POLITECNICO DI TORINO

MASTER's Degree in Data Science and Engineering

MASTER's Degree Thesis

# Time-Robust and Energy-Efficient Decoder for Real-Time Neural Decoding of Primary Motor Cortex Activity

Supervisors

Prof. Stefano DI CARLO

Prof. Alessandro SAVINO

Dr. Alessio CARPEGNA

Dr. Paolo VIVIANI

Candidate

Marco TASCA

October 2024

# Summary

The human brain may be the most complex organ we know, and despite significant advances, our understanding of its structure and function remains limited. We have only begun to uncover the intricate network of billions of neurons and their countless synaptic connections that enable the brain's continuous activity. Although much remains unknown, the pursuit of deeper insight into the brain's workings is a driving force in scientific research. In parallel with neuroscience, the field of Artificial Intelligence (AI) has emerged as a rapidly evolving area of study, with the potential to revolutionize our understanding and interaction with the brain.

It is no coincidence that the latter is inspired by the former—an attempt by humans to mimic nature. Within the expansive domains of Machine Learning, Data Science, and Neural Networks, a new revolution is underway. We are moving beyond traditional architectures and embracing a novel neuromorphic paradigm, commonly known as Spiking Neural Networks (SNNs). This third generation of AI algorithms presents both unresolved challenges and exciting new opportunities. Notably, SNNs are capable of learning complex temporal patterns while exhibiting significantly lower power requirements compared to conventional neural networks. Moreover, due to their inherent similarities to biological systems, they are supposedly well-suited for analyzing brain signals.

In recent years, neuroengineering and Artificial Intelligence have increasingly collaborated, leading to remarkable advancements. Linear filters, are gradually being replaced by more precise and powerful tools. This synergy has enabled significant progress, including algorithms that can read images from the human visual cortex, allow paralyzed patients to type on virtual keyboards, control a mouse without hands, and even play mind-controlled games.

Our study is to be collocated in the context of the European Project B-Cratos, aimed to develop a closed-loop Brain Computer Interface to control a robotic prosthesis. We will explore the potential and efficiency of this new class of neuromorphic algorithms for analyzing brain signals. Specifically, we will develop a Spiking Neural Network (SNN)-based decoder for a Brain-Machine Interface (BMI). The purpose of this decoder will be to interpret 2D hand kinematics from brain signals collected through Electrocorticography (ECoG) from the Primary Motor

Cortex (M1) of a nonhuman primate, Indy. Additionally, we will investigate a new method for preprocessing the signal, called Spiking Band Power (SBP), and its long-term stability compared to more traditional approaches, as well as unsupervised and supervised adaptation techniques to efficiently address the evolution in M1's firing dynamics through time. Our challenge is to presents a model able to reduce time consumption and power requirements, while enhancing long-term stability, making it suitable for real-world applications where efficiency and responsiveness are critical.

As a member of the scientific community, and a newcomer to it, I am eager to share the work we have undertaken over the past year. Our research, though modest in scope, carries great hope: to contribute meaningfully to the collective body of scientific knowledge and, ultimately, to achieve something beneficial for humankind.

—

**Keywords**: Artificial Intelligence (AI), Spiking Neural Networks (SNNs), Brain-Machine Interface (BMI), neuromorphic paradigm, Primary Motor Cortex (M1), ECoG technology, 2D kinematics, neuroengineering, temporal patterns, supervised adaptation, unsupervised adaptation.

