs I inputs to this layer. The summation node ⓐ corresponds to Equation 2.4. Backpropagation pathway is shown with right-to-left dashed lines highlighted in blue. The summation at node ⓑ and product at node ⓒ correspond to Equation 2.16. The priming pathway is shown with short yellow dashed lines highlighted in red. This is the sign of the resulting value from the backpropagation pathway, see Equation 4.1. The modulating pathway is shown in long pink dashed lines that enter each neuron from the environment, see Equation 4.2. The priming and modulating factors join at node ⓓ, together with the learning rate η and the relevant input to the neuron $A_i^{\ell-1}$, to drive the learning rule, corresponding to Equation 4.3.*

4.3.2.1 Priming Factor: $F_{P_j^\ell}$

Revisiting the derivation of the internal error for the CLDL algorithm in Equation 2.52, in this particular paradigm, only the sign of this term is utilised. This sign essentially acts as a stimulus that prepares the weights for subsequent adjustments—whether it involves an increase, decrease, or no change. As a result, within this paradigm, it is denoted as the priming factor:

$$F_{P_j^\ell} = \frac{\delta_j^\ell}{|\delta_j^\ell|} = \begin{cases} +1 & \textit{primes } \omega \textit{ to be increased} \\ 0 & \textit{primes } \omega \textit{ to remain unchanged} \\ -1 & \textit{primes } \omega \textit{ to be decreased} \end{cases} \tag{4.1}$$

4.3.2.2 Modulating Factor: F_M

Once primed, the extent of weight adjustment is determined by a secondary signal that contains collective cues acquired from the environment. These cues convey the significance of the learning experience at any given moment. In Figure 4.1, R_C and L_C are functions devised to extract pertinent cues from the reflex and predictive loops, respectively. Their correlation is illustrated at the product point ④. Consequently, this signal is termed the Modulating Factor F_M:

$$F_M = |ER_C \cdot I'L_C| \tag{4.2}$$

In contrast to the relevance signal in the SaR network, this modulating factor does not propagate through the layers. Instead, it directly influences the weights of all neurons after they have been primed to either increase, decrease, or remain unchanged.

4.3.2.3 Learning Rule

Given this, the update rule for the PaM paradigm can be defined as:

$$\Delta\omega_{ij}^\ell = \eta(F_P)_j^\ell \cdot F_M(z) \cdot \alpha_i^{\ell-1}(-z) \tag{4.3}$$
PaM

The correlation between the priming and modulating factors takes place at node ⓓ in Figure 4.2. This represents a generic learning rule, as the definition and customisation of the modulating factor, which hinges on environmental cues, need to be specified for particular applications. The subsequent section presents the definition of this factor for the application discussed in this study.

4.4 Application-Based Calculation of F_M

As shown in Figure 4.1 node ④, F_M is the product of the clues extracted from the reflex and learner's loops through R_C and L_C, respectively. This section shows how these transfer functions are defined for a line-following application.

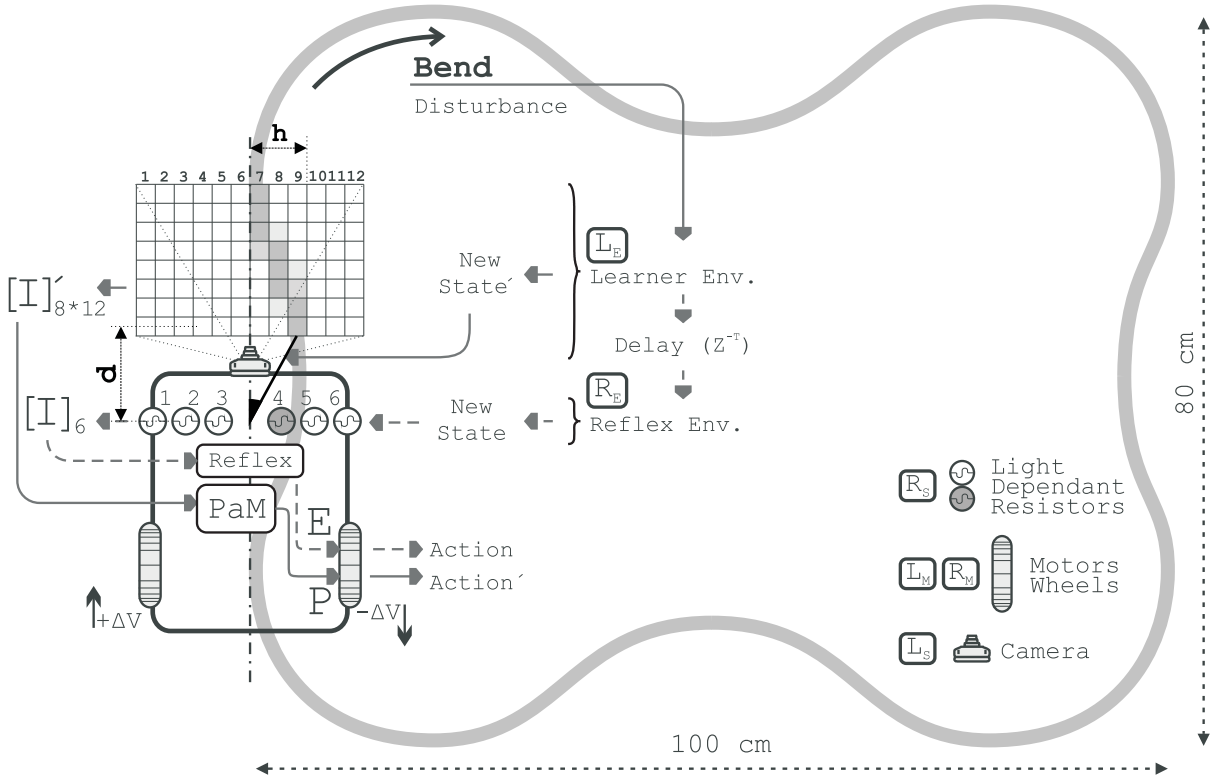

Figure 4.3: *This schematic diagram illustrates how the robot interacts with its environment. It depicts the reflex and learner components and how their inputs and outputs are connected to the environment. The diagram shows that a bend in the path is detected as a disturbance in the learner's field of view. This disturbance travels through the learner's loop, generating a predictive action. After a time delay of Z^{-T}, the disturbance reaches the reflex and passes through the reflex mechanism to produce a reflexive action. In the diagram, d represents the distance from the first point in the camera view to the error sensors, and h is the distance from the path to the centreline within the camera view. The angle of deviation is calculated and used as part of the modulating factor.*

Considering the derivative of the error signal, when $E > 0$, a positive change indicates a continued deviation, while a negative change implies a return to the centre. Conversely, when $E < 0$, a negative change suggests further deviation from the path, while a positive change implies returning to the centre. Consequently, the product of the error and its derivative, $E\frac{\partial E}{\partial t}$, can indicate performance deterioration when its value is positive, and conversely, signal improvement when negative. The exponential of this product represents the cue extracted from the reflex environment. This highlights the significance of learning

at each instance of the trial. We define the reflex clue R_C as below:

$$R_C = \mathcal{Z}\{r_c\} \tag{4.4}$$
$$r_c(e(t)) = exp\left(e(t) \cdot \frac{de(t)}{dt}\right)$$

Thus, the learning is mildly modulated when navigation is enhancing, while it becomes heavily modulated as navigation deteriorates.

The cue derived from the learner's environment is the angle of deviation. Referring to Figure 4.3, let d represent the distance from the first point in the camera view to the error sensors, and h be the distance from the path to the centreline within the camera view. The angle of deviation can be calculated as follows:

$$l_c(i') = \left|\arctan(\frac{h}{d})\right| \tag{4.5}$$

With this information, the modulating factor, as formulated in Equation 4.2, can be determined and computed.

4.5 Experimental Setup & Network Architecture

The experimental setup and common aspects of the network are detailed in Section 2.7. In this section, we use 48 predictors, thus the input layer of the network is initialised with 240 neurons. The network features a square topology, with 11 hidden layers and 11 neurons in each layer. This is the same architecture used for CLDL experiments.

4.6 Results

The performance of the PaM paradigm is compared with that of the CLDL paradigm, which serves as a benchmark, as discussed in Section 2.6. The following findings in this section refer to result (d) in Figure 1.1 in the preface.

4.6.1 Error Minimisation: Trial with $\eta = e^{-5}$

Figure 4.4A illustrates the error signal using solid lines and its corresponding moving average with dashed lines throughout a trial employing CLDL with a learning rate of $\eta = e^{-5}$. The success condition is achieved at time $t_1 = 266.3[s]$. In Panel B, the same information is presented for a trial utilising PaM, where the success condition is met at

time $t_2 = 23.1[s]$. The modulating factor F_M for the trial with PaM is depicted in Panel C. This set of trials illustrates a remarkable enhancement in both the speed of learning and the navigational performance of the robot with the application of PaM.

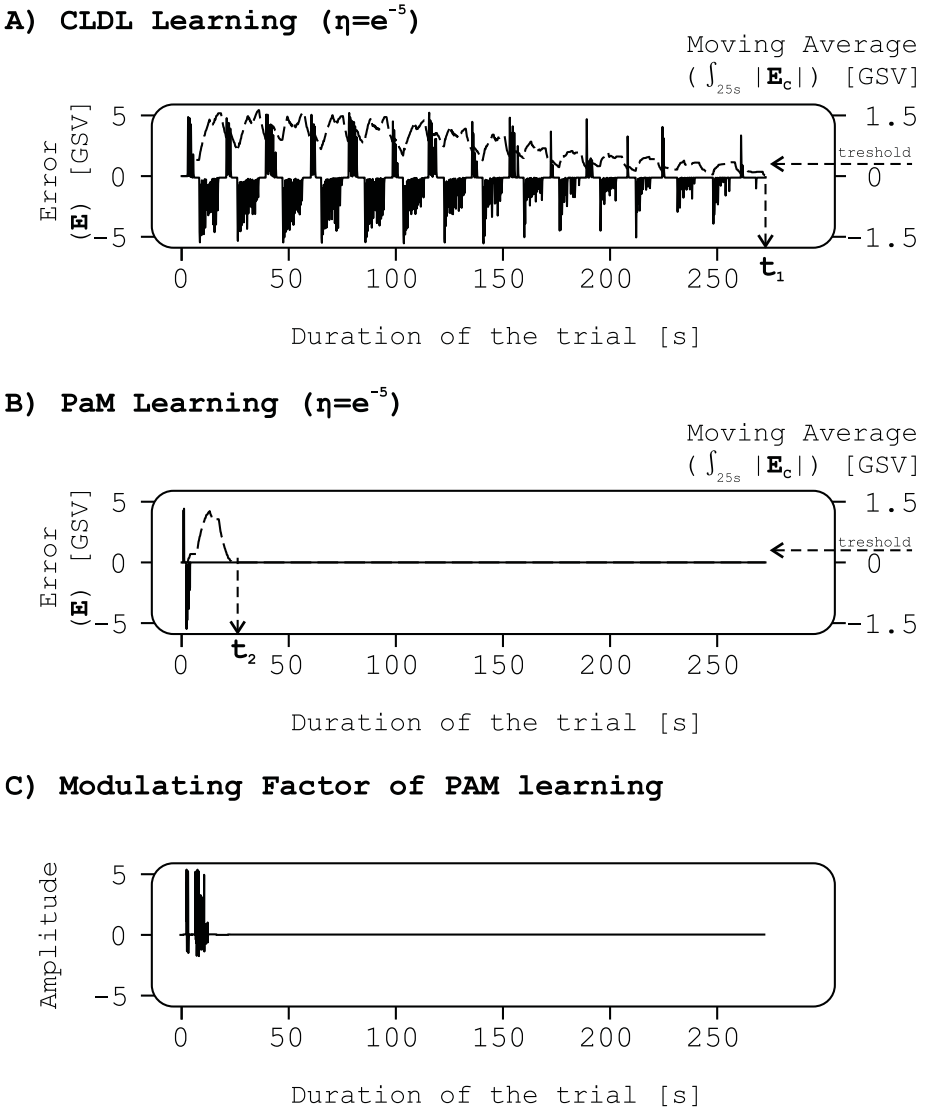

Figure 4.4: *Comparative trials were conducted using the CLDL and PaM paradigms on the physical robot, employing a low learning rate of $\eta = e^{-5}$.* A) *Displays the error signal (solid line, left y-axis) along with its moving average (dashed trace, right y-axis) against time (x-axis) for CLDL.* B) *Presents the same information for the PaM paradigm, and* C) *Illustrates the modulating factor for the PaM trial.*

4.6.2 Error Minimisation: Trial with $\eta = e^{-1}$

Another set of trials utilising a higher learning rate of $\eta = e^{-1}$ is presented in Figure 4.5 where remarkable results are demonstrated. Notably, this includes a one-shot learning scenario for PaM, with the success condition achieved at time $t_4 = 12.5[s]$. This outstand-

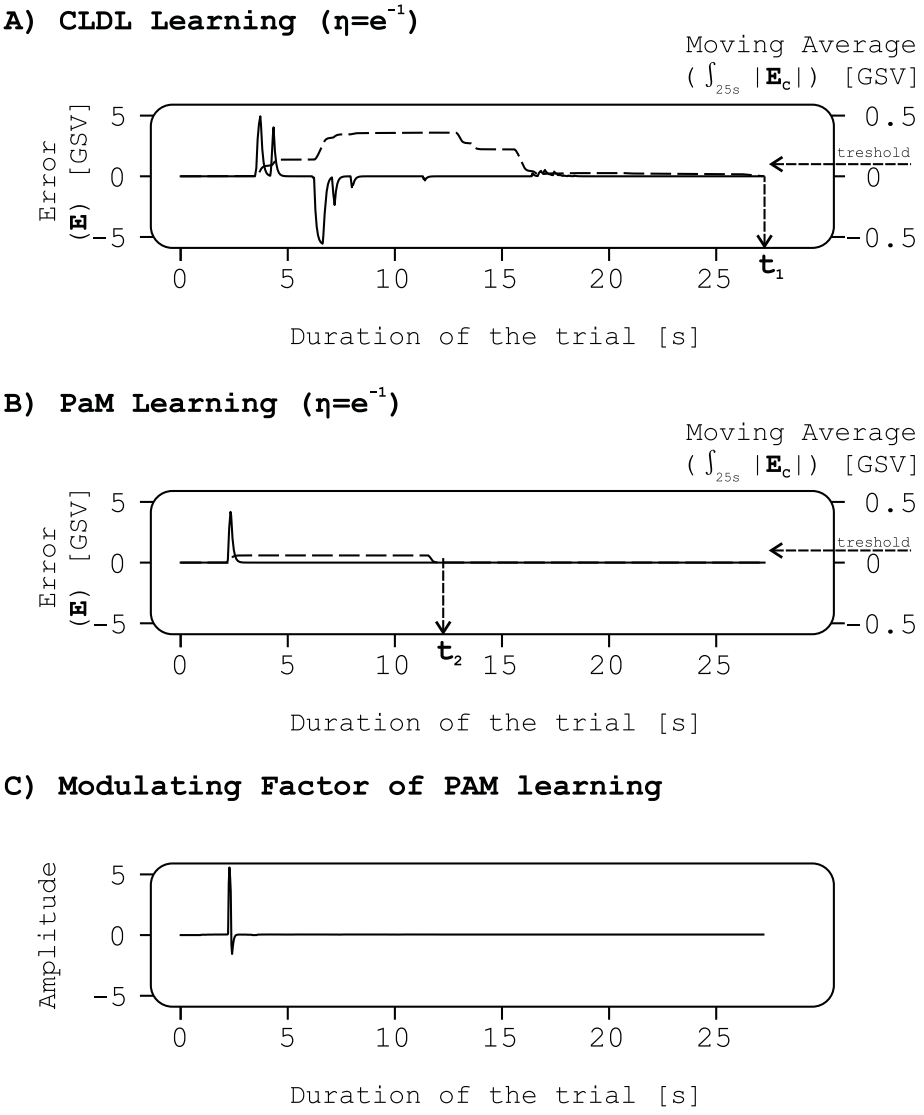

Figure 4.5: *Comparative trials were conducted using the CLDL and PaM paradigms on the physical robot, employing a high learning rate of* $\eta = e^{-1}$. A) *Displays the error signal (solid line, left y-axis) along with its moving average (dashed trace, right y-axis) against time (x-axis) for CLDL.* B) *Presents the same information for the PaM paradigm, and* C) *Illustrates the modulating factor for the PaM trial.*

ing performance stands in contrast to CLDL, where the success condition is attained at approximately $t_3 = 27[s]$.

4.6.3 Statistics & Reproducibility

These comparative trial pairs were replicated a total of 50 times across various learning rates: $\eta = [e^{-5}, e^{-4}, e^{-3}, e^{-2}, e^{-1}]$.

Figure 4.6A displays the total error integrals for trials utilising PaM (dotted trace) in comparison to those employing CLDL (dashed trace). As expected, the accumulation

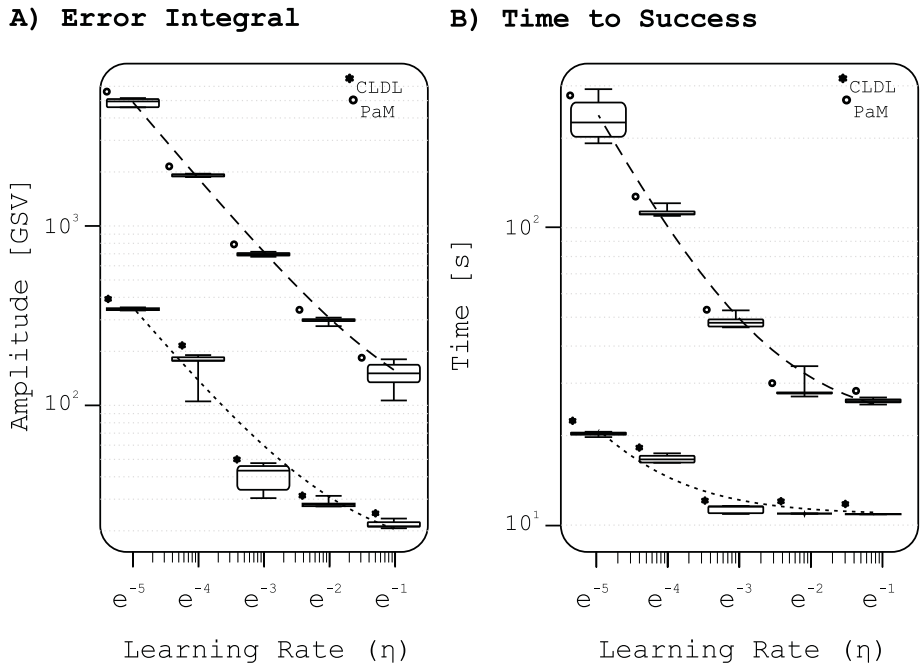

Figure 4.6: *Illustrates the reproducibility of results for the comparison of PaM (dotted lines) and CLDL (dashed lines) algorithms deployed on an actual robot.* A) *Presents the integral of error over the course of the trial across different learning rates.* B) *Depicts the time required to attain the success condition in seconds for varying learning rates.*

of the error signal is more pronounced in trials with slower learning rates. Although the total error accumulation is significantly smaller in PaM trials, both paradigms exhibit linear and equivalent influences, as evident from the slope of the fitted curves.

Figure 4.6B demonstrates the time taken for the robot to achieve the success state during trials using PaM (dotted trace) in contrast to those utilising CLDL (dashed trace). This graph reinforces the consistent trend observed in the preceding figures, highlighting that the PaM paradigm is notably faster than its counterpart. Both methods manifest accelerated learning with higher learning rates. However, the performance of CLDL is more significantly affected by variations in the learning rate, indicated by the curvature of the fitted curve.

4.7 Discussion

Compared to CLDL and SaR, the rapid convergence of PaM may render it susceptible to local minima. The biological authenticity of deep learning remains complex. A core concern revolves around the need for weight symmetry during forward and backward passes, restricting its feasibility to few layers (Lillicrap et al., 2016a). However, if errors are communicated solely through their sign, this symmetry constraint can be relaxed, provided there is appropriate interconnectivity ensuring the error's correct sign (Larkum,

2013). In neuroscience, this

suggests the ascending pathway governs processes like LTP and LTD. Simultaneously, neuromodulators, especially serotonin acting as a rectified reward prediction error, can modulate learning speed (Iigaya et al., 2018) as a third factor (Li et al., 2016). Given serotonin's prevalence over dopamine in cortical processing and profound neuronal architectures, the synergy of local and global learning methods appears promising for neuroscience and broader applications in machine learning and robotics.

Chapter 5

Forward Propagation Closed-Loop Learning (FCL)

5.1 Introduction

The forward propagation closed-loop learning (FCL) algorithm was introduced by Bernd Porr in 2020 (Porr and Miller, 2020). While it was not developed as part of this work, it is explained here as the subsequent chapter introduces the Echo algorithm, which drew inspiration from FCL. Moreover, comparing FCL with the CLDL algorithm presented earlier illustrates the versatility of their shared closed-loop learning platforms. Although these two algorithms exhibit significant differences, both demonstrate their efficacy and power within their respective contexts. While CLDL is characterised by mathematical robustness, FCL aligns more closely with biological plausibility.

5.2 Motivation

For nearly a century, neurophysiology has supported the idea of forward propagation, initially proposed by Hebb (Hebb, 2005). For LTP to occur, both pre- and post-synaptic neurons need to be active (Lüscher and Malenka, 2012). Unsupervised learning in these networks, particularly in the visual cortex, has been demonstrated in an open-loop fashion (Linsker, 1988; Song et al., 2000). However, these mechanisms alone may not suffice for training deep networks in closed-loop tasks.

The primary challenge with backpropagation is its requirement for backward-directed information, necessitating a second set of connections. Although reverse connections need not be symmetric with forward weights (Lillicrap et al., 2016b), backward-projecting weights introduce assumptions that may not hold in deeper architectures.

Emerging evidence suggests a mechanism for training deep networks in closed-loop settings. Different brain oscillations can carry distinct information through the same

neurons, allowing forward transmission of separate streams (Mizuhara and Yamaguchi, 2007; Canolty and Knight, 2010). These streams exert different effects on plasticity; higher-frequency components are likely to alter plasticity (Bliss and Lømo, 1973), while lower-frequency components maintain plasticity relevant to behaviour or mental processes (Mizuhara and Yamaguchi, 2007). This could enable biological transmission of both activity and error signals using the same weights.