# Table of Contents

**List of Tables** ix

**List of Figures** x

**Acronyms** xiii

**1 Introduction** 1

1.1 Introduction to Neuromuscular Physiology . . . 2

1.2 Introduction to Brain Computer Interfaces . . . 2

1.3 Aim and Challenges . . . 3

1.3.1 Performance Degradation Over Time . . . 3

1.3.2 Efficiency and Hardware Constraints . . . 4

1.4 B-Cratos Project . . . 4

1.4.1 Key Features and Innovations of B-Cratos Project . . . 4

1.4.2 Contribution of Our Study . . . 5

**2 Background** 8

2.1 Anatomy and Physiology of Neuromuscular System . . . 8

2.1.1 Central and Peripheral Nervous System . . . 9

2.1.2 The Brain . . . 9

2.1.3 The Neurons . . . 10

2.1.4 The Skeletal Muscle System . . . 12

2.1.5 The Genesis of Voluntary Movements . . . 12

2.2 Brain Signal Acquisition Methods for BMIs . . . 15

2.2.1 Invasive Methods . . . 15

2.2.2 Non-Invasive Methods . . . 16

2.2.3 Advantages of ECoG . . . 17

2.3 Signal Cleaning and Processing Techniques (ECoG) . . . 18

2.3.1 Data Acquisition and Initial Cleaning . . . 18

2.3.2 Noise Reduction, Referencing Schemes, and Our Approach . 19

2.3.3 Spike Detection and Spike Sorting (MUA and SUA) . . . 20

2.3.4 Firing Rate Estimation (TCR) . . . . . . . . . . . . . . . . . 20
2.3.5 Most Common Processing Techniques . . . . . . . . . . . . . 21
2.4 Spiking Band Power Overview . . . . . . . . . . . . . . . . . . . . . 22
2.4.1 Understanding Spiking Band Power . . . . . . . . . . . . . . 23
2.4.2 Performance Advantages and Power Efficiency . . . . . . . . 23
2.4.3 Methodology for SBP Extraction . . . . . . . . . . . . . . . 24
2.4.4 Applications of SBP . . . . . . . . . . . . . . . . . . . . . . 24
2.4.5 Conclusions and Future Directions . . . . . . . . . . . . . . 24
2.5 Spiking Neural Networks . . . . . . . . . . . . . . . . . . . . . . . . 25
2.5.1 Fully Connected Neural Networks . . . . . . . . . . . . . . . 26
2.5.2 Recurrent Neural Networks . . . . . . . . . . . . . . . . . . 27
2.5.3 Common Spiking Neuron Models . . . . . . . . . . . . . . . 28
2.5.4 The Appeal of LIF Neurons . . . . . . . . . . . . . . . . . . 28
2.5.5 The LIF Neuron in Detail . . . . . . . . . . . . . . . . . . . 29
2.5.6 Backpropagation for FCNNs . . . . . . . . . . . . . . . . . . 32
2.5.7 Spatio-Temporal Backpropagation (STBP) for Regression in SNNs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
2.5.8 Optimization Variants: AdamW and Dropout . . . . . . . . 38
2.5.9 Overview of Spike Encoding Methods . . . . . . . . . . . . . 39
2.6 Adaptation Techniques for Temporal Robustness . . . . . . . . . . . 42
2.6.1 Unsupervised Adaptation Techniques . . . . . . . . . . . . . 43
2.6.2 Aligning Neural Data with PCA . . . . . . . . . . . . . . . . 44
2.6.3 Supervised Adaptation Techniques . . . . . . . . . . . . . . 46
2.6.4 Conclusion and Future Directions . . . . . . . . . . . . . . . 47

**3 Related Works** 48
3.1 State-of-the-Art Hand Velocity Decoding in BMIs . . . . . . . . . . 48
3.1.1 Traditional Filters: The Foundation . . . . . . . . . . . . . . 49
3.1.2 Deep Learning Decoders: Embracing Complexity . . . . . . 49
3.1.3 Spiking Neural Networks: High-Performing and Energy-Efficient . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
3.2 Long-Term Stability in Brain Signal Decoding . . . . . . . . . . . . 51
3.2.1 From Single-Unit Activity to Spike Band Power . . . . . . . 51
3.2.2 Adaptive Strategies for Long-Term Stability . . . . . . . . . 51

**4 Methods** 53
4.1 The choice of M1 over S1 . . . . . . . . . . . . . . . . . . . . . . . 53
4.2 Selected Preprocessing Methods . . . . . . . . . . . . . . . . . . . . 55
4.3 Preprocessing Details . . . . . . . . . . . . . . . . . . . . . . . . . . 55
4.3.1 MUA and SUA Practical Implementation . . . . . . . . . . . 56
4.3.2 SBP Practical Implementation . . . . . . . . . . . . . . . . . 57