5.3 Design & Derivation of FCL Algorithm

Figure 5.1 illustrates the FCL platform. The reflex and learning loops function as previously described in Section 2.4.

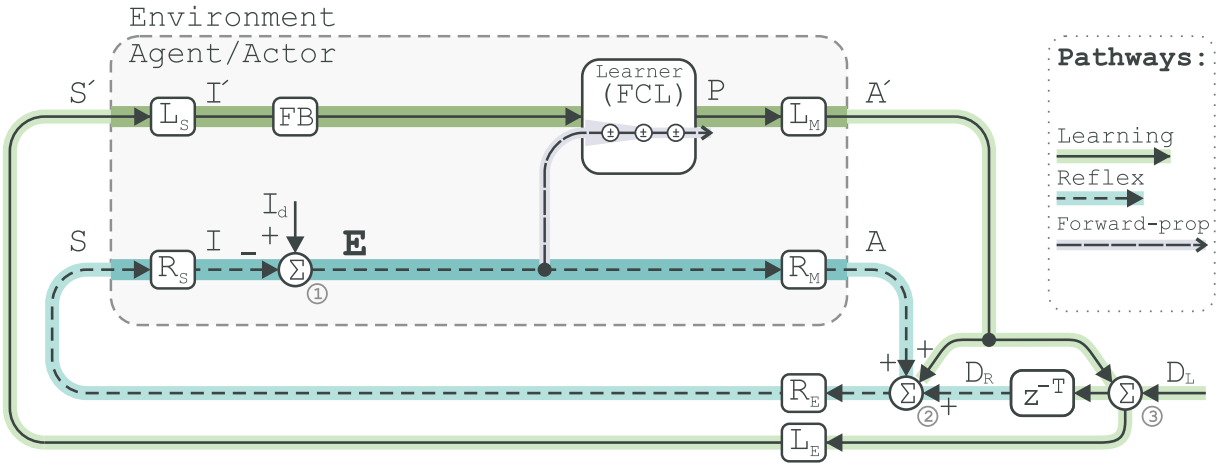

Figure 5.1: *(redrawn from (Porr and Miller, 2020)) Illustrates the integration of the FCL learner into the closed-loop platform. The reflex loop is depicted in blue, and the learning loop is represented in green. The error signal is received by the network at its input via the purple pathway.*

The forward propagation of predictive inputs adheres to the conventional rules outlined in Section 2.3. This process is depicted in Figure 5.2 with solid arrows highlighted in blue. The error signal is received at the input layer and is subsequently propagated forward using the same weights associated with the predictive inputs. This feedback pathway is illustrated in Figure 5.1 and Figure 5.2 with purple highlights.

With this, the internal error of the neurons can be defined as:

$$\delta_j^\ell = \frac{\Sigma_{i=0}^{I}(\omega_{ij}^\ell \delta_i^{\ell-1}) \cdot \sigma'(v_j^\ell)}{\frac{\Sigma_{i=0}^{I}\omega_{ij}^\ell}{\Sigma_{i=0}^{I}1}} \tag{5.1}$$

The internal error is normalised by the total sum of the weights and the number of

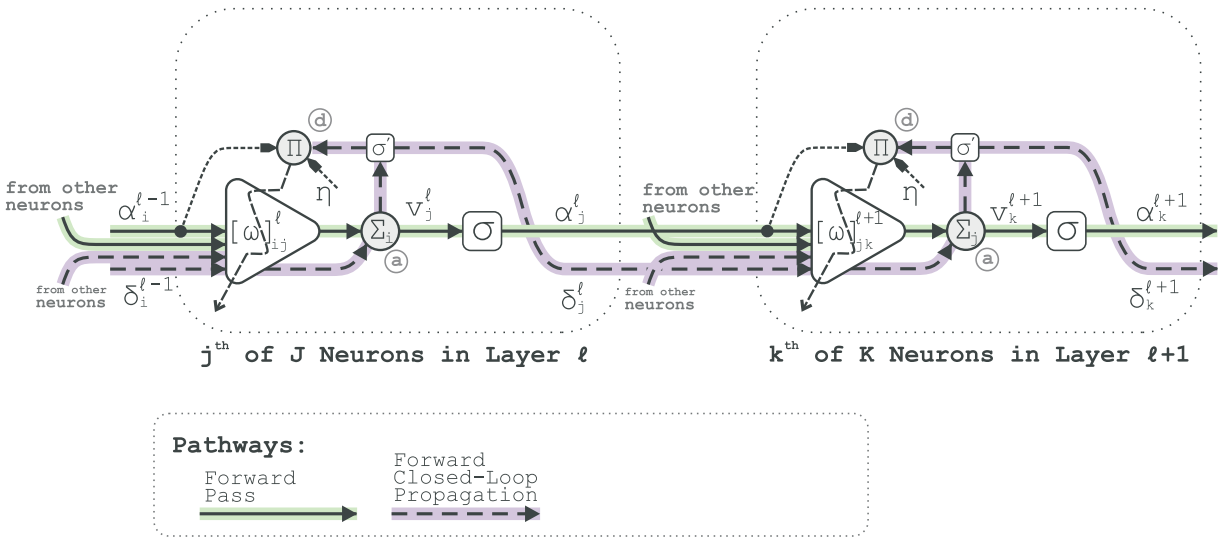

Figure 5.2: *(redrawn from (Porr and Miller, 2020)) Illustrates the internal connections of neurons in the FCL network. The blue pathway illustrates the forward propagation of predictive inputs, while the purple pathway demonstrates the forward propagation of the closed-loop error.*

neurons in that layer. It is important to note that the internal error of layer ℓ depends on the weighted sum of this term in layer $\ell - 1$. This is in contrast to the calculation of internal error for the conventional backpropagation algorithm, as shown in Equation 2.17, where the internal error calculation relies on the weighted sum of the term in the deeper layer $\ell + 1$.

Hence, in the first layer, the closed-loop error is used to initiate this propagation, which means $\delta_x^0 = E$. Finally, the learning rule for FCL is defined as:

$$\Delta\omega_{ij}^{\ell} = \eta\delta_j^{\ell}(z)\alpha_i^{\ell-1}(-z), \quad \eta << 1 \tag{5.2}$$
FCL

This weight change occurs at node ⓓ in Figure 5.2 (Porr and Miller, 2020).

5.4 Experimental Setup & Network Architecture

The experimental setup and shared features of the network are explained in Section 2.7. In this section, we employ 48 predictors, resulting in the input layer of the network being initialised with 240 neurons. The network is designed with a square topology, consisting of 11 hidden layers, each containing 11 neurons. This architecture is consistently used for the CLDL experiments.

5.5 Results

This section compares the FCL and CLDL algorithms. These results were gathered using the line-follower robot in simulation as explained in previous sections. The following findings in this section refer to result (e) in Figure 1.1 in the preface.

5.5.1 Error Minimisation

Figure 5.3 examines the error signals during two trials for these algorithms with a learning rate of e^{-1}. Panel A illustrates this data for a trial using the FCL algorithm, while Panel B showcases the same for the CLDL algorithm. CLDL exhibits a remarkable ability to expedite error minimisation in contrast to FCL. Specifically, CLDL accomplishes successful learning in 12.8 seconds, whereas FCL takes a significantly longer 55.6 seconds to reach a similar level of error reduction. Notably, the small arrows within the figures point out minuscule error magnitudes that might otherwise go unnoticed. Furthermore, the total average of the error signal during each trial was calculated: FCL generates only 0.7k units of error, while CLDL accumulates a more substantial 8.4k units of error.

5.5.2 Euclidean Weight Distance & Convergence

In these experiments, the neural network was initialised with 10 hidden layers. Figure 5.4 offers a visualisation of the weight changes observed during trials with both the FCL and CLDL algorithms, each utilising a learning rate of e^{-1}. Panel A showcases the Euclidean distance of weights from their initial random initialisation in the context of the FCL algorithm, while Panel B does the same for the CLDL algorithm.

Distinct traces for the weight changes in each layer, allowing for a detailed analysis. In the FCL network, the overall weight changes are substantially more pronounced, reaching levels of approximately 250 units. Conversely, within the CLDL network, we observe significantly smaller weight changes, with a maximum value of only 35 units. This could indicate that FCL is more prone to instability or overfitting. Furthermore, the weight change patterns exhibit intriguing characteristics. In the FCL network, the weight changes vary distinctly from layer to layer, while the CLDL network displays a discernible pattern, with layers falling into two distinct clusters, each exhibiting similar weight change profiles.

5.5.3 Input Layer Weight Distribution

In these experiments, three rows of predictors were utilised, each consisting of 48 units. Each predictor was processed through five distinct FIR filters, culminating in a total of 720 inputs for the neural network. Consequently, the weight matrix in the network's first

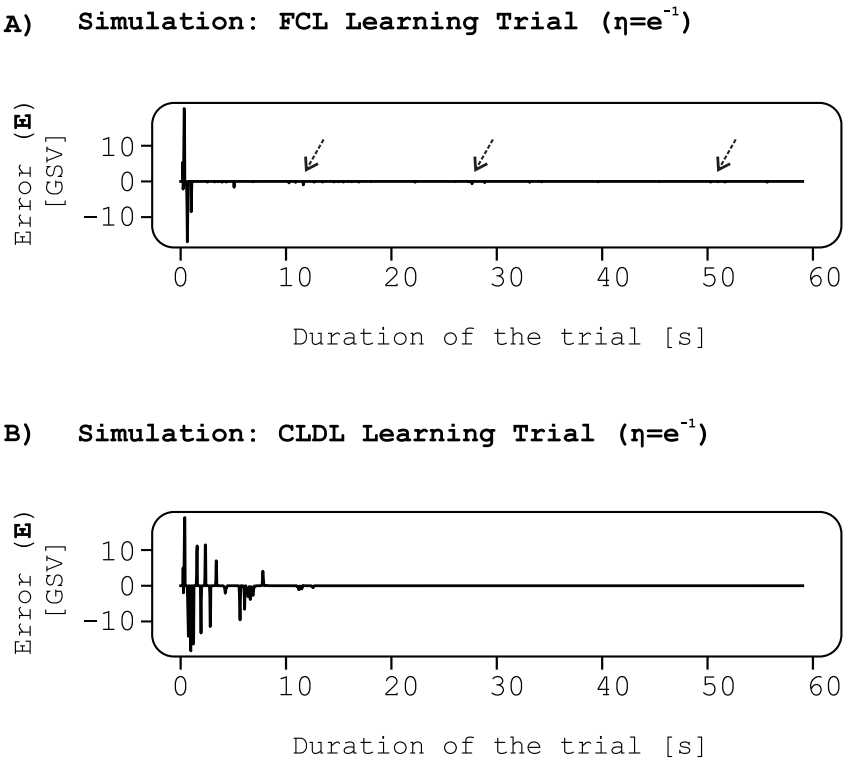

Figure 5.3: *Comparing the error signals from trials conducted with the FCL and CLDL algorithms, both employing a learning rate of* e^{-1}. A) *shows the error signal during a learning trial with the FCL algorithm. The success condition is achieved after 55.6 seconds of learning.* B) *shows the error signal during a learning trial with the CLDL algorithm. The success condition is reached much faster, within 12.8 seconds. The small arrows highlight subtle error spikes.*

layer comprises 720 by 11 elements. Upon achieving successful learning, the final values of these weights were documented and depicted as a greyscale image in Figure 5.5.

Panel A illustrates the weight distribution for the FCL, while Panel B displays the same for the CLDL. It is observable that both algorithms have identified the three rows of predictors by organising the weights into three segments with analogous patterns. Within each segment, the weights attributed to the signals filtered from the predictors on the outermost column are notably higher (resulting in a darker appearance in the greyscale image). In contrast, signals filtered from the innermost column of predictors are assigned lower weight values, making them appear lighter in the greyscale representation. This arrangement is indicative of the network's tendency to generate more pronounced steering responses for significant deviations, typically observed when the path is detected on the outer edges of the camera's field of view. This activates the outermost predictors. Inversely, the network produces more subtle steering responses for minor deviations, detected when the path is nearer to the centre of the camera view, thus activating the innermost predictors.

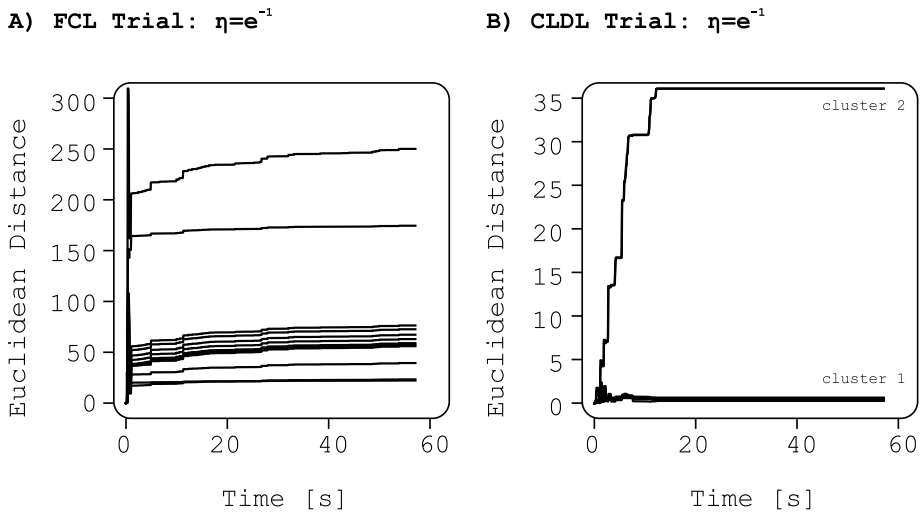

Figure 5.4: *Comparing the weight changes during trials conducted with the FCL and CLDL algorithms, both employing a learning rate of* e^{-1}. A) *Shows the Euclidean weight distance during a trail with FCL with the highest distance reaching around 250 units.* B) *Shows the Euclidean weight distance for CLDL algorithm, with maximum distance of only 35 units.*

However, this pattern is more pronounced in the FCL algorithm compared to the CLDL, where a seemingly random distribution is still evident. This observation correlates with the previously noted higher weight change in FCL in Figure 5.4, suggesting its greater divergence from the initial random distribution than CLDL. This observation is expected as FCL correlates the error with the visual input from the camera in the input layer.

5.5.4 Statistics & Reproducibility

To thoroughly evaluate the reproducibility of outcomes observed in prior experiments involving FCL and CLDL, a series of additional trials were executed, each varying in learning rates. The primary objective of this assessment was to compare the efficacy of FCL and CLDL using two critical performance metrics: the time required to attain successful learning and the average total error accumulated during this learning phase. Figure 5.6 represents these metrics for both FCL and CLDL across a spectrum of learning rates, specifically $\eta = \{e^{-3}, e^{-2}, e^{-1}, e^{0}, e^{1}, e^{2}\}$.

Panel A of the figure demonstrates that FCL consistently achieves lower overall error rates across the range of learning rates when compared to CLDL. Panel B shows the time duration each algorithm requires to reach a state of successful learning. CLDL tends to learn more rapidly than FCL.

Note that at high learning rates of $\eta = \{e^{1}, e^{2}\}$, the learning process becomes unstable. In these cases, the behaviour of the network is unreliable, with some trials resulting in success while others result in failure. This explains the inconsistent results seen at these

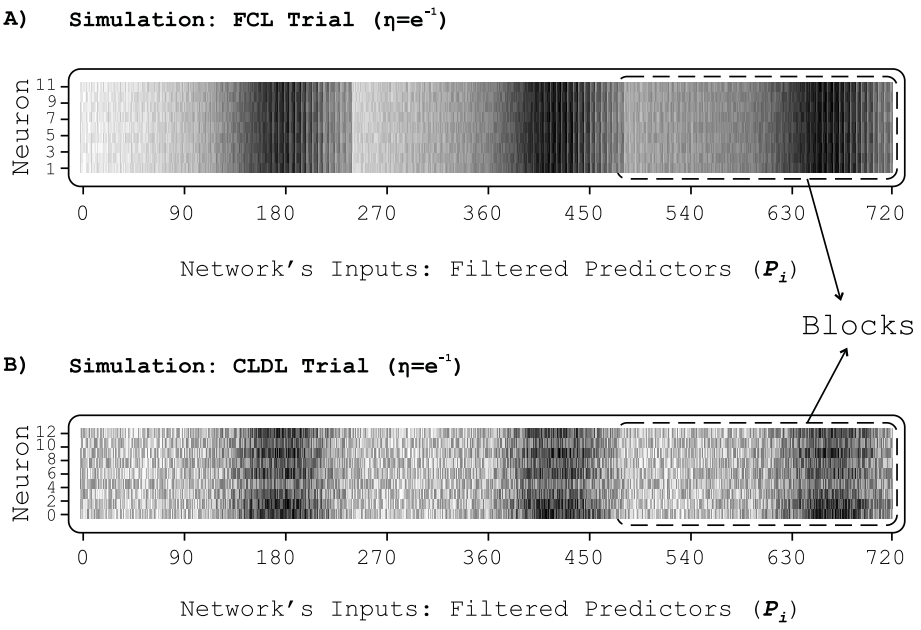

Figure 5.5: *Displays the distribution of weights in the initial layer of the neural network during the trials with FCL and CLDL in a simulated environment. The vertical axis indicates the neuron indices, whereas the horizontal axis denotes the indices of inputs, namely the filtered predictors. The greyscale colour map is used to represent the final weights, where black indicates the highest weight value and white the lowest. Each distinct segment corresponds to a row of predictors.* A) *Presents the distribution of weights for a trial using FCL.* B) *Illustrates the weight distribution in a trial involving CLDL. Both experiments were conducted with a learning rate of* e^{-1}.

learning rates.

These findings provide valuable insights into the learning dynamics and efficiency of the FCL and CLDL algorithms, contributing to a deeper understanding of their practical applications in neural network training.

5.6 Discussion

Comparing CLDL and FCL reveals that both separately propagate errors and activations with shared weight values but differ in the error propagation direction.

The ICO learning method (Porr and Wörgötter, 2006) learns by correlating an error signal at its input with an activation. Unlike FCL, ICO sums the error signal with the weighted activation, allowing direct equivalence with reflex actions. However, intertwining error and learned signals may lead to information loss, making it unsuitable for deep networks (Kulvicius et al., 2007). This intertwining implies direct behavioural implications for both signals. Conversely, FCL exhibits greater adaptability, where the error signal becomes increasingly adaptable as it moves through layers, avoiding information loss.

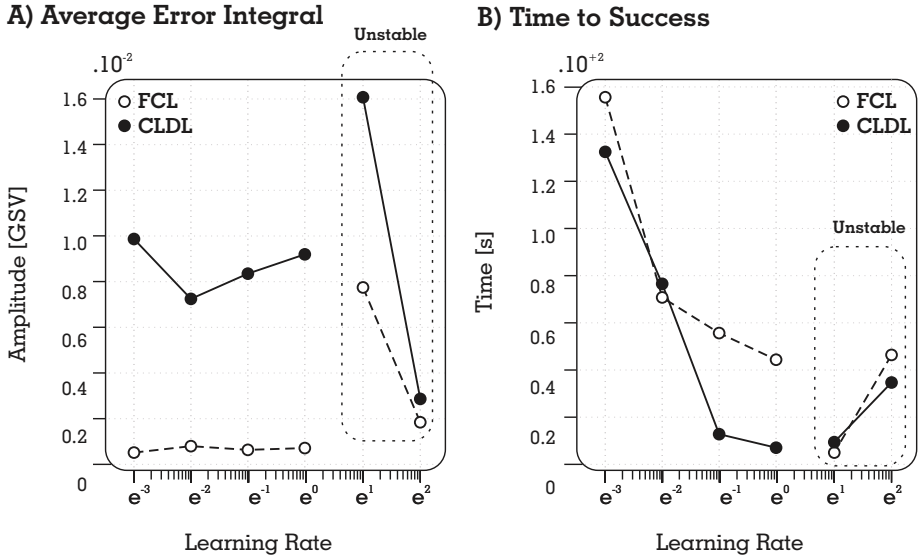

Figure 5.6: *Comparative evaluation of FCL and CLDL performance metrics at learning rates of* $\eta = \{e^{-3}, e^{-2}, e^{-1}, e^{0}, e^{1}, e^{2}\}$. A) *Illustrates the average total error, showing that FCL maintains lower error totals across all learning rates.* B) *Depicts the time to success, showing that CLDL generally learns faster.*

Neurophysiological debates often centre on plasticity, with high postsynaptic calcium concentration leading to LTP (Malenka et al., 1999; Bennett, 2000), and lower concentration resulting in LTD (Mulkey and Malenka, 1992). Achieving pronounced postsynaptic activity necessitates robust presynaptic drive, resulting in calcium influx (Meunier et al., 2017). This scenario might cause synaptic weight to continuously amplify itself. However, if the learning signal and actual activity pass through the same synapse but remain distinct (Lindsay et al., 2017) by employing varied frequencies, FCL emerges as a promising mechanism. It facilitates stable learning relevant to behaviour, steered by heterosynaptic plasticity and Hebbian learning for the error signal, ensuring consistent adjustments.

Chapter 6

Echo Learning: Bi-Directional Error Propagation

6.1 Introduction

This chapter introduces the Echo learning paradigm. Following a brief motivation, we present the learning rule, followed by the results and discussion.

6.2 Motivation

The inspiration for this paradigm emerged from the robust mathematical foundation underlying error backpropagation in CLDL and the biologically plausible rationale behind the forward propagation of error in FCL. The concept of Echo learning naturally evolves from synergising these two well-established paradigms.

In Echo learning, the idea is to leverage the strengths of both these paradigms. The error signal does not merely propagate in a unidirectional manner; instead, it oscillates back and forth until its magnitude diminishes to a negligible value, which is then discarded. This iterative process occurs in each learning iteration. Therefore, Echo learning effectively harnesses the capabilities of both CLDL and FCL.

Specifically, the instability observed in FCL can be mitigated in Echo learning due to its incorporation of backpropagation. Likewise, the exploding and vanishing gradient problem (EVGP) observed in CLDL can be effectively managed in Echo learning, as errors are propagated multiple times through FCL.

6.3 Design & Derivation of Echo Algorithm

Figure 6.1 illustrates the learning platform for the Echo algorithm. The reflex and learning loops were explained in Section 2.4. The feedback pathway is presented in purple, show-

casing the bidirectional flow of the closed-loop error signal, encompassing both backward and forward propagation.