4.3.3 Common Preprocessing Steps . . . . . . . . . . . . . . . . . 58
4.4 Unsupervised Adaptations with Normalization and PCA . . . . . . 58
4.4.1 Session Alignment Strategies based on Centering and Normalisation . . . . . . . . . . . . . . . . . . . . . . . . . . 59
4.4.2 Session Alignment Strategies based on PCA . . . . . . . . . 61
4.4.3 Principal Component Reordering and Sign Alignment . . . . 62
4.5 Supervised Adaptation with Fine-Tuning . . . . . . . . . . . . . . . 62
4.5.1 Two Baselines . . . . . . . . . . . . . . . . . . . . . . . . . 63
4.5.2 Fine-Tuning single layers . . . . . . . . . . . . . . . . . . . . 64
4.5.3 Efficiency of First-Layer Fine-Tuning . . . . . . . . . . . . . 64
4.6 Model Architecture and Configuration . . . . . . . . . . . . . . . . 64

**5 Results** 67
5.1 Dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
5.1.1 Data Acquisition and Preprocessing (SUA) . . . . . . . . . . 68
5.1.2 Session Selection and Preprocessing (MUA and SBP) . . . . 69
5.1.3 Behavioral Task . . . . . . . . . . . . . . . . . . . . . . . . 70
5.2 Metrics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71
5.2.1 Pearson Correlation Coefficient (CC) . . . . . . . . . . . . . 71
5.2.2 Root Mean Square Error (RMSE) . . . . . . . . . . . . . . . 71
5.2.3 Why We Used Both RMSE and CC . . . . . . . . . . . . . . 72
5.3 Single-Session Evaluation of SNN across MUA, SUA and SBP . . . 72
5.3.1 Comparison between preprocessing methods . . . . . . . . . 73
5.3.2 Spiking activity analysis . . . . . . . . . . . . . . . . . . . . 75
5.3.3 Training Loss analysis . . . . . . . . . . . . . . . . . . . . . 77
5.4 Unsupervised adaptations: Online Normalisation and PCA Alignment 78
5.4.1 Experiment 0: No Alignment . . . . . . . . . . . . . . . . . 79
5.4.2 Experiment 1: Impact of Online Vs. Offline Normalisation . 80
5.4.3 Experiment 2: Online Normalisation Vs. Online Centering . 86
5.4.4 Experiment 3: PCA Adaptation: Rotation and Ordering . . 87
5.4.5 Experiment 4: PCA Adaptation: Rotation and Scaling with Singular Values . . . . . . . . . . . . . . . . . . . . . . . . 90
5.4.6 Why PCA-based adaptations failed . . . . . . . . . . . . . . 91
5.5 Supervised Adaptation for Long-Term Stability . . . . . . . . . . . 92
5.5.1 Experiment 1: Extensive Retraining Across Sessions (Whole Model) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
5.5.2 Experiment 2: Short-Term Adaptations (Single Layers) . . . 95
5.6 Comparative Summary Analysis . . . . . . . . . . . . . . . . . . . . 97

**6 Conclusions and Future Works** 100

**Bibliography** 103

# List of Tables

2.1 Comparison between LIF neuron and perceptron . . . . . . . . . . . 31

4.1 Hyperparameters of our model. . . . . . . . . . . . . . . . . . . . . . 65

5.1 CC and RMSE for MUA, SUA, SBP1000 and SBP2000 (mean and std). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79
5.2 Statistical testing results of the Null Hp: MUA $\approx$ SBP, for both CC and RMSE. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
5.3 Performance metrics for SBP (centered and rescaled) . . . . . . . . 87
5.4 Performance metrics for MUA (centered and rescaled) . . . . . . . . 87
5.5 Performance metrics for SBP (offline and online normalised, with and without PCA) . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
5.6 Mean and standard deviation of CC and RMSE for the resumed and retrained models across sessions with SBP. . . . . . . . . . . . . . . 95
5.7 Statistical tests for layer 1 and layer 2 over CC and RMSE metrics. 97
5.8 Mean and standard deviation of CC and RMSE for different adaptations methods with SBP. . . . . . . . . . . . . . . . . . . . . . . . 98