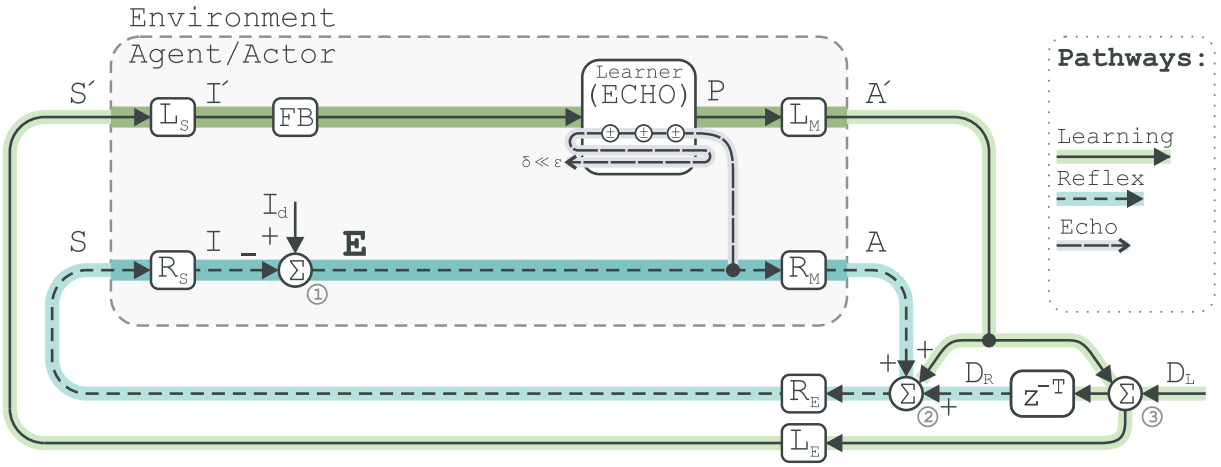

Figure 6.1: *Depicts the arrangement of the Echo learner on a generic learning platform. The blue loop illustrates the reflex pathway, while the green loop represents the learning pathway. The purple pathway showcases the bidirectional utilisation of the closed-loop error for the learning process.*

The distinctive feature of Echo learning lies in the fact that the weights undergo multiple rounds of adjustments until a specific condition is satisfied. This is in contrast to the paradigms discussed earlier, where weight changes occur once per cycle of the closed-loop platform. In the initial step, the network undergoes a round of weight changes dictated by the CLDL learning rule as described in Equation 2.54. The propagation according to CLDL terminates after causing weight changes in the first layer. The internal errors of neurons in the first layer are then propagated forward through the network to induce a second round of weight adjustments based on the FCL learning rule as in Equation 5.2. This recursive process repeats until the total sum of absolute weight changes in the terminating layer (first layer for CLDL and last layer for FCL) approaches zero $\Sigma_{ij}|\Delta\omega^{0\ or\ L}| << \epsilon$.

This conditional cycle can be represented mathematically as follows:

$$\xleftarrow[terminate]{\Sigma_{ij}|\Delta\omega^0|<<\epsilon} \underset{\text{CLDL}}{\Delta\omega}\Big|_{L\to 0} \xrightarrow[\Sigma_{ij}|\Delta\omega^0|>>\epsilon]{\Sigma_{yz}|\Delta\omega^L|>>\epsilon} \underset{\text{FCL}}{\Delta\omega}\Big|_{0\to L} \xrightarrow[terminate]{\Sigma_{yz}|\Delta\omega^L|<<\epsilon} \tag{6.1}$$

Here, ϵ represents the threshold set for terminating the learning process. In this study, it was established as 10^{-6}.

6.4 Experimental Setup & Network Architecture

The experimental setup and the network's common features are described in section 2.7. In this section, we utilise 48 predictors. The network employs an encoder topology, featuring 10 hidden layers that decrease linearly in the number of neurons: $\{\overbrace{240}^{\text{Input}}, \overbrace{13, 12, ..., 5, 4,}^{\text{Hidden}} \overbrace{3}^{\text{Output}}\}$.

6.5 Results

This section presents the results of a trial employing the Echo learning paradigm and compares it to the outcomes of trials using CLDL and SaR paradigms. The following findings in this section refer to result (f) in Figure 1.1 in the preface.

6.5.1 Error Minimisation

Figure 6.2 depicts the error signals using solid black traces and their corresponding moving averages with dashed traces, during $100[s]$ trials with a learning rate of $\eta = e^{-1}$ for the aforementioned paradigms. In Panel A, CLDL establishes a benchmark for evaluating Echo learning. It showcases a gradual reduction of the error signal in less than $90[s]$. The error signal spikes reach approximately $-10[GSV]$. The moving average of the error peaks at just under $1[GSV]$ around $20[s]$ and progressively diminishes until it falls below the threshold value for success condition at roughly $95[s]$, signifying the attainment of the success condition.

As seen in Section 3, SaR learning demonstrates quicker learning than CLDL. Panel B reinforces this observation as a trial with SaR exhibits error signal spikes lasting less than $20[s]$, with values reaching around $-5[GSV]$. The moving average of the error reaches its peak value of $0.5[GSV]$ around $15[s]$, and the success condition is met at about $25[s]$. This marks a noteworthy enhancement over CLDL performance.

Panel C presents the results for Echo learning, which outperforms even the SaR learner. The error spikes persist for only $10[s]$ but reach lower values of $-7[GSV]$. The moving average of the error reaches its peak of under $0.5[GSV]$ at around $10[s]$, and the success condition is achieved at $20[s]$. This outcome aligns with expectations, given that Echo learning leverages the strengths of both CLDL and FCL. However, the rapid learning speed raises concerns about stability and potential limitations related to local minima.

6.5.2 Euclidean Weight Distance & Convergence

Examining the weight changes provides insights into the stability of learning. Figure 6.3 illustrates the Euclidean distance of weights in the first layer for the trials mentioned above.

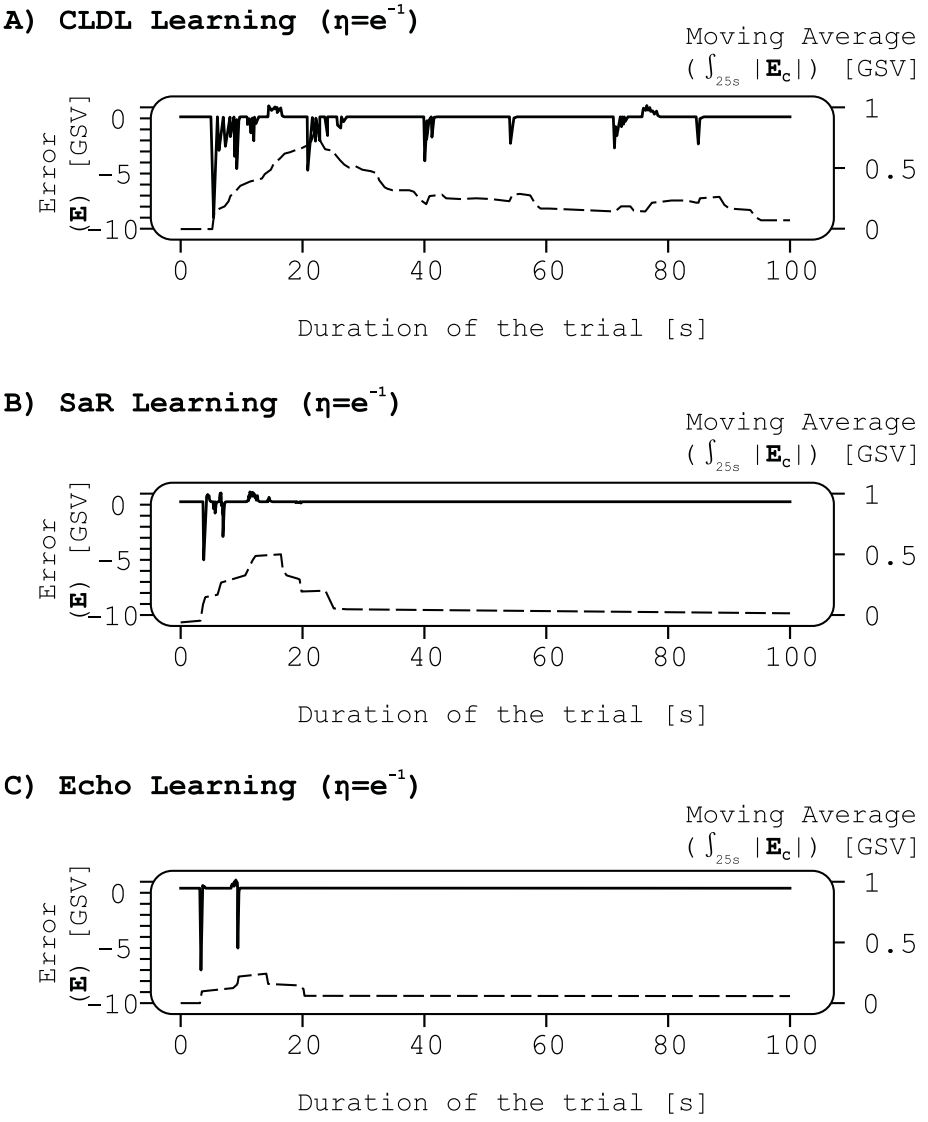

Figure 6.2: *Illustrates Comparative Trials involving CLDL, SaR, and Echo paradigm with learning rate of* $\eta = e^{-1}$*, deployed on a real robot. The error signals are presented with solid lines on the left y-axis, while their corresponding moving averages are depicted with dashed lines on the right y-axis, both in terms of GSV. The x-axis represents the duration of the trial in seconds.* A) *Depicts this information for CLDL.* B) *Demonstrates the same for SaR, and* C) *Presents this for Echo learning.*

In Panels A and B, CLDL and SaR exhibit total weight distances of approximately 30 and 45 respectively. This comparison indicates that the improved performance of SaR learning comes with a moderate increase in weight distance. However, for Echo learning, depicted in Panel C, the Euclidean distance of weights swiftly reaches its maximum value of 90 within the initial 20$[s]$ of the trial. This represents a significant increase compared to both CLDL and SaR. This suggests that Echo learning might be susceptible to unstable weight changes or potential convergence to local minima.

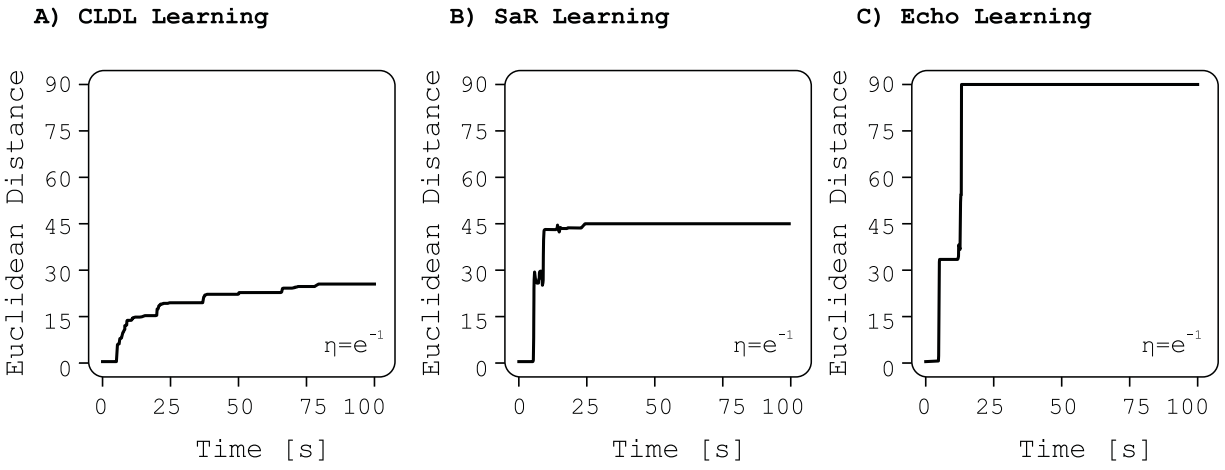

Figure 6.3: *Illustrates the Euclidean Weight Changes for trials involving CLDL, SaR, and Echo learning, all with a learning rate of* $\eta = e^{-1}$. A) *Depicts this information for CLDL.* B) *Demonstrates the same for SaR and* C) *Presents this for Echo learning.*

subsectionInput Layer Weight Distribution

Analysing the weight distribution in the first layer after meeting the success condition can provide further insights into the performance of these paradigms. Figure 6.4 presents this information for the trials discussed above.

In Panels A and B, the weight distributions for CLDL and SaR respectively exhibit trends consistent with those observed in Sections 2 and 3, respectively. In summary, the emergence of blocks and the gradients within each block demonstrate that the network has effectively assigned significance to each predictive signal. However, SaR learning displays a more distinct definition of these characteristics.

Panel C portrays this information for Echo learning. A distinctive observation here is the apparent shift of gradients within each block to the left. In other words, within each block, from left to right, the columns transition from white to black and then back to grey. This phenomenon can be attributed to Echo algorithm's remarkably fast learning, which has limited its exploration of the environment. Echo learning has predominantly employed only a few predictors for navigation, resulting in the grey colouration of other unused predictors, that appears after the black columns. This observation supports the notion that the Echo learner might be prone to converging to local minima and may not fully explore the entirety of the environment. However, such behaviour can be advantageous in specific applications.

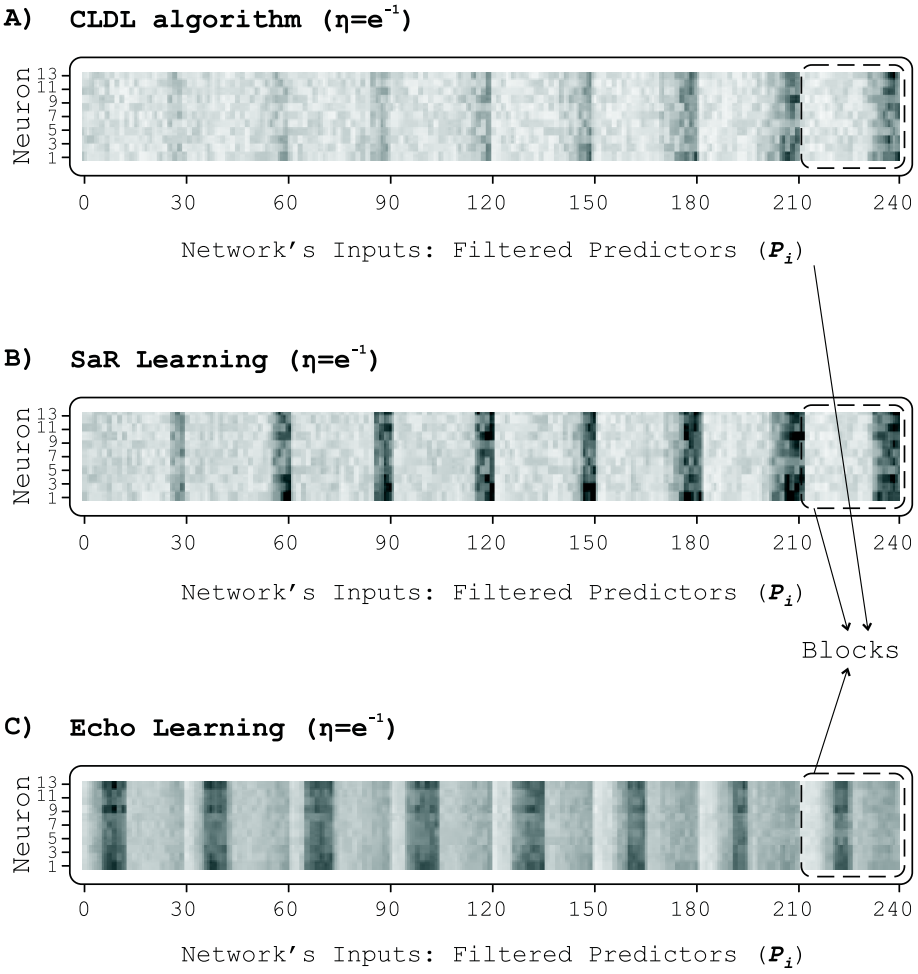

Figure 6.4: *Displays the distribution of weights in the first layer after the trial has concluded. The y-axis represents the index of neurons in the first layer, while the x-axis denotes the index of the filtered predictive inputs. The weight values are translated into a greyscale image, where black corresponds to the maximum value and white to the minimum. Each original predictor and its filtered signals constitute a block within this image.* A) *Depicts this information for CLDL.* B) *Demonstrates the same for SaR, and* C) *Presents this for Echo learning.*

6.6 Flexible Library for Future Algorithm Development

The evolution of the Echo learning algorithm, a type of reinforcement learning algorithm, led to the development of a versatile C++ library. Throughout the project, this library was continuously enhanced to support innovative and experimental learning methods.

One of the key features of this C++ library is its flexibility in handling error signals. These signals can be injected at any layer of the neural network and can propagate in multiple directions across numerous parallel streams. This allows for a highly customisable approach to error correction and learning.

Furthermore, the library enables the creative combination of internal error signals to adjust the weights within the neural network. These adjustments can occur once or multiple times during each iteration of the closed-loop learning platform. This flexibility makes the library a powerful tool for exploring and implementing a wide range of learning algorithms.

6.7 Discussion

Echo learning features a bidirectional oscillation of the error signal, ensuring thorough error minimisation. This iterative process enhances learning accuracy and reduces the likelihood of converging to local minima. Integrating backpropagation within this hybrid approach mitigates the instability typically observed in forward propagation closed-loop learning (FCL), stabilising the learning process and preventing divergence.

Echo learning mirrors the biological process observed in humans and other organisms when encountering new learning scenarios. Instead of learning from a single data point or experience in isolation, organisms reflect on and reprocess the experience multiple times to draw broader conclusions. For instance, if we touch a hot surface, we learn from the immediate pain and mentally iterate the experience to conclude that touching the surface with the other hand or from different angles would yield the same painful result. We do not need to physically repeat these actions; our brains extrapolate the outcomes through cognitive iterations. Similarly, Echo learning revisits each learning experience multiple times, refining its understanding and enhancing learning from a single instance.

This biological analogy is further supported by the fact that internal errors in the input layer still contain information about the predictive inputs. These errors can be seen as non-linear filtered versions of the original inputs that have propagated through the network. Consequently, these internal errors retain different forms of the same information and can be used to further train the network.

Echo learning also addresses the vanishing gradient problem (VGP) common in deep learning (DL). By propagating errors multiple times, the algorithm keeps gradients within a manageable range, improving training reliability, particularly in deep networks. The mathematical representation of the learning rule provides a clear framework for the termination condition. The threshold ϵ plays a crucial role in balancing convergence speed and accuracy. While the additional iterations increase computational complexity, the benefits in terms of stability and accuracy justify the extra computational effort.

Furthermore, Echo learning scales well for larger and more complex neural network architectures and is suitable for a wide range of applications, from simple classification tasks to complex scenarios like reinforcement learning and autonomous systems. Future research could focus on optimising the threshold ϵ, experimenting with different network architec-

tures, and exploring the impact of Echo learning on various types of neural networks, such as convolutional and recurrent neural networks. Studies on real-world applications could provide practical insights and guide further improvements.

Chapter 7

Deep Neuronal Filter (DNF)

7.1 Introduction

In the previous chapters, we explored six closed-loop applications of the proposed algorithms. This section presents the adaptability of this learning platform for open-loop applications. An open-loop platform essentially omits the environment, not in a physical sense, but in terms of its functional role in the feedback system. In other words, the agent's actions do not influence its subsequent sensory inputs. Signal processing stands as one of the most intriguing use cases for open-loop learning systems, where the system takes in raw data and generates desired variations in that data.

7.2 Motivation

In a variety of fields such as communications, acoustics, and biomedical engineering, the challenge of low signal to noise ratio (SNR) is prevalent. This is particularly evident in electroencephalogram (EEG) measurements (Green et al., 1985; Henry, 2006; Britton et al., 2016), which are characterised by low amplitude signals in the range of a few μV. These signals are often overwhelmed by interference from sources significantly stronger than the EEG signals (Fatourechi et al., 2007). This section focuses on EEG as a case study, aiming to eliminate non-stationary electromyogram (EMG) noise. However, the techniques for enhancing SNR proposed here are applicable to beyond just EEG signals.

Two primary methods exist for enhancing the SNR of EEG signals: real-time processing and offline post-processing. The latter is predominantly achieved through principal component analysis (PCA) or independent component analysis (ICA) (McMenamin et al., 2010; Fitzgibbon et al., 2007; Delorme et al., 2007). Both PCA and ICA techniques initially examine the unprocessed signals to distinguish and segregate signal and noise components. This process, conducted offline, necessitates steady signal and noise correlations over time and requires significant computational resources.

In contrast, real-time algorithms process EEG signals as they are received, on a sample-by-sample basis, without the need for pre-analysis. Examples of such methods include bandpass filters, the short-time fourier transform (STFT), and wavelet transform (Ahmadi et al., 2012; Jirayucharoensak et al., 2013, 2019). These techniques, however, still require some prior knowledge of the noise characteristics to appropriately adjust the filter settings. Muscle noise presents a particular challenge due to its non-stationary nature, stemming from both voluntary and involuntary muscle contractions in the facial area.

When EEG electrodes are positioned on top of the head, around the C_z region, it is often assumed that the noise affecting the EEG signals comes from external sources and impacts all electrodes uniformly. In contrast, EEG signals are believed to be generated locally (Fitzgibbon et al., 2015). To address this, a secondary, auxiliary electrode is employed exclusively to measure the noise, which can then be subtracted from the primary EEG electrode's signal. A commonly used design for this auxiliary electrode is a ring-shaped configuration surrounding the main EEG electrode. This method, where noise is simply subtracted from the signal, is known as the "Laplace operator" (Makeyev et al., 2016; Fitzgibbon et al., 2015; Garcia-Casado et al., 2019; Aghaei-Lasboo et al., 2020; Besio et al., 2006). While the concept of straightforward noise subtraction is theoretically sound, the practical application is more complex due to the dynamic and intricate relationship between brain-generated EEG and the signals captured at the electrodes. This complexity necessitates the use of a sophisticated, composite electrode that incorporates an adaptive filter, continuously learning and adjusting to the ever-changing signal and noise conditions.

This section introduces a pioneering proof of concept for an innovative, cost-effective, and easily manufacturable compound electrode. This electrode is paired with a novel deep learning algorithm, called deep neuronal filter (DNF), to form a system that adaptively filters out noise from EEG signals. The system operates by algorithmically generating a counter-signal that opposes the noise, effectively neutralising it. The efficacy of this approach is demonstrated through the successful elimination of wideband muscle EMG noise from EEG recordings.

The outcome of this collaborative effort is documented in a publication in Plos One (Porr et al., 2022). Notably, the establishment of the setup and the experimental procedures were conducted by two undergraduate students, Henry Cowan and Lucia Munoz Bohollo.

7.3 Design & Derivation of DNF Algorithm

In this section, we delve into the functionality and characteristics of the open-loop learning platform in a broad context. Subsequent sections detail the application of this platform to noise cancellation scenarios.

Figure 7.1 illustrates an open-loop platform designed for real-time signal processing employing deep-learning. It bears resemblance to the closed-loop platform that served as the foundation for all 6 closed-loop algorithms explained in preceding chapters. In this instance, however, the transfer functions of the reflex and learner environments, denoted as R_E and L_E in Figure 2.9, have been omitted. This alteration renders the previously interlocked reflex and learning loops now "open", thus the term "open-loop".

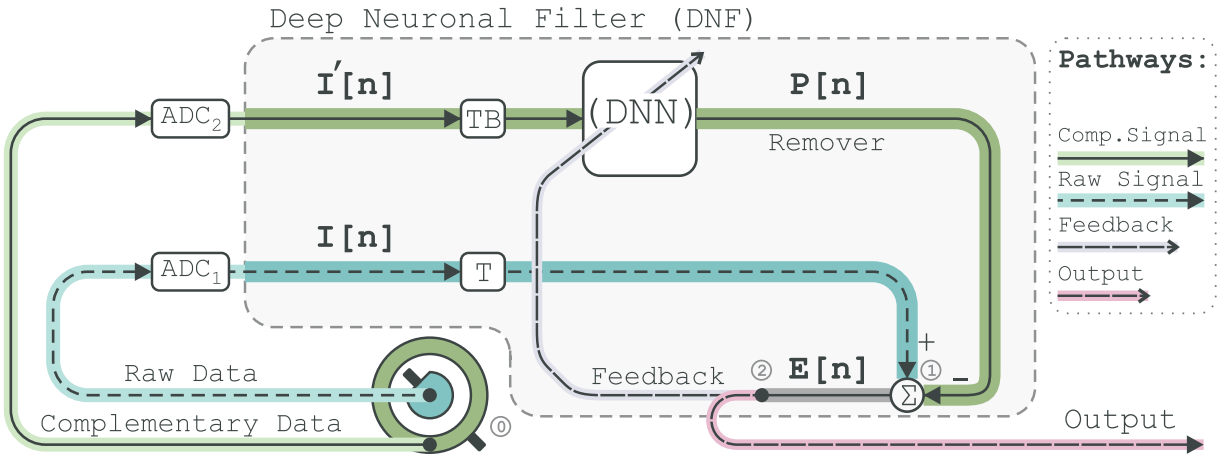

Figure 7.1: *Open-Loop Deep Learning Platform for Signal Processing. This illustration highlights the removal of the transfer functions of the environment, rendering the platform open at node ⓪, where the signals originate. The raw data follows the blue pathway, while the complementary data travels through the green pathway, housing the deep learner, represented by a DNN. At node ①, the DNN's output is subtracted from the raw data, resulting in the error signal used for DNN training and serving as the output of the DNF filter, as shown at node ②.*

The dashed grey shape marks the agent boundary; however, due to the absence of an environment, the term "agent" is now less fitting. Instead, this is now denoted as the deep neuronal filter (DNF), serving as the unit responsible for signal processing.

Highlighted in blue within Figure 7.1, the previous inner reflex loop has transformed into an open-loop pathway that carries raw data. The deep learner's task is to modify this data to align with user preferences. To achieve this, the network employs an additional input termed the complementary data. This traverses the open-loop pathway highlighted in green, previously recognised as the outer predictive learning loop.

The raw and complementary datasets originate from a physical entity like the electrode at node ⓪, these signals are fed into an ADC, ADC_1 and ADC_2, and the outcomes are labelled as $I[n]$ and $I'[n]$ respectively.

The network feeds the complementary data into a delay time buffer depicted as TB. Generating the output $P[n]$, referred to as the "remover", which aims to remove undesired components from the raw data at node ①. After a suitable delay introduced by T, the raw data reaches this node synchronously with the remover. The outcome of the subtraction

process at ① yields the open-loop desired signal, also known as the error signal $E[n]$ which is fed back into the learner for training and tuning.

7.3.1 A Paradox: Desired Output & Error Signal

In the above section, the data flow of both the raw and complementary pathways resulted in the generation of the intended signal, or the error signal $E[n]$. This signal holds significant importance within this framework, serving as both the feedback employed to train and fine-tune the deep learner, and also as the desired output of the DNF which is returned to the user. This presents a paradox, as it is conventionally anticipated that the error signal should approach zero following successful learning. However, this signal also functions as the output of the DNF which is expected to be a non-zero signal which carries the desired attributes of the raw data.

This seemingly paradoxical statement can be resolved by revisiting Section 2.4.3.3, which underscores that learning is not solely propelled by the error, but by its *correlation* with the inputs. Based on this premise, this signal is explored in-depth in the following section.

7.3.2 Superimposition of Signal & Noise

As mentioned earlier, the goal of the DNF is to eliminate unwanted components of the raw data. This section delves deeper into how this elimination takes place and derives an expression for the learning rule in the context of open-loop signal processing. All data sets are expressed as a superimposition of signal and noise. Signal $S[n]$ represents the desired portion of the data, while noise $N[n]$ represents the unwanted portion that the deep learner aims to remove. With this, the raw data can be expressed as:

$$I[n] = C_S S[n] + C_N N[n] \tag{7.1}$$

where C_S and C_N are the coefficients that describe the amount of the signal and noise in the raw data, respectively. Similarly, the complementary data is also a superimposition of the signal and the noise:

$$I'[n] = C'_S S[n] + C'_N N[n] \tag{7.2}$$

where C'_S and C'_N are the coefficients that describe the amount of the signal and noise in the complementary signal, respectively. Note that any of these coefficients could be

zero. Equation 7.3 below represents an alternative representation of the complementary data:

$$I'[n] = h[n] * (\alpha \cdot S[n] + N[n]) \tag{7.3}$$

where $h[n]$ is an arbitrary filter, and $0 < \alpha \ll 1$ models the crosstalk between the root $I[n]$ and ring $I'[n]$ electrode signals, as the signal of the root electrode will also stray into the outer ring. Therefore, the output of the network, which is a function of the complementary data and the internal parameters of the network, can be expressed as:

$$\begin{aligned} P[n] &= \mathbb{N}(I'[n], [\omega]) && (7.4)\\ &= \mathbb{N}(C'_S S[n], [\omega]) + \mathbb{N}(C'_N N[n], [\omega]) && (7.5)\\ &= C_{PS} S[n] + C_{PN} N[n] && (7.6) \end{aligned}$$

Here, $\mathbb{N}$ represents the neural network as a function and $[\omega]$ is the weight matrix of the network. The signals are separated under the assumption that the network operates in the linear regime of the activation function. Thus, the network's output can be expressed as a superimposition of signal and noise with coefficients C_{PS} and C_{PN}, respectively. Finally, the error signal can now be calculated as:

$$\begin{aligned} E[n] &= P[n] - I[n] && (7.7)\\ &= (C_{PS} S[n] + C_{PN} N[n]) - (C_S S[n] + C_N N[n]) && (7.8)\\ &= (C_{PS} S[n] - C_S S[n]) + (C_{PN} N[n] - C_N N[n]) && (7.9)\\ &= (C_{PS} - C_S) S[n] + (C_{PN} - C_N) N[n] && (7.10)\\ &= C_{ES} S[n] + C_{EN} N[n] && (7.11) \end{aligned}$$

This demonstrates that the error signal is also a superimposition of the original signal and noise. All four of the aforementioned data sets contain components of the signal and noise. This is a crucial concept for understanding the functionality of the DNF.

7.3.3 Learning Objective

The aim is to remove the noise component of the raw data. In mathematical terms, this is represented as below:

$$\text{Learning goal:} \quad \begin{cases} \text{Noise is removed} & C_{PN} = C_N \\ \text{The signal remains} & C_{PS} \neq C_S \end{cases} \quad \therefore E[n] = C_{ES} S[n] \tag{7.12}$$

This illustrates the ideal error signal. During the initial stages of learning, the error is a blend of noise and signal, as evident from Equation 7.7. Once learning is accomplished, however, the resultant error comprises solely the signal. While this outcome aligns with the desired output of the DNF, the question arises: how can a non-zero error signify the successful convergence of the network and yet induce no further adjustments? This query is tackled in the subsequent section by examining the learning rule.

7.3.4 Learning Rule

The learning rule in this context is governed by the open-loop update rule that was derived in Section 2.3. However, unlike typical applications of deep learning, the precise desired output of the network remains unknown here, as observed in the aforementioned closed-loop paradigms as well. Even though the desired output is not ascertainable, a closer examination of the learning rule reveals how the learning objective can still be accomplished with specific signal prerequisites.

Referring to Equation 2.18, the internal error that propels the alteration of weights within each neuron is contingent on the feedback error in this scenario. This function is essentially the *inverse* of the network's function from the last layer to the relevant layer ℓ. Meanwhile, the input to each neuron is determined by the complementary data fed into the network; this function corresponds to the network's *forward* function from the initial layer to the relevant layer ℓ. Consequently, the counterpart of weight change for the DNF can be formulated as:

$$\Delta\omega_{ij}^{\ell} = \eta \mathbb{N}^{-1}(E[n], [\omega]_{\ell \leftarrow L}^{T}) \cdot \mathbb{N}(I'[n], [\omega]_{0 \rightarrow \ell}) \tag{7.13}$$

$$\overset{\text{DNF}}{\equiv} \eta E[n] \cdot I'[n] \tag{7.14}$$

Hence, it can be demonstrated that the alteration in weight is indirectly influenced by the correlation between the error signal and the complementary data. These elements, as previously demonstrated, consist of both signal and noise. Upon substituting these elements, we obtain:

$$\underset{dnf}{\Delta\omega_{ij}^{\ell}} \equiv \eta(C_{ES}S[n] + C_{EN}N[n]) \cdot (C'_{S}S[n] + C'_{N}N[n]) \tag{7.15}$$

$$\equiv \eta(C_{ES}C'_{S})S[n] \cdot S[n] + (C_{ES}C'_{N} + \eta C'_{S}C_{EN})S[n] \cdot N[n] + \eta(C_{EN}C'_{N})N[n] \cdot N[n] \tag{7.16}$$

The correlation of similar waves leads to a substantial peak ($S[n] \cdot S[n]$ and $N[n] \cdot N[n]$), whereas the correlation of opposing waves results in a negligible value ($S[n] \cdot N[n]$). Therefore, the weight change can be approximated and summarised as:

$$\underset{\text{DNF}}{\Delta\omega_{ij}^{\ell}} \equiv \eta(C_{ES}C'_{S})S[n] \cdot S[n] + \eta(C_{EN}C'_{N})N[n] \cdot N[n] \tag{7.17}$$

Here, the noise and signal constituents of both the error and complementary data correlate to drive the weight change. This bears resemblance to the FIR match filter as the noise and signal components of the same data are being correlated here. However, the distinction is in the application and properties of the filter. Matched filters are used to detect repeated patterns in a signal, whilst the DNF is designed for noise removal. In matched filters the pattern is time-reversed and used as filter coefficients, whereas neither one of the signals from the two electrodes is time-reversed here.

7.3.4.1 Effective Learning Rate

It is crucial to emphasise that the effective learning rate η_{e} is directly proportional to the amplitude of the noise reference $x[n]$, as expressed in the equation:

$$\eta a_{i}^{\ell-1} \cdot \delta_{j}^{\ell} \equiv \underbrace{\eta I'[n]}_{\eta_{\text{e}}} \cdot E[n] \tag{7.18}$$

To maintain a constant effective learning rate, one can either normalise the noise reference $x[n]$ or dynamically adjust the learning rate when the average amplitude of $x[n]$ fluctuates. In this study, we opted to directly set the learning rates to accommodate the two different noise reference amplitudes of $x[n]$ for the P300 task ($\eta = 10$) and the jaw muscle task ($\eta = 2.5$). This adjustment ensured that the effective learning rates were consistent between the two tasks.

The equation above also illustrates that learning reaches convergence when the correlation between the noise reference $x[n]$ and

the error signal $e[n]$ diminishes. This signifies that no frequency components of the

noise present in the outer electrode signal persist in the output of the DNF filter, indicating successful noise removal.

7.3.5 Resolving the Paradox

Referring to Equation 7.17, it can be deduced that the noise components of the error and complementary data correlate up to the point where this element is eliminated from the error data (also being the desired output). Similarly, the signal component of both can correlate until it is eliminated from the desired output (also being the error).

Consequently, it is necessary for the complementary data to exclusively carry attributes of the noise, with $C'_N \neq 0$, while retaining none of the characteristics of the signal, $C'_S \approx 0$. This configuration would ensure that the noise components correlate, prompting a weight change that subsequently eradicates noise from the error (and the output). Simultaneously, this setup would prevent the signal component of the error from correlating with any aspects of the complementary data, thus averting its removal. Note that if the noise components do not correlate in time, they can pass through the filter, however, the large enough delay line can help capture noise components that might be delayed in time. Moreover, in the context of this application, the electrodes are centimetres apart and therefore it is unlikely for their noise component to experience a significant delay in time.

In effect, the earlier contradiction is resolved by recognising that as long as the complementary data avoids containing signal attributes, the error feedback, comprising solely signal traits, would yield a zero correlation, thereby halting further adjustments to the network or the output. Thus, the error signal can exclusively encompass the signal and also function as the desired output. This concludes the derivation of the DNF learning framework. The following section outlines the application of this learning paradigm to EEG signal processing.

7.4 Experimental Setup

The derivations discussed form part of this study. However, the experimental setup, electrode fabrication, and execution of the experiments were all conducted by Henry Cowan and Lucia Munoz Bohollo. These elements are briefly described here to ensure clarity and comprehensiveness for the reader.

7.4.1 Smart Deep Electrode

As previously demonstrated, the electrodes must be designed such that one channel captures the raw data containing both signal and noise. Conversely, the collection of complementary data requires a setup that primarily captures noise, minimising the signal

component. This rationale led to the development of the smart deep electrode tailored for EEG signals.

7.4.1.1 Architecture & Working Principle

Brain signals are generally believed to originate *"locally"*, from a small surface area of the scalp. In contrast, artefactual noise is thought to stem from more distant sources, resulting in a *"global"* uniform strength across the scalp. Consequently, the smart electrode is designed with two isolated electrodes.

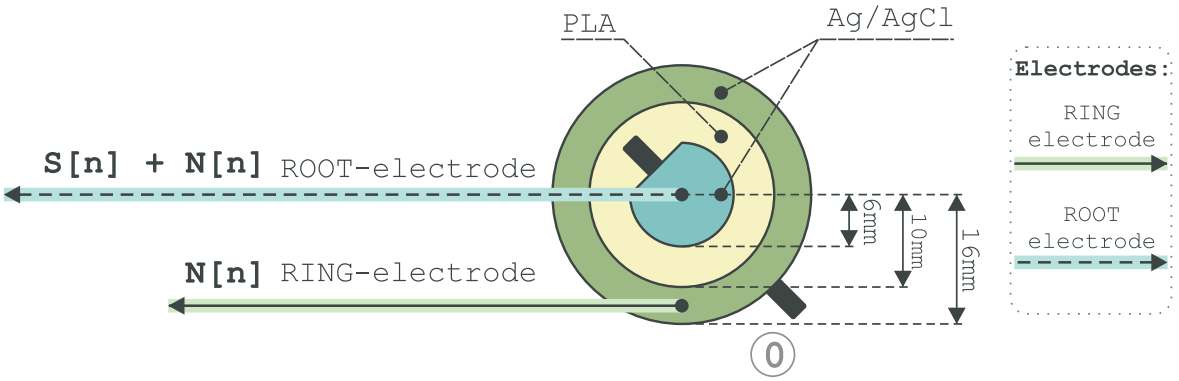

Figure 7.2: *Schematic Diagram of the Compound Electrode: Depicts the root and ring electrodes in blue and green, respectively, along with their respective material composition, and the dimensions of each electrode are indicated.*

Figure 7.2 presents the architectural layout of this compound electrode. It comprises an inner section referred to as the root electrode and an encompassing outer section known as the ring electrode. Collectively, they constitute the smart deep electrode. The root electrode acquires both the local EEG signal of interest and the inevitable global noise ($S[n] + N[n]$). On the other hand, the ring electrode is strategically designed to solely capture the global noise ($N[n]$), ideally excluding any portion of the EEG signal of interest. This electrode design aligns with the signal requisites discussed in Section 7.3.5 and is compatible with the open-loop learning platform.

7.4.1.2 Fabrication

The electrode was produced through 3D printing, chosen due to the intricate nature of the design and the proven success of this manufacturing technique (Velcescu et al., 2019; Rohaizad et al., 2019; Salvo et al., 2012). The team's proficiency with this approach (Ntagios et al., 2019) further supported this decision. The final product was crafted to be flexible, durable, and biocompatible.

Poly-lactic acid (PLA) was selected due to its ability to ensure proper adhesion to the backing material, which is crucial for the electrode's conductivity (Chen et al., 2014).

Subsequently, a layer of Ag/AgCl was applied to the electrodes, as this combination has proven effective in comparable applications (Rohaizad et al., 2019). The wires of the electrode were then affixed using pure silver paste and epoxy. Notably, this electrode design holds potential for scalability into a wearable cap for EEG data acquisition.

7.4.1.3 Electrode Positioning & Device

Data acquisition was performed using a two-channel DC-coupled bio-amplifier called Attys (Glasgow Neuro LTD) and software programs attys-ep and attys-scope. The data was digitised at the electrode measurement points without any analogue filtering (Porr et al., 2022).

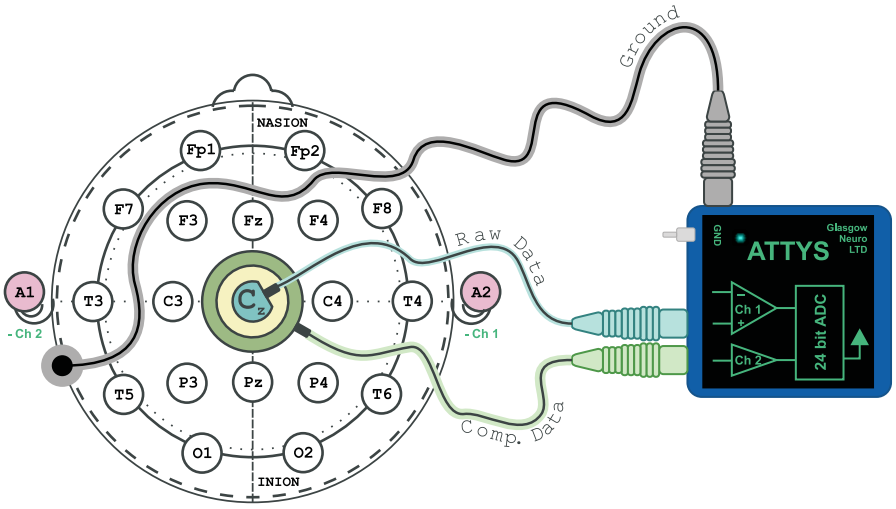

Figure 7.3: *Electrode Placement According to the International 10-20 System: It shows that the compound electrode was positioned at the subject's C_z location on the head. The root electrode was linked to the positive input of Channel 1, while the ring electrode was connected to the positive input of Channel 2 of the Attys. The A2 electrode was attached to the negative input of Channel 1, while the A1 electrode was connected to the negative input of Channel 2, serving as the ground reference.*

The compound electrode was placed on the subject's head at the C_z position of the international $10-20$ system. This electrode's inner part was connected to Channel 1, and the outer ring to Channel 2 of the Attys. Standard adhesive electrodes A2 and A1 were placed behind the right and left ears, respectively, connecting to the negative inputs of Channels 1 and 2. Channel 2 also served as the ground. This configuration is schematically depicted in Figure 7.3. Each subject participated in two sessions without intervals to ensure consistent electrode signals.

7.4.2 Data Acquisition

7.4.2.1 Session 1: EEG with EMG Noise Generation

The aim of the first session was to create an EEG signal contaminated with EMG noise. Participants were instructed to contract their jaw muscles every 15 seconds for two minutes, inducing EMG noise. The sampling rate was set at 500 Hz to ensure a uniform response in the EEG frequency band ranging from 0 to 100 Hz. The smooth roll-off of the sigma-delta converter towards the Nyquist frequency of 250 Hz was a key factor in this setting.

7.4.2.2 Session 2: Acquiring Noise-Free EEG Signal Power

The focus of the second session was to capture the power of a noise-free EEG signal. As complete muscle inactivity, which would eliminate EMG noise, was not feasible, evoked potentials were used to minimise EMG noise. P300 visually induced oddball stimuli were employed for this purpose. Participants were exposed to a black and white chequerboard pattern that inverted every second, with brightly coloured horizontal bars (oddball stimuli) appearing randomly between every 7 to 13 seconds. Participants were tasked with silently counting these oddball stimuli. The sampling rate for this session was 250 Hz, appropriate for measuring the peak power of evoked potentials, which primarily have low-frequency components.

7.4.2.3 P300 Signal

The signal obtained from the inner electrode is a composite of baseline EEG, EMG, and the deliberately created EEG signal $c[n]$. To realistically estimate $c[n]$, the power of the primary peak of the P300 evoked potential is used. This is based on the median power observed between 300 ms and 500 ms, accommodating the 100 ms latency in wireless transmission between the ADC and the P300 software. The P300 is conceptualised as a pulse at 300 ms, detectable by systems like a P300 speller. The use of a median over this interval slightly underestimates the power, which aligns with the reality of real-time brain computer interface (BCI) systems that typically average over shorter durations, thus dealing with lower signal strengths for $c[n]$. Using the median filter, therefore, corrects for potentially overly optimistic signal strength estimations.