# List of Figures

1.1 Overview of the B-Cratos project. Taken from [10]. . . . . . . . . . 6

2.1 Overview of brain anatomy. Taken from [12] . . . . . . . . . . . . . . 11
2.2 Simplified representation of the sensorimotor loop. M1 Cortex in green and S1 Cortex in blue. Taken from [14]. . . . . . . . . . . . . 13
2.3 The two maps present in our brain: the sensory and motor homunculus. Taken from [15] . . . . . . . . . . . . . . . . . . . . . . . . 14
2.4 Overview of brain signal recording methods. Taken from [20] . . . . 18
2.5 The main preprocessing steps to obtain SBP, MUA, and SUA. . . . 23
2.6 Structure and main components of a FCNN. Figure taken from [36]. 26
2.7 An RNN unfolded, with $S_0, S_1, ...$ being the internal states of the net at each time step. Taken from [37] . . . . . . . . . . . . . . . 27
2.8 LIF-neuron response to input spikes and behaviour of the membrane leaking dynamics . . . . . . . . . . . . . . . . . . . . . . . . . . 30
2.9 Comparison between LIF neuron and perceptron. . . . . . . . . . . 32
2.10 (a) Diagram of a spiking neuron. (b) The Sigmoid function used as surrogate gradient to approximate Heaviside function. Taken from [41]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
2.11 Error backpropagation through time in unfolded RNN. Image taken from [42]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36

3.1 In this figure from the work by Ahmadi et al. [3] we can see a comparison (CC and RMSE) of 8 different decoders over 26 sessions, spanning on 218 day. . . . . . . . . . . . . . . . . . . . . . . . . 50

4.1 MUA and SUA binning and summing spikes in 4 milliseconds time intervals. Figure taken from [48]. . . . . . . . . . . . . . . . . . . 57
4.2 Data folding into overlapping windows or "Sequences". Figure is taken from [48]. . . . . . . . . . . . . . . . . . . . . . . . . . . 59

5.1 Complete pipeline from acquisition to decoding of 2D hand kinematics of Indy (NHP). . . . . . . . . . . . . . . . . . . . . . . . . . 68

5.2 Example plot of three signals for CC and RMSE comparison. . . . . 73
5.3 Single session comparison between MUA, SUA, SBP1000 and SBP2000. 74
5.4 Evolution of CC and RMSE over epochs while evaluating on TSS and training on TRS. . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
5.5 (a) First hidden layer. (b) Second hidden layer. (c) Third. (d) Output non spiking layer (2 LI-Neurons). . . . . . . . . . . . . . . . 76
5.6 Training loss mean and std for each session (above). Real training loss batch by batch (below). Loss is MSE, data is preprocessed with SBP. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
5.7 Comparison between training loss and velocity of the finger. . . . . 78
5.8 Linear regression analysis for SBP and MUA without alignment (CC and RMSE). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 81
5.9 The different LIF neuron response to a signal with and without normalisation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
5.10 Linear regression analysis for SBP and MUA with online or offline normalisation (CC and RMSE). . . . . . . . . . . . . . . . . . . . . 83
5.11 Linear regression plots (above). Distribution of sessions over metric scores (below). Online normalisation comparison between MUA and SBP (CC and RMSE) . . . . . . . . . . . . . . . . . . . . . . . . . 85
5.12 Linear regression analysis for SBP and MUA. Online normalisation Vs online centering (CC and RMSE). . . . . . . . . . . . . . . . . 86
5.13 Linear regression analysis for 4 combinations of normalisation, PCA and reordering on SBP. . . . . . . . . . . . . . . . . . . . . . . . . 89
5.14 Similarity matrix of $W_{\text{ref}}$ and $W_{\text{inf}}$, before and after reordering. . . . 90
5.15 Linear regression analysis for PCA adaptation (rotation and scaling) on SBP . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 92
5.16 Summary of linear transformation methods for SBP . . . . . . . . . 93
5.17 Performance trends (CC and RMSE) for resumed and retrained models across sessions. . . . . . . . . . . . . . . . . . . . . . . . . . 95
5.18 CC and RMSE over time for short-term adaptations with SBP. Retraining all layers (Experiment 3) and individual layers (Experiment 4) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
5.19 Comparison between our 3 main results on both CC AND RMSE, with SBP. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99