7.4.2.4 Muscle Noise

The noise analysis primarily targets the power of EMG noise generated by facial and jaw muscles, excluding low-frequency components like EOG or electrode drift. The periodogram using the Welch method is employed for this analysis. The method's window length equals the sampling rate, providing power density in 1 Hz bins. The power den-

sity samples from 5 Hz to 125 Hz are summed to determine the total noise power in this frequency range.

7.4.3 SNR Calculation

The SNR is calculated using the following formula:

$$\mathrm{SNR} = \frac{\mathrm{median}(v^2_{\mathrm{P300},\pm 100\ \mathrm{ms}})}{\sum_{k=5\ \mathrm{Hz}}^{125\ \mathrm{Hz}} \mathrm{Welch}(v)[k]} \tag{7.19}$$

This equation represents how the SNR is determined in this application. The numerator involves the median of the squared values of the P300 signal within a ±100 ms window, reflecting the power of the signal component, particularly the P300 evoked potential. The denominator sums the power density (calculated using the Welch method) of the noise component in the frequency range of 5 Hz to 125 Hz, capturing the EMG and other noise present in the signal. The formula is applied to various signals to evaluate their SNR. These include: a) The root electrode signal $d[n]$, b) The output $e[n]$ from DNF, c) The output from a standard LMS-based FIR filter, and d) The result of a Laplace operator, calculated by directly subtracting the raw outer electrode signal $\tilde{d}[n]$ from the inner one $\tilde{x}[n]$. The Laplace operator is described in Appendix B.

This SNR calculation method aims to assess the effectiveness of different signal processing techniques in enhancing the quality of the EEG signal by comparing their ability to suppress noise while preserving the signal of interest.

7.5 Experimental Setup & Network Architecture

The experimental setup and common aspects of the network are detailed in section 2.7. The network used for DNF is designed with $L = 6$ layers. The number of neurons $I(\ell)$ per layer index ℓ is calculated as:

$$b = e^{\frac{\ln N_{\mathrm{taps}_x}}{L-1}} \tag{7.20}$$

$$I(\ell) = \lfloor \frac{N_{\mathrm{taps}_x}}{b^{\ell-1}} \rfloor \quad \textit{where:} \quad \ell : 1, \ldots, L \tag{7.21}$$

which guarantees that the output layer consists of exactly one neuron which generates the "remover" signal. In our case with $N_{\mathrm{taps}_x} = 50$ inputs to the DNF this results in: $I = 50, 22, 10, 4, 2, 1$ neuron(s) per layer which means that the first layer is fully connected with the same number of neurons to the delay line and then the number of neurons are reduced in the form of a funnel as done in auto-encoders.

7.6 Results

The following findings in this section refer to result (g) in Figure 1.1 in the preface.

7.6.1 Complete Trial

Figure 7.4 shows the real-time learning progress of the DNF over a 2-minute period for one subject. In Panel A, the raw signal represents the $I[n]$ signal from the root electrode. Notably, the voluntary jaw muscle contractions occurring every 15 seconds are clearly identifiable and marked with an asterisk (*). In the intervals between muscle contractions, the signal is likely a mixture of baseline EEG and lower-amplitude involuntary facial EMG activity. In Panel B, the complementary trace displays the signal from the ring electrode $I'[n]$, where the EMG bursts resulting from the jaw muscles are clearly visible. Both the raw and complementary data are subsequently fed into the DNF. The most critical internal signal is the remover $P[n]$ in Panel C, which serves to eliminate noise (as described in Equation 7.1). The outcome of the subtraction, denoted as $E[n]$, is observable in Panel D.

7.6.2 Euclidean Weight Distance & Convergence

The DNN consists of 6 layers, and its weight evolution, as related to Figure 7.4, is depicted in Figure 7.5 over the two-minute duration. The plot illustrates the change in weights from their initial random values. Learning is most rapid during jaw muscle contractions due to the higher amplitude of the noise reference $I'[n]$, resulting in a higher effective learning rate, as defined in Equation 7.18. Learning continues at a slower rate between EMG bursts. Around the 60-second mark, learning stabilises, with only minor weight adjustments until the end of the experiment. The filter operates in a closed-loop fashion, leading to corrective actions as the weights decrease following a jaw contraction. This indicates that jaw muscle recruitment and involuntary muscle activity exhibit slightly different correlations, prompting the network to adapt accordingly.

7.6.3 Noise Removal Process

Figure 7.6 shows truncated versions of the signals in Figure 7.6, specifically between $89[s]$ and $90[s]$, capturing the onset of a jaw clench around $89.5[s]$. Panel A shows the raw signal $\bar{I}[n]$ originating from the root electrode. To illustrate the noise removal process, this signal is delayed by T steps according to Figure 7.1. Panel B shows the noise reference $I'[n]$ from the ring electrode, which is fed into the DNF and enters its tapped delay line TB. The DNF then generates the remover signal $P[n]$ shown in Panel C, which effectively cancels

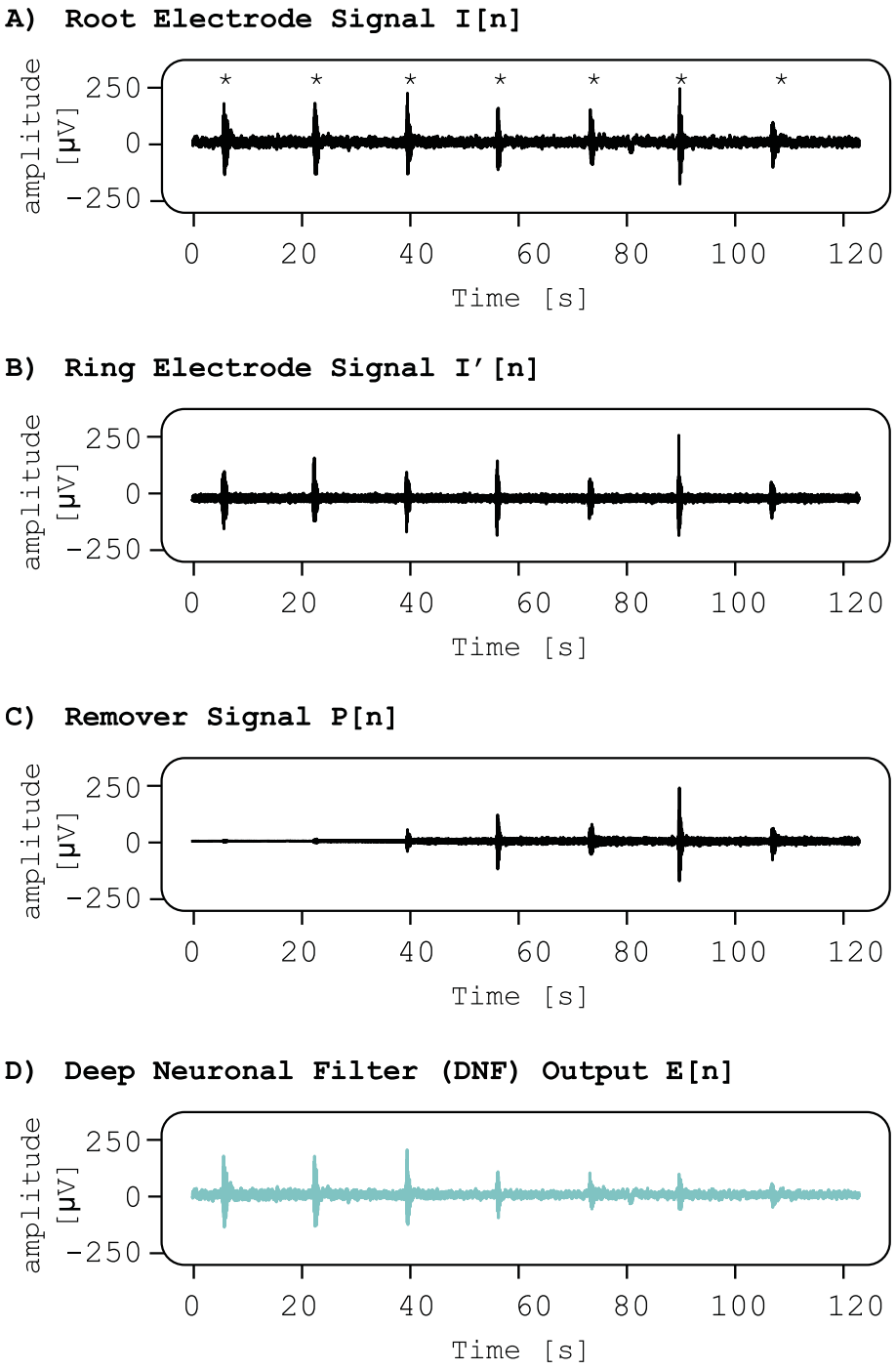

Figure 7.4: *Signal Traces:* A) *The raw signal* $I[n]$ *from the root electrode, carrying a mixture of EEG and EMG* B) *The complementary signal* $I'[n]$ *from the ring electrode, serving as the noise reference* C) *The output of the DNN known as the remover* $P[n]$ D) *And the DNF output* $E[n]$*, which serves as the output of the DNF and the error signal for training.*

out the noise in $\bar{I}[n]$. Panel D shows the output of the DNF $E[n]$, which also serves as the error signal used for training.

Recall that the DNF eliminates anything present in both the contaminated signal $I[n]$

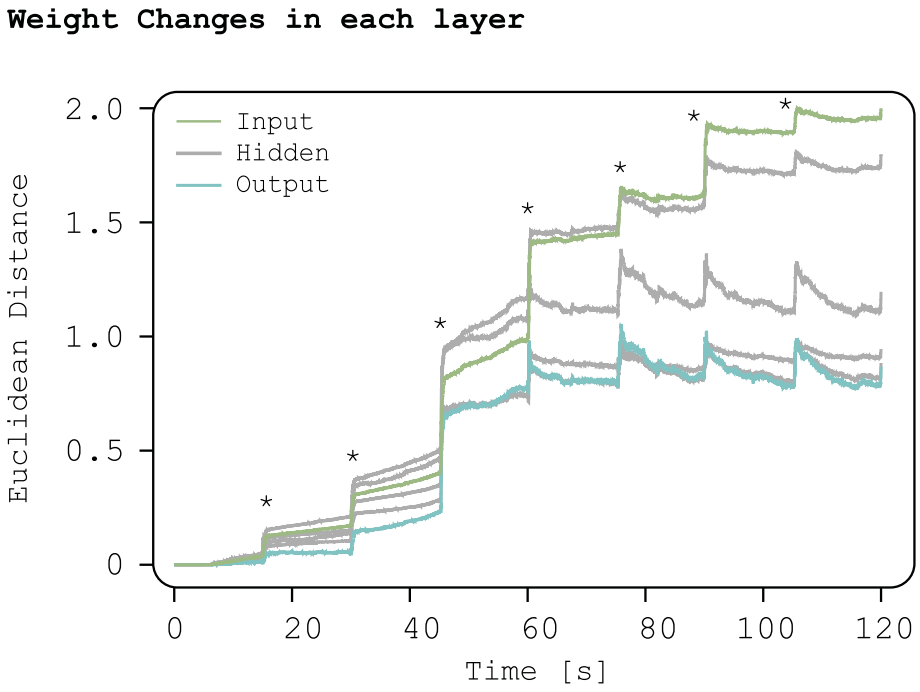

Figure 7.5: *Weight Evolution Over a 2-Minute Learning Trial with DNF: Illustrates the Euclidean distance of weights within the DNN, showcasing the input layer in green, the output layer in blue, and the hidden layers in grey.*

and the noise reference $I'[n]$. This is clearly evident in the presence of the large peak in both the contaminated signal and the noise reference. Consequently, the DNF learns to eliminate this peak while preserving the remainder of the signal. It is worth noting that $E[n]$ is also the error signal, which is no longer correlated with the noise reference $I'[n]$. This de-correlation is reflected in the learning rule described in Equation 7.17, leading to the stabilisation of weights, as observed around the $90[s]$ mark in the learning process as seen in Figure 7.5.

7.6.4 P300 Averages

To compute the SNR, the power of the signal and the power of the noise must be computed separately, as seen in Equation 7.19. Initially, we focus on evaluating the signal power. This entails estimating the power of the primary P300 peak observed during experimental session 7.4.2.2. Importantly, there is no need to pass the EEG containing the P300 through the DNF since event-related averaging effectively eliminates the EMG noise. However, for verification, the P300 peaks were examined before and after noise reduction, as demonstrated in Figure 7.7.

The P300 inherently has a low frequency, and with the DNF and least mean-squares (LMS) filter eliminating higher EMG frequencies, one would expect minimal alteration in the P300 shape. This expectation is validated by comparing the unfiltered P300 in

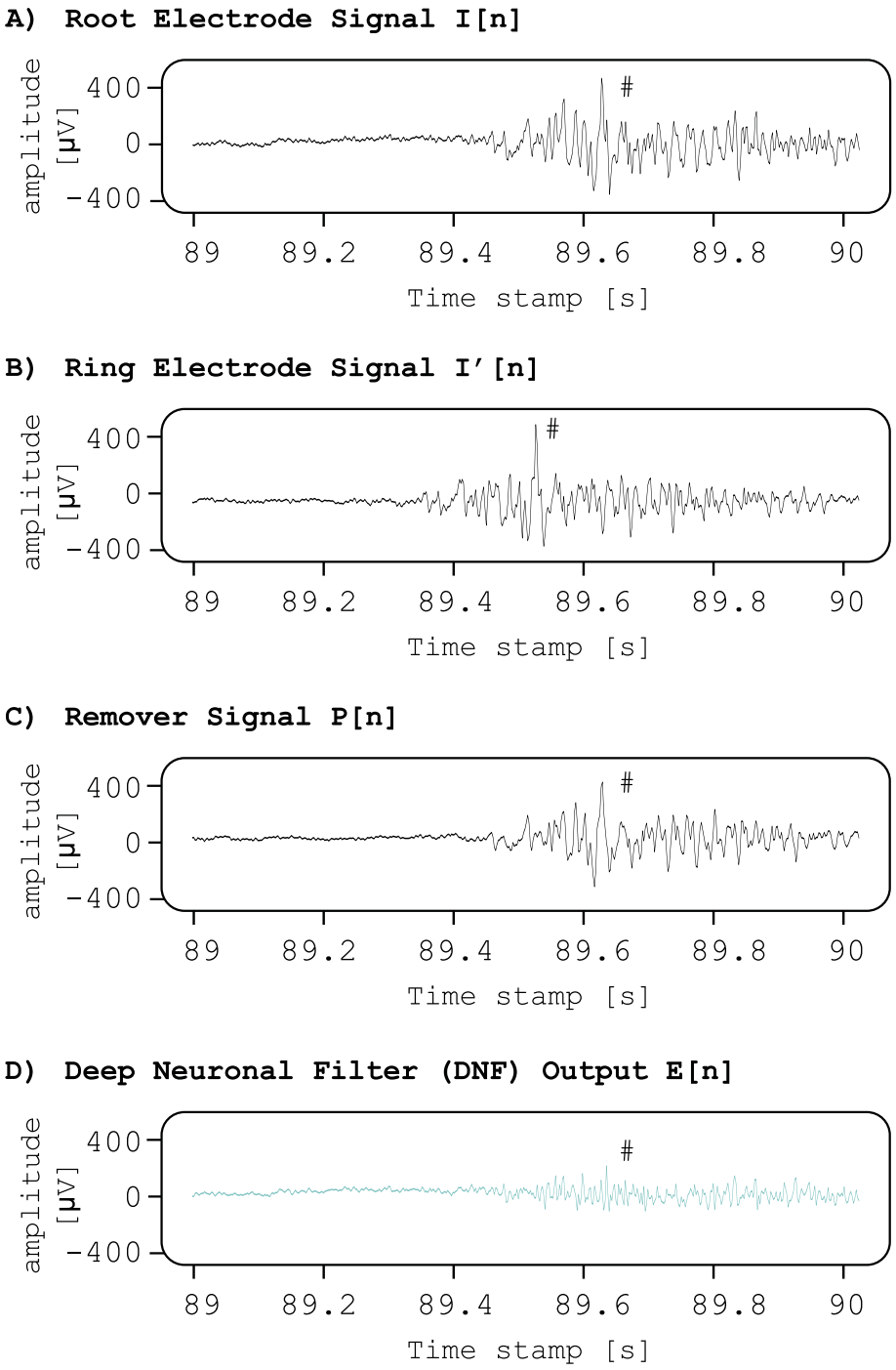

Figure 7.6: *Truncated Signal Traces focusing on one EMG activity:* A) *The delayed raw signal* $\bar{I}[n]$ *from the root electrode, carrying a mixture of EEG and EMG* B) *The complementary signal* $I'[n]$ *from the ring electrode, serving as the noise reference* C) *The output of the DNN known as the remover* $P[n]$ D) *And the DNF output* $E[n]$*, which serves as the final output of the DNF filter and the error signal for training.*

Figure 7.7 Panel A with the filtered P300 in Panel B and C. In all cases, the original EEG, the DNF output, and the LMS filter exhibit distinct peaks, and the square of these peak values represents the signal power. A comparison of the P300 from the original electrode

signal in Panel A with that of the DNF output in Panel B reveals a reduction in the DNF's P300 peak by approximately a quarter

(from 10 μV to 7.5 μV), whereas the LMS filter has minimal impact. This implies that the DNF filter must further reduce noise compared to the LMS filter to achieve an overall SNR improvement, even though it diminishes the P300 peak. LMS filters are described in Appendix C.

This outcome is expected because there is a crosstalk between the root and the ring electrodes as described in Equation 7.3, where EEG from the root electrode is partially present at the ring electrode. Since the DNF removes anything present in both the noise reference $I'[n]$ and its input signal $I[n]$, it treats the $\alpha > 0$ crosstalk of the EEG signal at the outer electrode as noise, reducing the amplitude of the noise-free EEG at its output.

Lastly, in Panel D, it is evident that the Laplace operator entirely eliminates the P300 peak, making it impossible to calculate SNR solely based on the Laplace operator. With the signal power for SNR calculated, we can proceed to assess the noise power.

7.6.5 Noise Power Density

Figure 7.8 illustrates the power spectral density of the root signal $d[n]$ and the outcomes from both the DNF and a standard LMS-based adaptive FIR filter. Remarkably, the DNF filter consistently reduces the noise to approximately $0.1 \cdot 10^{-11}[V^2/Hz]$ for frequencies over 10 Hz. In contrast, the noise from the inner electrode $d[n]$ exhibits substantial fluctuations, ranging from 0.2 to $0.8 \cdot 10^{-11}[V^2/Hz]$. Although the LMS-tuned FIR filter, a linear filter with a single layer, does achieve noise reduction, it underperforms by merely diminishing spectral components proportionally and fails to completely eliminate noise peaks, such as those at $35[Hz]$, $40[Hz]$, and $45[Hz]$.

7.6.6 Statistics & Reproducibility of SNR Improvements

Figure 7.9 A and B displays the individual SNR variations for different subjects using DNF and LMS filters respectively. In each Panel, the SNR calculations for the raw data $I[n]$ are presented on the left-hand side. It is clear that subjects with the poorest SNR at $-20dB$, particularly those with strong EMG bursts from jaw muscles, see the most significant improvement, as depicted in this figure. However, some subjects experienced only marginal enhancement, possibly due to poor electrode contact, resulting in minimal correlation between the inner and outer electrodes.

To verify the statistical significance of the noise reduction, we calculated the SNR for each subject pre- and post-filtering (expressed in dB) to determine the SNR enhancement:

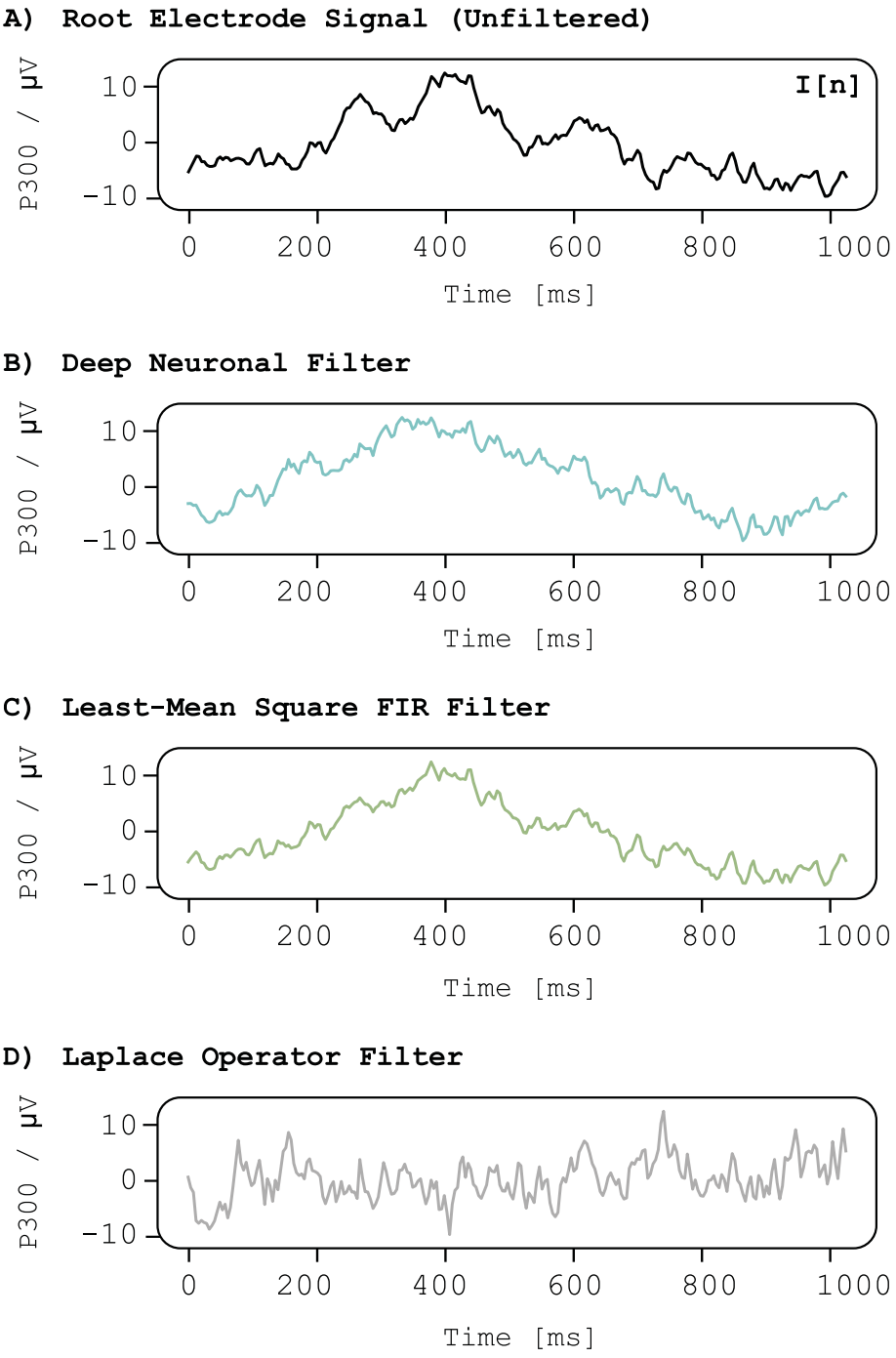

Figure 7.7: *P300 Averages for one subject. While observing a chequerboard that alternated every second, the subject was presented with sporadic stimuli every 7 seconds to 13 seconds in a randomised pattern. The recording duration was 5 minutes.* A) *Event-triggered average derived from the root electrode* $I[n]$. B) *Event-triggered average derived from the output* $E[n]$ *of the DNF.* C) *Output obtained from the LMS filter (adaptive FIR filter).* D) *Output obtained from the Laplace filter:* $\tilde{I}[n] - \tilde{I}'[n]$*, with DC and 50 Hz components removed following the subtraction operation.*

$$\Delta\text{SNR} = \text{SNR}_{\text{Root}} - \text{SNR}_{\text{DNF or LMS}} \tag{7.22}$$

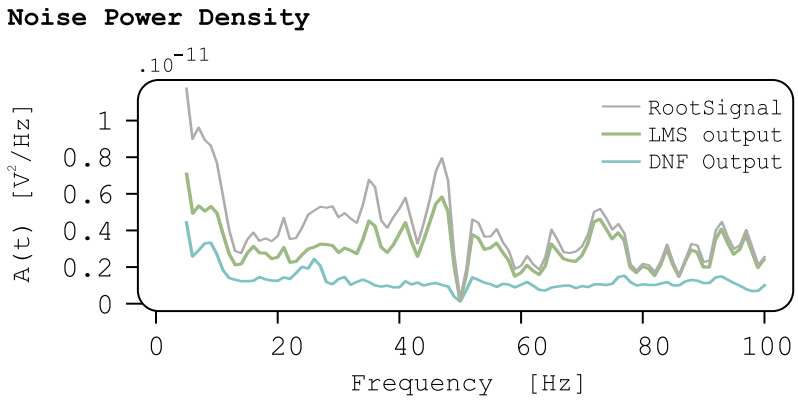

Figure 7.8: *Noise power density: This is calculated in 1 Hz bins for the root electrode $I[n]$, the DNF output $E[n]$, and the output of the conventional LMS-based adaptive FIR filter.*

Figure 7.9 C exhibits the SNR improvements for both DNF and LMS filters. Our novel DNF ($p = 0.000013$) and an LMS-tuned adaptive FIR filter ($p = 0.000192$) both significantly enhanced the SNR. However, the DNF outperforms the LMS filter ($p = 0.000026$) in terms of effectiveness.

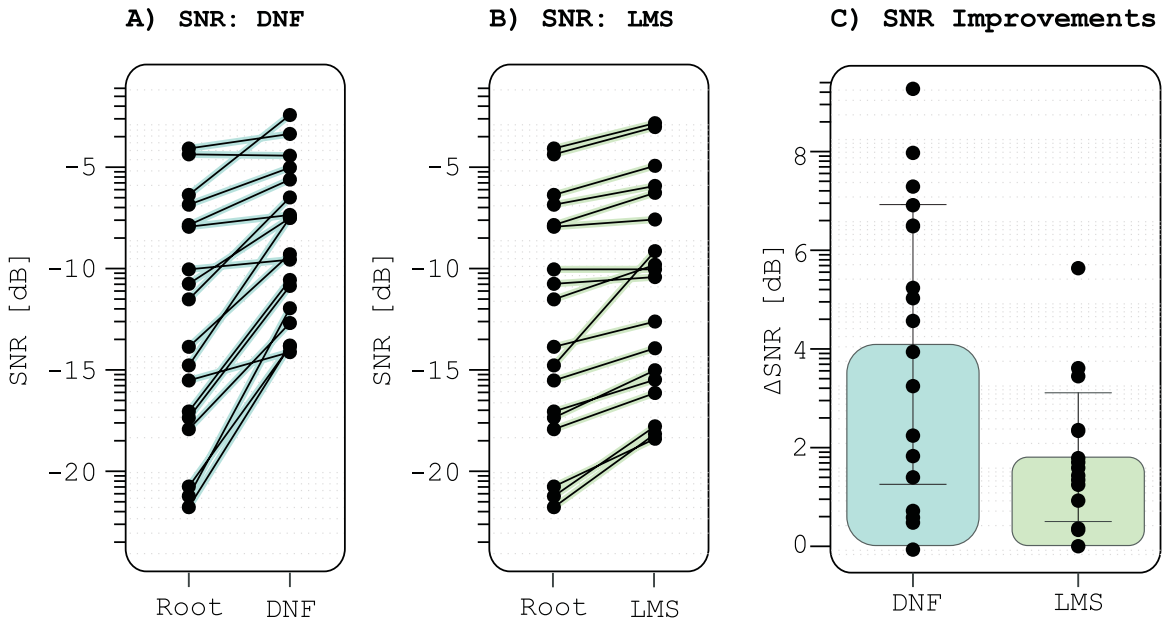

Figure 7.9: *SNR calculations using Equation 7.19:* A) *SNR in dB measured at both the root electrode $I[n]$ and the DNF output $E[n]$ for each subject.* B) *SNR in dB for the conventional LMS-based adaptive FIR filter, analysed across all subjects.* C) *SNR improvements for both DNF ($\Delta SNR_{DNF} = 4.1 \pm 2.8$ dB) and LMS-based FIR filter ($\Delta SNR_{LMS} = 1.8 \pm 1.3$ dB).*

7.7 Discussion

The LMS technique requires additional electrodes around the eyes. Alternative strategies, such as neural networks, have been utilised to create a 'remover' signal to eliminate EEG artefacts (Islam et al., 2016). Our approach employs a standard deep net with a non-linear activation function, differing from methods using radial basis functions (Mateo et al., 2013) or functional-link neural networks (FLNNs) for non-linear decision boundaries (Jafarifarmand and Badamchizadeh, 2013). Some have integrated FLNN with adaptive neural fuzzy systems for enhanced efficiency (Hu et al., 2015). Our model is computationally advantageous due to its broad availability and compatibility with optimised hardware.

DL, traditionally used as a classifier, has been effectively applied in EEG artefact detection, achieving up to 90% accuracy (Craik et al., 2019; Webb et al., 2021; Bahador et al., 2020). It aids ICA-based algorithms in identifying EMG noise components (Makeig et al., 1995; Delorme et al., 2007; Lee et al., 2020). Research has explored various network structures, such as fully connected neural networks, convolutional networks, and the unique encoder/decoder architecture DeepSeparator, for direct EMG noise removal (Zhang et al., 2021; Yu et al., 2022). Only certain networks proved stable for EMG removal. In contrast, DNF model achieves real-time continuous training and filtering. These models were trained by focusing on the error between clean EEG and filter output rather than reference noise (Yang et al., 2018; Nguyen et al., 2012). Since obtaining clean EEG is challenging, these are often synthetically generated (Zhang et al., 2021). Most EEG noise reduction studies rely on synthetic data and lack detailed analysis (Urigüen and Garcia-Zapirain, 2015).

Traditional mechanical EEG electrodes, unchanged since the first EEG (Umlauf, 1948), predominantly use Ag/AgCl cups (McAdams, 2006). Electrode-skin resistance impacts EEG's SNR (Schwab and Chock, 1953). New BCI and consumer EEG often use dry electrodes (Guger et al., 2012), spurring innovations in electrode design, such as spring contact probes, to reduce resistance (Krachunov and Casson, 2016; Velcescu et al., 2019; Nathan and Jafari, 2015; Liao et al., 2011). However, these designs focus on skin contact, not the spatial signal-noise distribution, necessitating compound electrodes.

BCI advancements (McFarland et al., 1997) show that electrode spatial distribution enhances SNR, as evidenced in ECG and EEG studies (Besio et al., 2006; Garcia-Casado et al., 2019; Aghaei-Lasboo et al., 2020). Optimal ring spacing improves the efficiency of the Laplace operator (Besio et al., 2006; Makeyev et al., 2016; Makeyev, 2018). However, real-world electrode impedance variability limits the effectiveness of analogue spatial averaging. Software-based Laplacian approximation (Fitzgibbon et al., 2015) offers a solution but faces computational costs and spatial resolution limits. Concentric ring electrodes are practical (Besio et al., 2006) yet assume ideal recording conditions. Our adaptive algorithm addresses these imperfections, particularly with dry electrodes, focusing on relevant noise (e.g., EMG) through high-pass filtering.

Chapter 8

Conclusion

This work has successfully achieved its primary objective of developing a learning algorithm capable of augmenting reactive agents with proactive capabilities. This objective was realised through a carefully structured approach, consisting of two primary tasks: the development of an integrated platform and the construction of the learning unit.

The integrated platform was designed to seamlessly integrate reflex mechanisms with learning processes, enhancing collaborative operations and signal exchanges between these components. This platform was engineered with adaptability and flexibility in mind, allowing it to be compatible with a wide range of learning algorithms, thus making it highly versatile and domain-agnostic.

The learning unit, a critical component embedded within the platform, was meticulously designed with a robust mathematical and technical framework. This framework optimally positioned the learning unit within the system, ensuring effective coordination with other platform elements. The deliberate separation between the platform and the learning unit underscored the adaptability and flexibility of the final product, offering a standardised approach to enhancing various applications with advanced learning functionalities.

Furthermore, the applicability of the developed algorithm was demonstrated through its implementation on a line-following robot, showcasing its ability to learn and navigate a path without relying solely on reflex actions. Additionally, the algorithm was transformed into a stand-alone C++ logic library, along with deployment packages for both simulation and physical robot scenarios, illustrating its broader utility for predictive learning in various contexts. This is left behind as a legacy for future researchers and developers to build upon.

Throughout the course of this research, several algorithms were developed, building upon each other's foundations and contributing to the field of closed-loop control and learning. These algorithms were: CLDL, SaR, PaM, Echo, and DNF. Each chapter of this thesis represents one of these algorithms, which made up a substantial portion of the

work. The many result sections presented in each chapter focused on comparative results and analyses. These algorithms are summarised and concluded below.

8.1 Comparative Summary of Developed Algorithms

8.1.1 Closed-Loop Deep Learning (CLDL) Algorithm

8.1.1.1 Pros

- **Real-Time Adaptation:** Integrates deep learning with a closed-loop control system for real-time learning and prediction.
- **Continuous Processing:** Operates within a continuous loop, enhancing learning speed and effectiveness.
- **Predictive Modelling:** Builds forward models for anticipatory actions, improving decision-making and reflexive responses.

8.1.1.2 Cons

- **Generalisation Limitations:** Requires further work to generalise to different forward models.
- **Complexity in Integration:** Combining CLDL with traditional Q-learning and other methods may add complexity to implementation.

8.1.1.3 Applications

- **Control Systems:** Effective for applications requiring fast and continuous learning, such as robotics and automated systems.
- **Real-Time Adaptation:** Suitable for environments needing rapid adaptation, like a line-follower robot in both simulations and real-world scenarios.

8.1.2 Sign and Relevance (SaR)

8.1.2.1 Pros

- **Biologically Inspired:** SaR leverages insights from neuroscience, incorporating neuromodulators like dopamine and serotonin, which enhance learning efficiency and convergence.
- **Multi-Layered Architecture:** Supports complex and nuanced learning processes.

- **Closed-Loop Framework:** Mimics biological processes, enhancing the algorithm's adaptability and real-world application.
- **Error Signal Separation:** Improves the learning process and network convergence by separating error signal sign and magnitude.

8.1.2.2 Cons

- **Complexity:** The sophisticated, multi-layered structure and biologically inspired mechanisms may make it more challenging to implement and tune compared to simpler algorithms.
- **Resource Intensive:** Requires significant computational resources due to its complexity and multi-layered approach.

8.1.2.3 Applications

- **Mental Health Research:** Understanding mental illnesses and reward-based learning processes.
- **Machine Learning Enhancement:** Improving the efficiency and convergence of machine learning algorithms.

8.1.3 Prime and Modulate (PaM)

8.1.3.1 Pros

- **Mitigates EVGP:** Addresses error signal vanishing and exploding gradient problems, which are common in traditional neural networks.
- **Biologically Inspired:** Utilises neurophysiological processes, combining local error back-propagation with global modulation.
- **Dynamic Adaptation:** Incorporates environmental cues, allowing dynamic adjustment of learning rates.

8.1.3.2 Cons

- **Local Minima Susceptibility:** Rapid convergence can lead to the algorithm being trapped in local minima.
- **Weight Symmetry Reduction:** Reduces the need for weight symmetry but may require careful balancing of local and global learning signals.

8.1.3.3 Applications

- **Self-Driving Vehicles:** Enhancing decision-making and safety by dynamically adjusting learning rates based on traffic conditions.
- **General Deep Learning:** Improving stability and performance in deep learning models.

8.1.4 Echo

8.1.4.1 Pros

- **Bidirectional Error Propagation:** Enhances learning accuracy and prevents instability by oscillating errors through the network until minimised.
- **Vanishing Gradient Mitigation:** Maintains gradients within a manageable range, improving training reliability.
- **Scalability and Versatility:** Suitable for various neural network architectures and applications.

8.1.4.2 Cons

- **Computational Demand:** Bidirectional oscillation can be computationally intensive, potentially slowing down training times.
- **Implementation Complexity:** The iterative error minimisation process may be complex to implement and fine-tune.

8.1.4.3 Applications

- **Classification Tasks:** Reliable and accurate training for simple to complex classification tasks.
- **Reinforcement Learning:** Robust framework for reinforcement learning scenarios.

8.2 Summary of Features, Strengths, and Constraints

The table below presents a summary of the features, strengths and constrains of each of the algorithms developed as part of this work.

Algorithm	Features	Strengths	Constraints	Applications
SaR	Multi-layered architecture, neuromodulators, error signal separation	Biologically realistic, enhanced learning efficiency	Complexity, resource-intensive	Mental health research, machine learning enhancement
PaM	Local error back-propagation, global modulation, dynamic learning rate adjustment	Addresses EVGP, rapid convergence	Local minima susceptibility, balancing local and global signals	Self-driving vehicles, general deep learning
Echo	Bidirectional error propagation, iterative minimisation	Improved learning accuracy, gradient management	Computational demand, implementation complexity	Classification tasks, reinforcement learning
CLDL	Closed-loop control, real-time learning, predictive modelling	Real-time adaptation, continuous learning	Generalisation limitations, integration complexity	Control systems, real-time adaptation

Table 8.1: Comparison of Learning Algorithms

8.3 Future Work

The exploration of learning algorithms such as SaR, PaM, Echo, and CLDL has demonstrated the potential of biologically inspired mechanisms and advanced error handling techniques in improving learning efficiency and robustness. To build upon this foundation, future work can be directed towards the following areas:

8.3.1 Exploring Other Network Architectures

While the current study focused on feed-forward neural networks, future research should investigate the application of these algorithms to other architectures, such as convolutional

neural networks (CNNs), recurrent neural networks (RNNs), and transformers. These architectures are highly relevant for tasks involving image and sequence data, and understanding how SaR, PaM, Echo, and CLDL algorithms perform in these contexts could reveal new strengths and application areas.

8.3.2 Complex Robotic Applications

Extending the application of these learning algorithms to more sophisticated robotic systems represents a promising direction. For example, developing a robotic arm designed for stroke rehabilitation that mimics the movements of a healthy arm can provide significant therapeutic benefits. Implementing the CLDL algorithm in this scenario could facilitate real-time adaptation and precise control, while SaR and PaM could enhance the learning of nuanced motor patterns based on the patient's unique recovery progress.

8.3.3 Incorporating Discrete Decision-Making Capabilities

Integrating discrete decision-making processes within these learning frameworks could significantly broaden their utility. For instance, in the context of autonomous driving, an algorithm that combines continuous learning with discrete decisions, such as overtaking a car or navigating complex intersections, would enhance both safety and efficiency. This could involve the development of hybrid models that incorporate reinforcement learning for decision-making alongside the continuous learning capabilities of CLDL or the stability of the Echo algorithm.

8.3.4 Scalability and Efficiency Improvements

Addressing the computational demands and complexity of implementing these algorithms is crucial for their widespread adoption. Future research should focus on optimising these algorithms to run more efficiently on modern hardware, potentially through parallel processing techniques or leveraging specialised hardware such as GPUs and TPUs. Additionally, simplifying the implementation and tuning processes will make these algorithms more accessible to practitioners.

8.3.5 Cross-Disciplinary Applications

Exploring the application of these algorithms beyond traditional machine learning and robotics domains can uncover new opportunities. For instance, the SaR algorithm's incorporation of neuromodulators could be applied in cognitive neuroscience research to model and understand complex brain functions. Similarly, PaM's dynamic adaptation capabilities could be valuable in financial modelling and other dynamic, data-driven industries.

By pursuing these avenues, future research can build on the promising results of SaR, PaM, Echo, and CLDL algorithms, leading to more versatile, efficient, and reliable intelligent systems.

8.4 Final Words

In conclusion, this research project has not only achieved its primary objectives but has also contributed to the advancement of knowledge in the field of predictive learning for reactive agents. The algorithms developed within this work offer a valuable framework for enhancing the capabilities of a wide range of systems, paving the way for more adaptive and intelligent applications in various domains. The rigorous mathematical derivations and experimental validations conducted throughout this work ensure the effectiveness and reliability of the proposed approach, making it a valuable contribution to the fields of reinforcement learning, robotics, and control systems.

The research stands apart from conventional trends within the field due to its distinctive characteristics and innovative approach. Unlike many research endeavours that primarily focus on theoretical simulations or offline analysis, this work is firmly rooted in real-time applications involving physical platforms. This aspect introduces a level of complexity and practicality that is not commonly explored in the field. Moreover, the algorithms developed and the manner in which they are applied to enhance the capabilities of reactive agents are remarkably novel and unique. This research delves into uncharted territories by pushing the boundaries of what is achievable in the realm of autonomous decision-making in dynamic environments. Echo learning is a prime example of this. As a result, it not only contributes to the advancement of knowledge in this field but also opens up new horizons for the development of intelligent and adaptable systems that can operate effectively in real-world scenarios.

In the realm of future work, an exciting avenue of exploration could involve the development of algorithms for making hard decisions in dynamic environments. Such decisions might include determining whether to overtake an obstacle on the path, whether to stop abruptly, or to execute a complete turn-around manoeuvre. This endeavour would complement the soft decision-making algorithm presented in this work, as it would extend the scope of the learning capabilities to encompass complex and high-stakes scenarios.

Appendix A

Digital Signal Processing (DSP)

The entirety of this appendix is adopted from Bernd Porr's lecture notes for Digital Signal Processing course (Porr, 2020).

A.1 Fourier Transform

The Fourier transform, represented by the formula:

$$\mathcal{F}\{x(t)\} = X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}\, dt \tag{A.1}$$

provides a powerful method to analyse the frequency content of a signal. However, it is not always suitable for causal systems. One limitation of the Fourier transform is its reliance on the entire signal spanning from $-\infty < t < +\infty$. This becomes impractical for causal signals, which are typically defined only for $t \geq 0$ (Porr, 2020).

A.2 Laplace Transform

To the limitation of Fourier transform, the Laplace transform was introduced. It is designed to handle continuous, causal signals more effectively. The Laplace transform is given by:

$$\mathcal{L}\{h(t)\} = H(s) = \int_{0}^{+\infty} h(t)e^{-st}\, d \tag{A.2}$$

where s is a complex parameter $s = \sigma + j\omega$.

The Laplace transform, unlike the Fourier transform, not only analyses the frequency components but also provides a broader view by taking the region of convergence into account. This makes it more suited for systems that are both causal and described by differential equations (Porr, 2020).

A.2.1 Properties

The properties Laplace transform simplify the analysis of linear time-invariant (LTI) systems and the solution of differential equations. These properties are as below in detail:

A.2.1.1 Integration

Integration property shows that integration in the time domain corresponds to multiplication by $\frac{1}{s}$ in the s-domain:

$$\mathcal{L}\left\{\int_0^t f(\tau)d\tau\right\} = \frac{1}{s}F(s) \tag{A.3}$$

A.2.1.2 Differentiation

Differentiation property implies that differentiation in the time domain becomes multiplication by s in the s-domain:

$$\mathcal{L}\{f'(t)\} = sF(s) - f(0^-) \tag{A.4}$$

A.2.1.3 Time Shift

Time Shift property reveals that a time delay (or shift) in the time domain translates to a multiplication by an exponential term in the s-domain:

$$\mathcal{L}\{f(t-a)u(t-a)\} = e^{-as}F(s) \tag{A.5}$$

where $u(t)$ is the unit step function.

A.2.1.4 Convolution

Convolution property asserts that convolution in the time domain corresponds to multiplication in the s-domain. If $\mathcal{L}\{f_1(t)\} = F_1(s)$ and $\mathcal{L}\{f_2(t)\} = F_2(s)$ then:

$$\mathcal{L}\{f_1(t) * f_2(t)\} = F_1(s)F_2(s) \tag{A.6}$$

where $*$ denotes convolution.

A.3 Filters

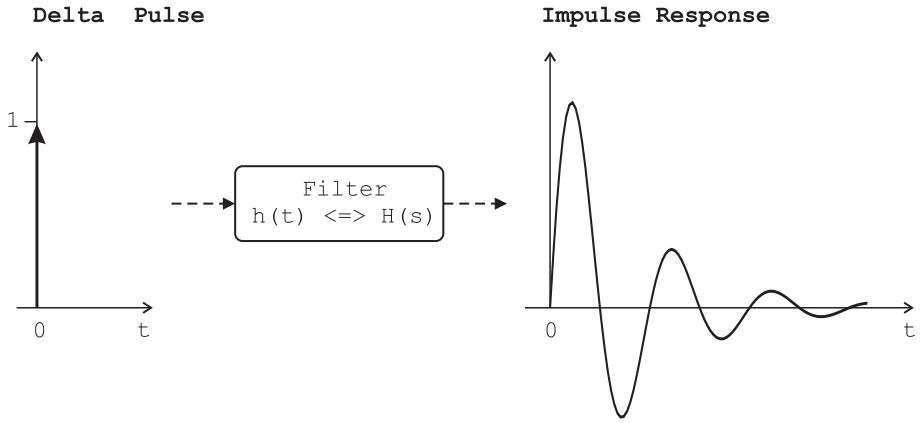

Figure A.1: *(redrawn from (Porr, 2020)) Shows the filter operation as a black box with a Delta pulse as in input and its time-domain impulse response.*

Consider Figure A.1 , which depicts a causal filter operating in continuous time. This filter is illustrated as a "black box" to emphasise that we are focusing on its overall behaviour rather than its internal structure or workings.