# Acronyms

**AdEx**

adaptive exponential neuron

**AI**

artificial intelligence

**ANN**

artificial neural network

**ALS**

amyotrophic lateral sclerosis

**AP**

action potential

**BCI**

brain computer interface

**BMI**

brain machine interface

**BPTT**

back propagation through time

**CAR**

common average referencing

**CC**

Pearson's correlation coefficient

| | |
|---|---|
| **CCA** | canonical correlation coefficient |
| **CNS** | central nervous system |
| **DBS** | deep brain stimulation |
| **EEG** | electroencephalography |
| **ESA** | entire spiking activity |
| **ECoG** | Electrocorticography |
| **FCNN** | fully connected neural network |
| **FMRI** | functional magnetic resonance imaging |
| **fNIRS** | functional near infrared spectroscopy |
| **GRU** | gated recurrent unit |
| **HH** | Hodgkin-Huxley neuron |
| **IF** | integrate and fire |
| **IZH** | Izhikevich neuron |

| | |
|---|---|
| **KF** | Kalman filter |
| **LFP** | local field potential |
| **LI** | leaky integrate |
| **LIF** | leaky integrate and fire |
| **LSTM** | long short term memory |
| **M1** | primary motor cortex |
| **MEG** | magnetoencephalography |
| **MND** | motor neuron disease |
| **MSE** | mean square error |
| **MUA** | multi unit activity |
| **NLIF** | non linear leaky integrate and fire |
| **ON** | online normalisation |
| **PC** | principal components |

**PCA**

principal components analysis

**PNS**

peripheral nervous system

**QRNN**

quasi recurrent neural network

**RMSE**

root mean square error

**RNN**

recurrent neural network

**S1**

primary somatosensory cortex

**SBP**

spiking band power

**SCI**

spinal cord injury

**SEEG**

stereo-electroencephalography

**SMA**

supplementary motor area

**SNN**

spiking neural network

**SNR**

signal to noise ratio

**STBP**

spatio temporal back propagation

**STDP**

spiking time dependent plasticity

| | |
|---|---|
| **SUA** | singular unit activity |
| **TCR** | threshold crossing rate |
| **TEL** | trainable encoding layer |
| **UKF** | unscented Kalman filter |
| **WCF** | Wiener cascade filter |
| **WF** | Wiener filter |

# Chapter 1

# Introduction

Loss of motor ability and control over the upper and lower limbs, even when the brain remains intact, can result from several conditions, including spinal cord injuries (SCI), whether complete or incomplete, as well as by lower motor neuron diseases (MNDs) like amyotrophic lateral sclerosis (ALS), or amputation. These conditions disrupt the communication pathway between the brain and muscles, leading to severe impairments in movement and daily functioning. Addressing these challenges necessitates the development of highly efficient, fast-to-train, and adaptive Brain-Machine Interfaces (BMIs) that can accurately decode brain signals and restore motor function. Such solutions must be able to quickly adapt to the brain's evolving neural activity while being power-efficient and suitable for real-world applications. My thesis work is dedicated to advancing a neuromorphic approach using Spiking Neural Networks (SNNs) to create a BMI that meets these critical needs, overcoming the challenges of long-term stability and power-intensive decoding processes.

We made a deliberate architectural choice that comes with a significant assumption. Our approach was to implement the decoder as a component in the loop between M1, the Primary Motor Cortex, and S1, the Primary Somatosensory Cortex. By doing so, we chose not to process any feedback from S1, focusing solely on the motor commands from M1. We leave the task of processing feedback and making adjustments to the brain itself, as we believe strongly in the brain's superior adaptive capabilities. The primary goal of our decoder is to accurately interpret the underlying dynamics of M1.

In the following section, we will present a general overview of the neuromuscular system, we will introduce Brain-Computer Interfaces (BCIs) and their pivotal role in restoring lost motor functions. In connection with this, we will outline the aim and challenges of our work, alongside our collaboration with the Links Foundation and the European project B-Cratos.

## 1.1 Introduction to Neuromuscular Physiology

The neuromuscular system enables voluntary movement through the intricate coordination of brain regions and muscles. At the heart of this process is the motor cortex, which includes the Primary Motor Cortex (M1), the Premotor Cortex, and the Supplementary Motor Area (SMA). Together, these areas are responsible for planning, initiating, and executing voluntary movements. They send motor commands through neural pathways, eventually reaching the skeletal muscles to produce movement.