If we denote: $x(t)$ as the input signal, $h(t)$ as the system's impulse response, and $y(t)$ as the output signal,

then the relationship between the input and the output in the time domain, for a LTI system, is given by the convolution integral:

$$y(t) = x(t) * h(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)\ d\tau \tag{A.7}$$

In the Laplace domain, the convolution operation in the time domain becomes a multiplication. If: $X(s)$ is the Laplace transform of $x(t)$, $H(s)$ is the Laplace transform of $h(t)$, and $Y(s)$ is the Laplace transform of $y(t)$, then:

$$Y(s) = X(s) \cdot H(s) \tag{A.8}$$

Thus, the filter's operation can be succinctly described as a convolution in the time domain and a multiplication in the Laplace space (Porr, 2020).

A.3.1 Characterisation

There are two primary ways to characterise a filter:

A.3.1.1 Impulse Response

The impulse response of a filter is the output $y(t)$ when a delta pulse, $\delta(t)$, is applied as the input. Mathematically:

$$x(t) = \delta(t) \quad \text{(Delta Pulse)} \tag{A.9}$$

$$h(t) = y(t) \quad \text{(Impulse Response)} \tag{A.10}$$

The filter is fully characterised by its impulse response, $h(t)$.

A.3.1.2 Transfer Function

The Laplace transform of the impulse response is termed the Transfer Function, $H(s)$. When evaluated with $s = j\omega$, we obtain the frequency response of the filter, $H(j\omega)$. The frequency response conveys the following:

Magnitude Response:

The absolute value of the frequency response is found as:

$$|H(j\omega)| \tag{A.11}$$

which provides the amplitude or magnitude for every frequency ω, similar to the Fourier Transform magnitude spectrum.

Phase Response: The angle (or phase) of the frequency response, denoted as θ, offers insights into the phase shift introduced by the filter at each frequency:

$$\phi = \arg(H(j\omega)) \tag{A.12}$$

A.4 z-Transformation

In Digital Signal Processing (DSP), signals are discrete (or sampled). This means the Laplace transform, which is used for continuous signals, is not directly applicable. However, if we consider sampled signals as a sum of shifted impulses, we can still use the concept of Laplace transform and then transition to the more suitable z-transform (Porr, 2020).

Given a sampled signal:

$$x(t) = \sum_{n=0}^{\infty} x(n)\delta(t - nT) \tag{A.13}$$

where $\delta(t)$ is the Dirac delta function and T is the sampling period.
If we feed this signal into the Laplace transform, we get:

$$X(s) = \sum_{n=0}^{\infty} x(n)e^{-snT} \tag{A.14}$$

Expanding upon this, we can express $X(s)$ as:

$$X(s) = \int_0^{\infty} x(t)e^{-st}dt \tag{A.15}$$

$$= \sum_{n=0}^{\infty} x(n) \int_0^{\infty} \delta(t - nT)e^{-st}dt \tag{A.16}$$

$$= \sum_{n=0}^{\infty} x(n)(e^{-sT})^n \tag{A.17}$$

If we introduce a new complex variable z such that:

$$z^{-1} = e^{-sT} \tag{A.18}$$

we arrive at the z-transform:

$$X(z) = \sum_{n=0}^{\infty} x(n)z^{-n} \tag{A.19}$$

Here, $e^{-sT} = z^{-1}$ acts as a unit delay in the z-domain, meaning it delays the signal by one sample. Referring to time shift property of Laplace transform in Equation A.5, this is analogous to the e^{-sT} term in the Laplace domain that introduces a delay of T seconds.

A.5 Finite Impulse Response (FIR) Filters

Finite Impulse Response (FIR) filters are a class of digital filters that have a finite number of non-zero impulse response values. One way to derive an FIR filter is by sampling the impulse response of an analogue filter. An impulse response of an analogue filter, can be

formulated as:

$$h(t) = \sum_{n=0}^{\infty} h(nT)\delta(t - nT) \tag{A.20}$$

When we transform this impulse response into the Laplace domain, we have:

$$H(s) = \sum_{n=0}^{\infty} h(nT)(e^{-sT})^n \tag{A.21}$$

This can be further represented in terms of the z-transform. It is worth noting that e^{-sT} can be interpreted as a delay by one time step (as given by Equation A.5). Consequently, z^{-n} represents a delay by n time steps. This can be mathematically expressed as:

$$H(z) = \sum_{n=0}^{\infty} h(nT)(z^{-1})^n \tag{A.22}$$

This equation gives the z-transform of the impulse response $h(t)$ of the filter. When we want to filter a signal $X(z)$ using $H(z)$, we can use the following formula:

$$H(z)X(z) = \sum_{n=0}^{\infty} h(nT)z^{-n}\ X(z) \tag{A.23}$$

This summation offers a direct procedure to filter the signal $X(z)$. To set up a digital filter (as depicted in Fig. 16), all that is required is the impulse response of the filter, $h(nT)$. However, in practical scenarios, this impulse response does not extend to infinity. Instead, it spans a limited number of samples, often referred to as "taps". For instance, a filter utilising 100 samples of $h(nT)$ has 100 "taps". This necessitates a delay line capable of holding $M = 100$ samples:

$$H(z)X(z) = \sum_{m=0}^{M} h(mT)z^{-m}X(z) \tag{A.24}$$

This represents the equation for a Finite Impulse Response filter, obtained by sampling an analogue impulse response $h(t)$ at time intervals of T. When we transition into the time domain, the FIR filter performs the following operation:

$$y(n) = \sum_{m=0}^{M-1} h(m)x(n - m) \tag{A.25}$$

A.5.1 Convolution

While FIR filters draw inspiration from the concept of convolution, they do not perform a perfect convolution operation.

In the above section the FIR filter was derived from the analogue domain. When represented in the time domain, this translates to a discrete and causal convolution operation, primarily using the sampled Laplace transform. However, this process encounters a significant hurdle: it extends towards infinite time. This necessitates an infinite number of delay steps, which would practically take infinite time to execute.

Therefore, it is more appropriate to state that FIR filters approximate the convolution operation which arises due to the limitations posed by the finite number of taps (N). To further complicate matters, to compensate for the limitations of these taps, a technique called windowing is employed. While this aids in the design and performance of FIR filters, it simultaneously pushes the filter's operation further from the pure convolution operation (Porr, 2020).

Appendix B

Laplace Operator

The Laplace operator is used as a spatial filter to improve the spatial resolution of EEG signals. It involves subtracting the weighted average of signals received from neighbouring electrodes from from each electrode's signal. This process aims to emphasise signal features that are specific to a particular electrode while reducing the contribution of common sources of noise that are shared across neighbouring electrodes (Makeyev et al., 2016).

Mathematically, for a given electrode at a particular time point, the Laplace-transformed signal is obtained by taking the difference between the electrode's signal and the average of its neighbouring electrode signals, each multiplied by a weight. Consider a discrete EEG signal recorded at a specific electrode i and time point t. The Laplace-transformed signal $L_i(t)$ for that electrode and time point can be calculated as:

$$L_i(t) = X_i(t) - \frac{1}{n-1} \sum_{j=1}^{n-1} w_{ij} X_j(t) \tag{B.1}$$

Where $L_i(t)$ is the Laplace-transformed signal for electrode i at time point t. $X_i(t)$ is the raw EEG signal recorded at electrode i and time point t. n is the total number of electrodes. w_{ij} are the weights assigned to the neighbouring electrodes (Besio et al., 2006).

Appendix C

Least Mean Square (LMS) Filters

Another significant technique for noise cancellation, equalisation, and adaptive signal processing is the LMS-based FIR filter (Widrow et al., 1975). The effectiveness of DNF is also assessed against this technique in the results section of this chapter. An LMS-based FIR filter refers to a digital FIR filter that uses the least mean-squares (LMS) algorithm for adaptive filtering (Hayes, 1996). The goal of such a filter is to adjust its coefficients in real-time to minimise the difference between the desired output and the actual output of the filter. The operational procedure of LMS filters can be broken down into five distinct steps:

C.1 Filter Coefficients

Consider an FIR filter with coefficients $h[k]$, where k is the index of the coefficient. The input signal at time n is denoted as $x[n]$, and the desired output (target) at time n is denoted as $d[n]$.

C.2 Output Calculation

The output of the FIR filter at time n is calculated as the sum of weighted input samples:

$$y[n] = \sum_{k=0}^{M-1} h[k] \cdot x[n-k] \tag{C.1}$$

where M is the number of filter taps (coefficients).

C.3 Error Calculation

The error at time n is the difference between the desired output $d[n]$ and the actual output $y[n]$:

$$e[n] = d[n] - y[n] \tag{C.2}$$

C.4 Coefficient Update

The LMS algorithm updates the filter coefficients based on the error and the input signal. The update equation for the k^th coefficient is given by:

$$h[k](n+1) = h[k](n) + \mu \cdot e[n] \cdot x[n-k] \tag{C.3}$$

where μ is the learning rate, controlling the speed of coefficient adjustments.

C.5 Iteration

The above steps are performed iteratively for each time instant n, adapting the filter coefficients in such a way that the error is minimised over time.

Bibliography

Abbott, L. F. and Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nature neuroscience*, 3(11):1178–1183.

Aghaei-Lasboo, A., Inoyama, K., Fogarty, A. S., Kuo, J., Meador, K. J., Walter, J. J., Le, S. T., Graber, K. D., Razavi, B., and Fisher, R. S. (2020). Tripolar concentric eeg electrodes reduce noise. *Clinical Neurophysiology*, 131(1):193–198.

Ahmadi, A., Dehzangi, O., and Jafari, R. (2012). Brain-computer interface signal processing algorithms: A computational cost vs. accuracy analysis for wearable computers. In *2012 Ninth International Conference on Wearable and Implantable Body Sensor Networks*, pages 40–45.

Alpaydin, E. (2016). *Machine learning: the new AI*. MIT press.

Andresen, S. L. (2002). John mccarthy: father of ai. *IEEE Intelligent Systems*, 17(5):84–85.

Ashby, W. R. (1956). An introduction to cybernetics.

Bahador, N., Erikson, K., Laurila, J., Koskenkari, J., Ala-Kokko, T., and Kortelainen, J. (2020). A correlation-driven mapping for deep learning application in detecting artifacts within the EEG. *Journal of Neural Engineering*, 17(5):056018.

Bellman, R. (1957). A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684.

Bellman, R. (1966). Dynamic programming. *Science*, 153(3731):34–37.

Bennett, M. (2000). The concept of long term potentiation of transmission at synapses. *Progress in neurobiology*, 60(2):109–137.

Berthoud, H. (2004). Mind versus metabolism in the control of food intake and energy balance. *Physiol Behav*, 81(5):781–793.

Besio, G., Koka, K., Aakula, R., and Dai, W. (2006). Tri-polar concentric ring electrode development for laplacian electroencephalography. *IEEE Transactions on Biomedical Engineering*, 53(5):926–933.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press.

Bliss, G. A. (1930). The problem of lagrange in the calculus of variations. *American Journal of Mathematics*, 52(4):673–744.

Bliss, T. and Lomo, T. (1973). Long-lasting potentiation of synaptic transmission in the dentrate area of the anaesthetized rabbit following stimulation of the perforant path. *J Physiol*, 232(2):331–356.

Bliss, T. V. and Lømo, T. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *The Journal of physiology*, 232(2):331–356.

Boltyanskii, V. G., Gamkrelidze, R. V., and Pontryagin, L. S. (1960). The theory of optimal processes. i. the maximum principle. *Izv. Akad. Nauk SSSR Ser. Mat*, 24(1):3–42.

Botvinick, M. and Weinstein, A. (2014). Model-based hierarchical reinforcement learning and human action control. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1655):20130480.

Breitenfeld, T., Jurasic, M., and Breitenfeld, D. (2014). Hippocrates: the forefather of neurology. *Neurological Sciences*, 35(9):1349–1352.

Britton, J. W., Frey, L. C., Hopp, J. L., Korb, P., Koubeissi, M. Z., Lievens, W. E., Pestana-Knight, E. M., St. Louis, E. K., and Frey, L. C. (2016). *Electroencephalography (EEG): An Introductory Text and Atlas of Normal and Abnormal Findings in Adults, Children, and Infants*. American Epilepsy Society, 1 edition.

Bromberg-Martin, E. S., Matsumoto, M., and Hikosaka, O. (2010). Dopamine in motivational control: rewarding, aversive, and alerting. *Neuron*, 68(5):815–34.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Buchanan, B. G. and Smith, R. G. (1988). Fundamentals of expert systems. *Annual review of computer science*, 3(1):23–58.

Campbell, M., Hoane Jr, A. J., and Hsu, F.-h. (2002). Deep blue. *Artificial intelligence*, 134(1-2):57–83.

Canolty, R. T. and Knight, R. T. (2010). The functional role of cross-frequency coupling. *Trends in cognitive sciences*, 14(11):506–515.

Castellani, G. C., Quinlan, E. M., Cooper, L. N., and Shouval, H. Z. (2001). A biophysical model of bidirectional synaptic plasticity: Dependence on AMPA and NMDA receptors. *Proc. Natl. Acad. Sci. (USA)*, 98(22):12772–12777.

Chauvin, Y. and Rumelhart, D. E. (2013). *Backpropagation: theory, architectures, and applications*. Psychology press.

Chen, Y.-H., de Beeck, M., Vanderheyden, L., Carrette, E., Mihajlovic, V., Vanstreels, K., Grundlehner, B., Gadeyne, S., Boon, P., and Van Hoof, C. (2014). Soft, comfortable polymer dry electrodes for high quality ecg and eeg recording. *Sensors*, 14(12):23758–23780.

Clark, D. (1997). Deep thoughts on deep blue. *IEEE Intelligent Systems*, 12(04):31–31.

Cohen, J. Y., Amoroso, M. W., and Uchida, N. (2015). Serotonergic neurons signal reward and punishment on multiple timescales. *eLife*, 4:e06346.

Craik, A., He, Y., and Contreras-Vidal, J. L. (2019). Deep learning for electroencephalogram (EEG) classification tasks: a review. *Journal of Neural Engineering*, 16(3):031001.

Crockett, M. J., Clark, L., and Robbins, T. W. (2009). Reconciling the role of serotonin in behavioral inhibition and aversion: acute tryptophan depletion abolishes punishment-induced inhibition in humans. *J. Neurosci.*, 29(38):11993–11999.

Damasio, H., Grabowski, T., Frank, R., Galaburda, A. M., and Damasio, A. R. (1994). The return of phineas gage: clues about the brain from the skull of a famous patient. *Science*, 264(5162):1102–1105.

Daryanavard, S. and Porr, B. (2020a). Closed-loop deep learning: generating forward models with backpropagation. *Neural Computation*, 32(11):2122–2144.

Daryanavard, S. and Porr, B. (2020b). Sama-Darya/CLDL: Flexible closed-loop deep learning.

Daryanavard, S. and Porr, B. (2020c). Sama-Darya/lineFollowerRobot: Physical line-follower robot with deep learning in a closed-loop platform.

Dayan, P. and Balleine, B. W. (2002). Reward, motivation, and reinforcement learning. *Neuron*, 36(2):285–298.

Dela Cruz, J. A. D., Coke, T., and Bodnar, R. J. (2016). Simultaneous detection of c-fos activation from mesolimbic and mesocortical dopamine reward sites following naive sugar and fat ingestion in rats. *J. Vis. Exp.*, (114).

Delorme, A., Sejnowski, T., and Makeig, S. (2007). Enhanced detection of artifacts in eeg data using higher-order statistics and independent component analysis. *NeuroImage*, 34(4):1443–1449.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE.

Diniz, P. S., Da Silva, E. A., and Netto, S. L. (2010). *Digital signal processing: system analysis and design*. Cambridge University Press.

Dolan, R. J. and Dayan, P. (2013). Goals and habits in the brain. *Neuron*, 80(2):312–325.

Dwivedi, Y. K., Hughes, L., Ismagilova, E., Aarts, G., Coombs, C., Crick, T., Duan, Y., Dwivedi, R., Edwards, J., Eirug, A., et al. (2021). Artificial intelligence (ai): Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy. *International Journal of Information Management*, 57:101994.

Fatourechi, M., Bashashati, A., Ward, R. K., and Birch, G. E. (2007). Emg and eog artifacts in brain computer interface systems: A survey. *Clinical neurophysiology*, 118(3):480–494.

Feldman, A. G. (1986). Once more on the equilibrium-point hypothesis (λ model) for motor control. *Journal of motor behavior*, 18(1):17–54.

Fitzgibbon, S. P., DeLosAngeles, D., Lewis, T. W., Powers, D. M. W., Whitham, E. M., Willoughby, J. O., and Pope, K. J. (2015). Surface Laplacian of scalp electrical signals and independent component analysis resolve EMG contamination of electroencephalogram. *Int J Psychophysiol*, 97(3):277–84.

Fitzgibbon, S. P., Powers, D. M. W., Pope, K. J., and Clark, C. R. (2007). Removal of eeg noise and artifact using blind source separation. *Journal of clinical neurophysiology : official publication of the American Electroencephalographic Society*, 24(3):232–243.

Garcia-Casado, J., Ye-Lin, Y., Prats-Boluda, G., and Makeyev, O. (2019). Evaluation of bipolar, tripolar, and quadripolar laplacian estimates of electrocardiogram via concentric ring electrodes. *Sensors*, 19(17):3780.

Ghosh, A. and Ghosh, A. (2015). The engineered systems. *Dynamic Systems for Everyone: Understanding How Our World Works*, pages 19–41.

Glickstein, M. (2006). Golgi and cajal: The neuron doctrine and the 100th anniversary of the 1906 nobel prize. *Current Biology*, 16(5):R147–R151.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Green, R. M., Messick, W. J., Ricotta, J. J., Charlton, M. H., Satran, R., McBride, M. M., and DeWeese, J. A. (1985). Benefits, shortcomings, and costs of EEG monitoring. *Ann. Surg.*, 201(6):785–92.

Gross, C. G. (2007). The discovery of motor cortex and its background. *Journal of the History of the Neurosciences*, 16(3):320–331.

Grosz, B. J., Sparck-Jones, K., and Webber, B. L. (1986). *Readings in natural language processing*. Morgan Kaufmann Publishers Inc.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous deep q-learning with model-based acceleration. In *International conference on machine learning*, pages 2829–2838. PMLR.

Guger, C., Krausz, G., Allison, B. Z., and Edlinger, G. (2012). Comparison of dry and gel based electrodes for p300 brain-computer interfaces. *Front Neurosci*, 6:60.

Guo, X., Singh, S., Lee, H., Lewis, R. L., and Wang, X. (2014). Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *Advances in neural information processing systems*, pages 3338–3346.

Haber, S., Kunishio, K., Mizobuchi, M., and Lynd-Balta, E. (1995). The orbital and medial prefrontal circuit through the primate basal ganglia. *J Neurosci*, 15(7 Pt 1):4851–4867.

Hanin, B. (2018). Which neural net architectures give rise to exploding and vanishing gradients? *Advances in neural information processing systems*, 31.

Harbor, R. D. and Phillips, C. L. (2000). Feedback control systems. *(No Title)*.

Haruno, M., Wolpert, D. M., and Kawato, M. (2001). Mosaic model for sensorimotor learning and control. *Neural computation*, 13(10):2201–2220.

Hawkins, R. D., Kandel, E. R., and Siegelbaum, S. A. (1993). Learning to modulate transmitter release: themes and variations in synaptic plasticity. *Annual review of neuroscience*, 16(1):625–665.

Hayes, M. H. (1996). *Statistical digital signal processing and modeling*. John Wiley & Sons.

Hebb, D. (1949a). The organization of behavior; a neuropsychological theory.

Hebb, D. O. (1949b). *The organization of behavior: A neurophychological study*. Wiley-Interscience, New York.

Hebb, D. O. (2005). *The organization of behavior: A neuropsychological theory*. Psychology press.

Henry, J. C. (2006). Electroencephalography: basic principles, clinical applications, and related fields. *Neurology*, 67(11):2092–2092.

Hu, J., sheng Wang, C., Wu, M., xiao Du, Y., He, Y., and She, J. (2015). Removal of eog and emg artifacts from eeg using combination of functional link neural network and adaptive neural fuzzy inference system. *Neurocomputing*, 151:278 – 287.

Humphries, M. D., Stewart, R. D., and Gurney, K. N. (2006). A physiologically plausible model of action selection and oscillatory activity in the basal ganglia. *Journal of Neuroscience*, 26(50):12921–12942.

Iigaya, K., Fonseca, M. S., Murakami, M., Mainen, Z. F., and Dayan, P. (2018). An effect of serotonergic stimulation on learning rates for rewards apparent after long intertrial intervals. *Nature Communications*, 9(1):2477.

Imamizu, H. and Kawato, M. (2009). Brain mechanisms for predictive control by switching internal models: implications for higher-order cognitive functions. *Psychological Research PRPF*, 73:527–544.

Imamizu, H., Miyauchi, S., Tamada, T., Sasaki, Y., Takino, R., PuÈtz, B., Yoshioka, T., and Kawato, M. (2000). Human cerebellar activity reflecting an acquired internal model of a new tool. *Nature*, 403(6766):192–195.

Inglebert, Y., Aljadeff, J., Brunel, N., and Debanne, D. (2020). Synaptic plasticity rules with physiological calcium levels. *Proc Natl Acad Sci U S A*, 117(52):33639–33648.

Islam, M. K., Rastegarnia, A., and Yang, Z. (2016). Methods for artifact detection and removal from scalp eeg: A review. *Neurophysiologie Clinique/Clinical Neurophysiology*, 46(4):287 – 305.

Jafarifarmand, A. and Badamchizadeh, M. A. (2013). Artifacts removal in eeg signal using a new neural network enhanced adaptive filter. *Neurocomputing*, 103:222 – 231.

Jirayucharoensak, S., Israsena, P., Pan-ngum, S., and Hemrungrojn, S. (2013). Online eeg artifact suppression for neurofeedback training systems. In *The 6th 2013 Biomedical Engineering International Conference*, pages 1–5. IEEE.

Jirayucharoensak, S., Israsena, P., Pan-Ngum, S., Hemrungrojn, S., and Maes, M. (2019). A game-based neurofeedback training system to enhance cognitive performance in healthy elderly subjects and in patients with amnestic mild cognitive impairment. *Clinical interventions in aging*, 14:347–360.

Kawato, M. (1999). Internal models for motor control and trajectory planning. *Current opinion in neurobiology*, 9(6):718–727.

Kawato, M., Furukawa, K., and Suzuki, R. (1987). A hierarchical neural-network model for control and learning of voluntary movement. *Biological cybernetics*, 57:169–185.

Kirk, D. E. (2004). *Optimal control theory: an introduction*. Courier Corporation.

Klopf, A. H. (1986). A drive-reinforcement model of single neuron function: An alternative to the hebbian neuronal model. In *AIP Conference Proceedings*, volume 151, pages 265–270. American Institute of Physics.

Kopp, R. E. (1962). Pontryagin maximum principle. In *Mathematics in Science and Engineering*, volume 5, pages 255–279. Elsevier.

Krachunov, S. and Casson, A. (2016). 3d printed dry eeg electrodes. *Sensors*, 16(10):1635.

Kublik, S. and Saboo, S. (2022). *GPT-3*. O'Reilly Media, Incorporated.

Kulvicius, T., Kolodziejski, C., Tamosiunaite, M., Porr, B., and Wörgötter, F. (2010). Behavioral analysis of differential hebbian learning in closed-loop systems. *Biological cybernetics*, 103:255–271.

Kulvicius, T., Porr, B., and Wörgötter, F. (2007). Chained learning architectures in a simple closed-loop behavioural context. *Biological Cybernetics*, 97:363–378.

Larkum, M. (2013). A cellular mechanism for cortical associations: an organizing principle for the cerebral cortex. *Trends in neurosciences*, 36(3):141–151.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lee, S. S., Lee, K., and Kang, G. (2020). Eeg artifact removal by bayesian deep learning amp; ica. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, pages 932–935.

Lewis, F. L. and Vrabie, D. (2009). Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE circuits and systems magazine*, 9(3):32–50.

Lewis, F. L., Vrabie, D., and Vamvoudakis, K. G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine*, 32(6):76–105.

Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Li, Y., Zhong, W., Wang, D., Feng, Q., Liu, Z., Zhou, J., Jia, C., Hu, F., Zeng, J., Guo, Q., Fu, L., and Luo, M. (2016). Serotonin neurons in the dorsal raphe nucleus encode reward signals. *Nature Communications*, 7(1):10503.

Liao, L.-D., Wang, I.-J., Chen, S.-F., Chang, J.-Y., and Lin, C.-T. (2011). Design, fabrication and experimental validation of a novel dry-contact sensor for measuring electroencephalography signals without skin preparation. *Sensors (Basel)*, 11(6):5819–34.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016a). Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7:13276.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016b). Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., and Hinton, G. (2020). Backpropagation and the brain. *Nature Reviews Neuroscience*, 21:335–346.

Lindsay, G. W., Rigotti, M., Warden, M. R., Miller, E. K., and Fusi, S. (2017). Hebbian learning in a random network captures selectivity properties of the prefrontal cortex. *Journal of Neuroscience*, 37(45):11021–11036.

Linley, S. B., Hoover, W. B., and Vertes, R. P. (2013). Pattern of distribution of serotonergic fibers to the orbitomedial and insular cortex in the rat. *Journal of chemical neuroanatomy*, 48-49:29–45.

Linsker, R. (1988). Self-organization in a perceptual network. *Computer*, 21(3):105–117.

Lovinger, D. M. (2010). Neurotransmitter roles in synaptic modulation, plasticity and learning in the dorsal striatum. *Neuropharmacology*, 58(7):951–61.

Lu, W., Man, H., Ju, W., Trimble, W. S., MacDonald, J. F., and Wang, Y. T. (2001). Activation of synaptic NMDA receptors induces membrane insertion of new AMPA receptors and LTP in cultured hippocampal neurons. *Neuron*, 29(1):243–54.

Lund, B. D. and Wang, T. (2023). Chatting about chatgpt: how may ai and gpt impact academia and libraries? *Library Hi Tech News*, 40(3):26–29.

Luo, M., Zhou, J., and Liu, Z. (2015). Reward processing by the dorsal raphe nucleus: 5-HT and beyond. *Learn Mem*, 22(9):452–60.

Lüscher, C. and Malenka, R. C. (2012). Nmda receptor-dependent long-term potentiation and long-term depression (ltp/ltd). *Cold Spring Harbor perspectives in biology*, 4(6):a005710.

MacKinnon, C. D., Bissig, D., Chiusano, J., Miller, E., Rudnick, L., Jager, C., Zhang, Y., Mille, M.-L., and Rogers, M. W. (2007). Preparation of anticipatory postural adjustments prior to stepping. *Journal of neurophysiology*, 97(6):4368–4379.

Maffei, G., Herreros, I., Sanchez-Fibla, M., Friston, K. J., and Verschure, P. F. (2017). The perceptual shaping of anticipatory actions. *Proceedings of the Royal Society B: Biological Sciences*, 284(1869):20171780.

Makeig, S., Bell, A., Jung, T.-P., and Sejnowski, T. J. (1995). Independent component analysis of electroencephalographic data. In Touretzky, D., Mozer, M., and Hasselmo, M., editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press.

Makeyev, O. (2018). Solving the general inter-ring distances optimization problem for concentric ring electrodes to improve laplacian estimation. *BioMedical Engineering OnLine*, 17(1).

Makeyev, O., Ding, Q., and Besio, W. G. (2016). Improving the accuracy of laplacian estimation with novel multipolar concentric ring electrodes. *Measurement*, 80:44–52.

Malenka, R. C., Nicoll, and A, R. (1999). Long-term potentiation–a decade of progress? *Science*, 285(5435):1870–1874.

Markram, H., Gerstner, W., and Sjöström, P. J. (2011). A history of spike-timing-dependent plasticity. *Frontiers in synaptic neuroscience*, 3:4.

Markram, H., Lübke, J., Frotscher, M., and Sakman, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science*, 275:213–215.

Mateo, J., Torres, A. M., and García, M. A. (2013). Eye interference reduction in electroencephalogram recordings using a radial basis function. *IET Signal Processing*, 7(7):565–576.

Mattson, M. P., Maudsley, S., and Martin, B. (2004). Bdnf and 5-ht: a dynamic duo in age-related neuronal plasticity and neurodegenerative disorders. *Trends in Neurosciences*, 27(10):589–594.

McAdams, E. (2006). *Bioelectrodes*. American Cancer Society.

McCarthy, J. (1959). Programs with common sense.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.

McFarland, D., McCane, L., David, S., and Wolpaw, J. (1997). Spatial filter selection for eeg-based communication. *Electroencephalography and Clinical Neurophysiology*, 103(3):389–394.

McMenamin, B. W., Shackman, A. J., Maxwell, J. S., Bachhuber, D. R. W., Koppenhaver, A. M., Greischar, L. L., and Davidson, R. J. (2010). Validation of ica-based myogenic artifact correction for scalp and source-localized eeg. *NeuroImage*, 49(3):2416–2432.

Mehta, R. and Merhav, S. (1986). Performance characteristics of an adaptive controller based on least-mean-square filters. In *Astrodynamics Conference*, page 2160.

Meunier, C. N., Chameau, P., and Fossier, P. M. (2017). Modulation of synaptic plasticity in the cortex needs to understand all the players. *Frontiers in synaptic neuroscience*, 9:2.

Michie, D. (1963). Experiments on the mechanization of game-learning part i. characterization of the model and its parameters. *The Computer Journal*, 6(3):232–236.

Mishra, B. K. and Kumar, R. (2020). *Natural language processing in artificial intelligence*. CRC Press.

Miyamoto, H., Kawato, M., Setoyama, T., and Suzuki, R. (1988). Feedback-error-learning neural network for trajectory control of a robotic manipulator. *Neural networks*, 1(3):251–265.

Mizuhara, H. and Yamaguchi, Y. (2007). Human cortical circuits for central executive function emerge by theta phase synchronization. *Neuroimage*, 36(1):232–244.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Mountcastle, V. B. (1998). *Perceptual neuroscience: the cerebral cortex*. Harvard University Press.

Mulkey, R. M. and Malenka, R. C. (1992). Mechanisms underlying induction of homosynaptic long-term depression in area ca1 of the hippocampus. *Neuron*, 9(5):967–975.

Murphy, R. R. (2019). *Introduction to AI robotics*. MIT press.

Nathan, V. and Jafari, R. (2015). Design Principles and Dynamic Front End Reconfiguration for Low Noise EEG Acquisition With Finger Based Dry Electrodes. *IEEE Trans Biomed Circuits Syst*, 9(5):631–40.

Nguyen, H.-A. T., Musson, J., Li, F., Wang, W., Zhang, G., Xu, R., Richey, C., Schnell, T., McKenzie, F. D., and Li, J. (2012). Eog artifact removal using a wavelet neural network. *Neurocomputing*, 97:374–389.

Nilsson, N. J. (2009). *The quest for artificial intelligence*. Cambridge University Press.

Ntagios, M., Nassar, H., Pullanchiyodan, A., Navaraj, W. T., and Dahiya, R. (2019). Robotic hands with intrinsic tactile sensing via 3d printed soft pressure sensors. *Advanced Intelligent Systems*, 2(6):1900080.

Oja, E. (1997). The nonlinear pca learning rule in independent component analysis. *Neurocomputing*, 17(1):25–45.

OptiTrack (2019). Optitrack. June 2019.

O'Reilly, R. C. and Frank, M. J. (2006). Making Working Memory Work: A Computational Model of Learning in the Prefrontal Cortex and Basal Ganglia. *Neural Computation*, 18(2):283–328.

Pavlov, I. P. (1932). The reply of a physiologist to psychologists.

Penfield, W. and Rasmussen, T. (1950). The cerebral cortex of man; a clinical study of localization of function.

Phillips, C. and Harbor, R. (1991). *Feedback Control Systems*. Prentice Hall.

Phillips, C. L. and Harbor, R. D. (2000). *Feedback control systems*. Prentice-Hall, Inc.

Popa, L. S. and Ebner, T. J. (2019). Cerebellum, predictions and errors. *Frontiers in cellular neuroscience*, 12:524.

Porr, B. (2020). Digital signal processing lecture notes. `https://github.com/berndporr/digital_signal_processing`.

Porr, B. (2021). CLDL-CUDA: CUDA version of CLDL. `https://github.com/berndporr/CLDL-CUDA`.

Porr, B. and Daryanavard, S. (2020). Sama-Darya/enkiSimulator: Virtual line-follower robot with deep learning in a closed-loop platform.

Porr, B., Daryanavard, S., Bohollo, L. M., Cowan, H., and Dahiya, R. (2022). Real-time noise cancellation with deep learning. *Plos one*, 17(11):e0277974.

Porr, B. and Miller, P. (2020). Forward propagation closed loop learning. *Adaptive Behavior*, 28(3):181–194.

Porr, B. and Wörgötter, F. (2002a). Learning a forward model of a reflex. *Advances in neural information processing systems*, 15.

Porr, B. and Wörgötter, F. (2003). Isotropic Sequence Order learning. *Neural Comp.*, 15:831–864.

Porr, B. and Wörgötter, F. (2003). Isotropic-sequence-order learning in a closed-loop behavioural system. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 361(1811):2225–2244.

Porr, B. and Wörgötter, F. (2005). Inside embodiment – what means embodiment to radical constructivists? *Kybernetes*, 34:105–117.

Porr, B. and Wörgötter, F. (2006). Strongly improved stability and faster convergence of temporal sequence learning by using input correlations only. *Neural computation*, 18(6):1380–1412.

Porr, B. and Wörgötter, P. (2002b). Isotropic sequence order learning using a novel linear algorithm in a closed loop behavioural system. *Biosystems*, 67(1-3):195–202.

Porr, B. and Wörgötter, F. (2007). Learning with "Relevance": Using a Third Factor to Stabilize Hebbian Learning. *Neural Computation*, 19(10):2694–2719.

Prescott, T. J., González, F. M. M., Gurney, K., Humphries, M. D., and Redgrave, P. (2006). A robot model of the basal ganglia: behavior and intrinsic processing. *Neural networks*, 19(1):31–61.

Prinz, W. (2016). Ideo-motor action. In *Perspectives on perception and action*, pages 47–76. Routledge.

Proakis, J. G. (2007). *Digital signal processing: principles, algorithms, and applications, 4/E*. Pearson Education India.

Reynolds, J. N. and Wickens, J. R. (2002). Dopamine dependent plasticity of corticostriatal synapses. *Neural Networks*, 15:507–521.

Roberts, A. C. (2011). The importance of serotonin for orbitofrontal function. *Biol. Psychiatry*, 69(12):1185–91.

Rohaizad, N., Mayorga-Martinez, C. C., Novotny, F., Webster, R. D., and Pumera, M. (2019). 3d-printed ag/agcl pseudo-reference electrodes. *Electrochemistry Communications*, 103:104–108.

Rolls, E. T. (2016). Reward systems in the brain and nutrition. *Annual review of nutrition*, 36:435–470.

Rolls, E. T. and Grabenhorst, F. (2008). The orbitofrontal cortex and beyond: From affect to decision-making. *Progress in Neurobiology*, 86(3):216–244.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Rummery, G. A. and Niranjan, M. (1994). *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK.

Russell, S. J. (2010). *Artificial intelligence a modern approach*. Pearson Education, Inc.

Salvo, P., Raedt, R., Carrette, E., Schaubroeck, D., Vanfleteren, J., and Cardon, L. (2012). A 3d printed dry electrode for ecg/eeg recording. *Sensors and Actuators A: Physical*, 174:96–102.

Samoili, S., Cobo, M. L., Gómez, E., De Prato, G., Martínez-Plumed, F., and Delipetrev, B. (2020). Ai watch. defining artificial intelligence. towards an operational definition and taxonomy of artificial intelligence.

Sanders, J. and Kandrot, E. (2010). *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional.

Saudargiene, A., Porr, B., and Wörgötter, F. (2005). Local learning rules: predicted influence of dendritic location on synaptic modification in spike-timing-dependent plasticity. *Biological cybernetics*, 92(2):128–138.

Schmidhuber, J. (2014). Who invented backpropagation. *More.[DL2]*.

Schmidhuber, J., Hochreiter, S., et al. (1997). Long short-term memory. *Neural Comput*, 9(8):1735–1780.

Schmidt, R. A. (1987). The acquisition of skill: Some modifications to the perception-action relationship through practice.

Schmidt, R. A. (2016). The acquisition of skill: Some modifications to the perception-action relationship through practice. In *Perspectives on perception and action*, pages 77–103. Routledge.

Schmidt, R. A., Lee, T. D., Winstein, C., Wulf, G., and Zelaznik, H. N. (2018). *Motor control and learning: A behavioral emphasis*. Human kinetics.

Schultz, W. (2004). Neural coding of basic reward terms of animal learning theory, game theory, microeconomics and behavioural ecology. *Curr Opin Neurobiol*, 14(2):139–147.

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275:1593–1599.

Schultz, W. and Suri, R. E. (2001). Temporal difference model reproduces anticipatory neural activity. *Neural Comp.*, 13(4):841–862.

Schwab, R. S. and Chock, Y. C. (1953). A circuit for checking both electrode continuity and resistance during EEG recording. *Electroencephalogr Clin Neurophysiol*, 5(3):447–9.

Sewak, M. (2019). *Deep reinforcement learning*. Springer.

Shadmehr, R., Smith, M. A., and Krakauer, J. W. (2010). Error correction, sensory prediction, and adaptation in motor control. *Annual review of neuroscience*, 33:89–108.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shouval, H. Z., Bear, M. F., and Cooper, L. N. (2002). A unified model of NMDA receptor-dependent bidirectional synaptic plasticity. *Proc. Natl. Acad. Sci. (USA)*, 99(16):10831–10836.

Siciliano, B., Khatib, O., and Kröger, T. (2008). *Springer handbook of robotics*, volume 200. Springer.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Skinner, B. F. (1963). Operant behavior. *American psychologist*, 18(8):503.

Song, S., Miller, K. D., and Abbott, L. F. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926.

Sutton, R. (1988a). Learning to predict by method of temporal differences. *Machine Learning*, 3(1):9–44.

Sutton, R. S. (1988b). Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Swazey, J. P. (1968). Sherrington's concept of integrative action. *Journal of the History of Biology*, pages 57–89.

Szeliski, R. (2022). *Computer vision: algorithms and applications*. Springer Nature.

Szepesvári, C. (2022). *Algorithms for reinforcement learning*. Springer Nature.

Takahashi, Y. K., Batchelor, H. M., Liu, B., Khanna, A., Morales, M., and Schoenbaum, G. (2017). Dopamine neurons respond to errors in the prediction of sensory features of expected rewards. *Neuron*, 95(6):1395–1405.e3.

Tamosiunaite, M., Porr, B., and Wörgötter, F. (2007). Self-influencing synaptic plasticity: Recurrent changes of synaptic weights can lead to specific functional properties. *Journal of Computational Neuroscience*, 23(1):113–127.

Thorndike, E. L. (1927). The law of effect. *The American journal of psychology*, 39(1/4):212–222.

Todorov, E. (2004). Optimality principles in sensorimotor control. *Nature neuroscience*, 7(9):907–915.

Tsukahara, N. and Kawato, M. (1982). Dynamic and plastic properties of the brain stem neuronal networks as the possible neuronal basis of learning and memory. In *Competition and Cooperation in Neural Nets: Proceedings of the US-Japan Joint Seminar held at Kyoto, Japan February 15–19, 1982*, pages 430–441. Springer.

Turban, E. (1995). *Decision support and expert systems Management support systems*. Prentice-Hall, Inc.

Turing, A. M. (2012). Computing machinery and intelligence (1950). *The Essential Turing: the Ideas That Gave Birth to the Computer Age*, pages 433–464.

Turrigiano, G. G. (2008). The self-tuning neuron: synaptic scaling of excitatory synapses. *Cell*, 135(3):422–435.

Uden, L. and Guan, S. (2022). Neuroscience and artificial intelligence. In *Handbook of Research on New Investigations in Artificial Life, AI, and Machine Learning*, pages 212–241. IGI Global.

Umlauf, C. W. (1948). A Simplified Basal Electrode for Routine EEG Use. *Science*, 107(2770):121.

Urigüen, J. A. and Garcia-Zapirain, B. (2015). EEG artifact removal—state-of-the-art and guidelines. *Journal of Neural Engineering*, 12(3):031001.

Velcescu, A., Lindley, A., Cursio, C., Krachunov, S., Beach, C., Brown, C. A., Jones, A. K. P., and Casson, A. J. (2019). Flexible 3D-Printed EEG Electrodes. *Sensors (Basel)*, 19(7).

Verschure, P. F. and Coolen, A. C. (1991). Adaptive fields: Distributed representations of classically conditioned associations. *Network: Computation in Neural Systems*, 2(2):189–206.

von Helmholtz, H. (1925). Helmholtz's treatise on physiological optics, (southall jp, transl.). *New York: Optical Society of America*.

von Uexküll, B. J. J. (1926). *Theoretical biology*. Kegan Paul, Trubner, London.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Webb, L., Kauppila, M., Roberts, J. A., Vanhatalo, S., and Stevenson, N. J. (2021). Automated detection of artefacts in neonatal eeg with residual neural networks. *Computer Methods and Programs in Biomedicine*, 208:106194.

Webster, R. (2001). *Neurotransmitters, drugs and brain function*. John Wiley & Sons.

Werbos, P. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. *PhD thesis, Committee on Applied Mathematics, Harvard University, Cambridge, MA*.

Widrow, B., Glover, J., McCool, J., Kaunitz, J., Williams, C., Hearn, R., Zeidler, J., Eugene Dong, J., and Goodlin, R. (1975). Adaptive noise cancelling: Principles and applications. *Proceedings of the IEEE*, 63(12):1692–1716.

Wiener, N. and von Neumann, J. (1949). Cybernetics or control and communication in the animal and the machine. *Physics Today*, 2(5):33–34.

Wiering, M. A. and Van Otterlo, M. (2012). Reinforcement learning. *Adaptation, learning, and optimization*, 12(3):729.

Winston, P. H. (1984). *Artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc.

Wolpert, D. M., Diedrichsen, J., and Flanagan, J. R. (2011). Principles of sensorimotor learning. *Nature reviews neuroscience*, 12(12):739–751.

Wolpert, D. M. and Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural networks*, 11(7-8):1317–1329.

Wood, J., Simon, N. W., Koerner, F. S., Kass, R. E., and Moghaddam, B. (2017). Networks of VTA Neurons Encode Real-Time Information about Uncertain Numbers of Actions Executed to Earn a Reward. *Front Behav Neurosci*, 11:140.

Wörgötter, F. and Porr, B. (2005). Temporal sequence learning, prediction, and control: a review of different models and their relation to biological mechanisms. *Neural computation*, 17(2):245–319.

Yang, B., Duan, K., Fan, C., Hu, C., and Wang, J. (2018). Automatic ocular artifacts removal in eeg using deep learning. *Biomedical Signal Processing and Control*, 43:148 – 158.

Yu, J., Li, C., Lou, K., Wei, C., and Liu, Q. (2022). Embedding decomposition for artifacts removal in EEG signals. *Journal of Neural Engineering*, 19(2):026052.

Zenke, F., Gerstner, W., and Ganguli, S. (2017). The temporal paradox of hebbian learning and homeostatic plasticity. *Current opinion in neurobiology*, 43:166–176.

Zhang, H., Zhao, M., Wei, C., Mantini, D., Li, Z., and Liu, Q. (2021). EEGdenoiseNet: a benchmark dataset for deep learning solutions of EEG denoising. *Journal of Neural Engineering*, 18(5):056057.

Zou, F., Shen, L., Jie, Z., Zhang, W., and Liu, W. (2019). A sufficient condition for convergences of adam and rmsprop. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 11127–11135.