Adjacent to M1, the Primary Somatosensory Cortex (S1) is involved in processing sensory feedback, which is crucial for the fine-tuning of motor actions. This feedback loop helps ensure that movements are smooth and accurately executed.

In the context of brain-machine interfaces (BMIs), we focus on recording and interpreting the neural activity from M1 using Electrocorticography (ECoG), a method that captures the brain's electrical signals. The signals obtained from M1 will be the primary input for our decoder, which aims to translate these neural impulses into meaningful outputs for controlling external devices.

## 1.2 Introduction to Brain Computer Interfaces

Brain-computer interfaces (BCIs), also known as brain-machine interfaces (BMIs), are groundbreaking systems that forge a direct link between the brain and external devices, bypassing traditional neuromuscular pathways. By interpreting neural activity that reflects a user's intentions, BCIs convert these signals into commands for controlling various assistive technologies. This capability is transformative for individuals with severe motor impairments, such as those caused by spinal cord injuries, stroke, or ALS, empowering them to regain motor functions and enhance their quality of life through the control of computer cursors, robotic limbs, or electrical stimulation systems.

At the core of BCIs are sophisticated algorithms that decode neural activity into actionable commands. These range from classical techniques like linear regression to cutting-edge deep learning models. Although BCIs are closely associated with motor rehabilitation, their scope extends to communication, environmental control, entertainment, and cognitive enhancement. However, challenges remain, including ensuring the robustness and precision of decoding algorithms, achieving long-term stability of neural recordings, and navigating ethical concerns related to privacy and the risks of invasive methods. As our understanding of neuroscience deepens and technology advances, BCIs are poised to become increasingly refined and transformative in their applications.

## 1.3 Aim and Challenges

The primary objective of this research is to accurately and efficiently predict the two-dimensional kinematics of hand movements using neural signals from the Primary Motor Cortex (M1) obtained through Electrocorticography (ECoG). The study addresses two major challenges: sustaining decoder performance over time and optimizing energy and memory efficiency. Overcoming these challenges is essential for developing stable, low-power, implantable decoders that provide consistent and reliable performance in real-world BCI and BMI applications over extended periods. This will be achieved by experimenting with supervised learning techniques alongside a blend of supervised and unsupervised transfer learning methods. Additionally, we will investigate a new preprocessing approach, called Spiking Band Power (SBP), that, when combined with trainable feature encoding, is expected to improve long-term stability and reduce memory and energy demands. This effort will be supported by employing a novel neuromorphic decoder paradigm, based on Spiking Neural Networks (SNN), designed for exceptionally low power consumption.

### 1.3.1 Performance Degradation Over Time

Time performance degradation in brain-computer interfaces (BCIs), particularly those reliant on spike recordings, presents a significant obstacle to their long-term viability and clinical translation. This instability primarily stems from the brain's biological response to implanted electrodes and the inherent characteristics of neural recordings [1, 2].

The formation of glial scars, mostly in intracortical implants, is a major cause. This natural defense mechanism encapsulates the foreign electrodes, impeding their ability to effectively record neural activity [1]. The micromotion of electrodes, often exacerbated by the brain's natural movements, including those caused by blood flow, further compounds the problem. This movement disrupts the delicate interface between electrodes and neurons, leading to a degradation in signal quality and necessitating frequent recalibration [3, 2]. Insulation degradation is another significant factor that compromises signal reliability over time by causing signal leakage [1]. The brain's inherent plasticity, while crucial for learning and adaptation, presents an additional layer of complexity for BCI stability. As the brain dynamically reorganizes its neural connections, neural activity patterns shift, impacting the decoding performance of BCIs and demanding adjustments to maintain accuracy [4, 1].

The cumulative effect of these factors is a progressive decline in observable neurons and a weakening of recorded signals, ultimately limiting the longevity of spike-based BCIs. This degradation underscores the need for the right implant, and adaptation techniques to enhance the long-term stability of BCIs, as a necessary step to their successful clinical translation and long-term use.

### 1.3.2 Efficiency and Hardware Constraints

BCIs are designed for direct communication between the brain and external devices, this requires efficient algorithms and hardware. These systems need to analyze neural signals in real-time, to effectively control external tools, like computer cursors or prosthetic. This processing must be done efficiently, particularly for implantable BCIs with limited power, memory and computational resources [4, 5, 6].

In this field a significant challenge lies in balancing the complexity of decoding algorithms with the constraints of hardware implementations. More sophisticated algorithms, such as deep learning models, though demonstrably very accurate, require significantly more processing power and memory [7]. This demand clashes with the limited battery life and heat dissipation capabilities of implantable devices. This challenge necessitates ongoing research into efficient algorithms and specialized hardware like neuromorphic chips, as well as optimization techniques such as data quantization and compression. Striking a balance between decoding accuracy, speed, and power consumption remains crucial for creating robust and clinically viable BCIs.

## 1.4 B-Cratos Project

This master's thesis was conducted at the LINKS Foundation [8], a research institution established through a partnership between Compagnia di San Paolo and Politecnico di Torino, as well as at the SMILIES Lab [9] at Politecnico di Torino. LINKS specializes in applied research, digital technology, and innovation, with active projects both nationally and internationally.

This work is part of the B-Cratos (Wireless Brain-Connect inteRfAce TO machineS) project [10], a European initiative funded under the EU Horizon 2020 research and innovation program (grant agreement 965044). The primary goal of B-Cratos is to develop a wireless, bidirectional, and battery-free brain-machine interface (BMI) aimed at restoring hand functionality and sensory feedback for individuals with paralysis or amputation.

### 1.4.1 Key Features and Innovations of B-Cratos Project

**Design and Development of a High-Channel, High-Speed, Wireless Brain Implant** : A central focus of the project is the creation of a fully-implantable, battery-less brain interface capable of high-resolution neural signal sensing and precise cortical stimulation. This system utilizes the Utah Array, which is connected to a compact, biocompatible, and hermetically sealed implant housing custom electronics for amplifying, digitizing, and processing neural activity. An external wearable module, developed by NTNU researchers, facilitates wireless bidirectional communication using an innovative transmission technique.

**Fat Intra-Body Communication (FAT-IBC) Platform** : The project aims to establish a high-speed communication platform based on microwave propagation through subdermal body fat. This method, known as FAT-IBC, is designed to minimize interference from external electronic devices while ensuring reliable data transmission.

**HPC-Based AI Computing for BMI Control** : Advanced machine learning (ML) and deep learning (DL) algorithms are being developed to control arm movements and provide sensory feedback. These algorithms are crucial for pattern recognition and classification tasks.

**Development of Artificial Skin** : The sensory feedback system incorporates a novel combination of triboelectric nanogenerators (TENGs) and graphene-based hydrogels, capable of generating a time-dependent force map in digital format.

**Integration with a Biomechatronic Prosthetic Arm** : The project will integrate the 5-axis Mia robotic arm from Prensilia s.r.l., a spin-off of Scuola Superiore Sant'Anna. This prosthetic arm, an evolution of the IH2 Azzurra, is recognized for its enhanced strength, speed, and lightweight design, and is widely used in research institutions globally.

The B-Cratos BMI is classified as an invasive interface due to its use of intracortical microelectrode arrays (specifically the Utah Array by Blackrock Microsystems). These arrays are connected to a biocompatible implant that houses the necessary electronics for neural signal processing and stimulation. The system's wireless functionality is supported by an external wearable module that manages power supply and high-speed data transfer. The AI module plays a critical role in facilitating bidirectional communication between the brain and the prosthetic device, decoding neural signals from the implanted electrodes and encoding sensory inputs from the prosthetic hand's sensorized skin.

### 1.4.2 Contribution of Our Study

Our study significantly contributes to the B-Cratos project by enhancing the development of energy-efficient, stable, and real-time neural decoding systems suitable for long-term implantation. These contributions are crucial for achieving

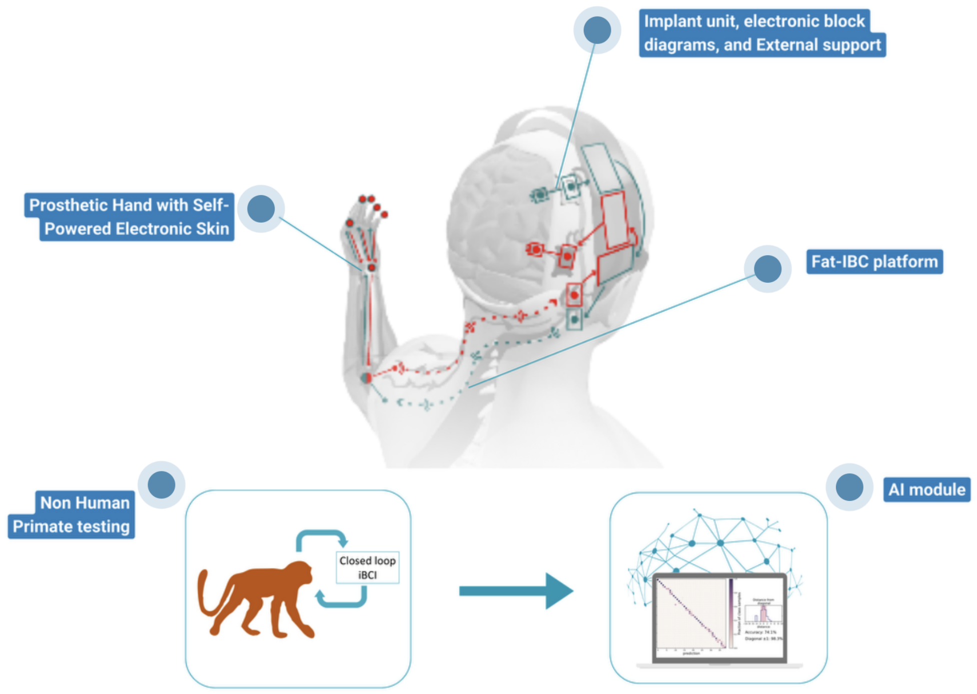

Implant unit, electronic block diagrams, and External support
Prosthetic Hand with Self-Powered Electronic Skin
Fat-IBC platform
Non Human Primate testing
Closed loop iBCI
AI module

**Figure 1.1:** Overview of the B-Cratos project. Taken from [10].

the project's goals of creating a high-performance, bidirectional Brain-Machine Interface (BMI) with minimal power consumption and long-term reliability.

Firstly, we propose the implementation of a neuromorphic, low-power decoder that may be directly implanted within the brain. This decoder is designed to function with minimal power requirements, thereby eliminating the necessity for a large power source. Its energy efficiency can also, with further optimisation studies, guarantee that it does not overheat the surrounding neural tissue, which is essential for the long-term safety and functionality of the implant.

In addition to the decoder, our research introduces a novel approach to preprocessing neural signals using Spiking Band Power (SBP). This is an efficient and straightforward method for processing raw neural data in real-time, making it particularly suitable for BMI applications. The SBP preprocessing technique effectively captures the relevant information while maintaining computational simplicity, thereby enhancing the overall efficiency and reliability of the system.

To further improve the utility of SBP in real-time decoding, we introduce a trainable feature encoder. This encoder is designed to translate the continuous SBP signals into discrete spikes that can then be fed directly into the neuromorphic decoder. The encoder is integrated in the decoder architecture, in this way the calibration is performed for both at the same time, with minimal computational overload. Moreover, it operates in real-time, maintaining the system's simplicity and efficiency while allowing for the effective transformation of neural data into a format suitable for neuromorphic processing. This integration of a trainable encoder ensures that the SBP method remains not only efficient but also highly adaptable to changes in neural signals and conditions, enhancing the potential long-term stability of our approach.

To further improve the time robustness of the proposed method, we conducted investigations on its behaviour over sessions recorded in an extended period (223 days). This study is critical for the viability of chronic neural implants.

Finally, we explored alternative strategies for rapid and efficient recalibration of the neural decoder, experimenting on both supervised and unsupervised adaptations. Given the dynamic nature of the neural interface environment, the ability to quickly recalibrate the decoder is vital for maintaining high performance. Our contributions in this area ensure that the B-Cratos system remains adaptable and robust, even in the face of changing neural conditions.

Through these advancements, our study tries to enhance the B-Cratos project's objective to develop a safe, efficient, and durable brain-machine interface, thus contributing to the creation of more advanced and clinically viable neural prosthetics.

# Chapter 2

# Background

In this chapter, we will introduce and explain the key topics present in our thesis, providing relevant citations along the way.

We will begin with an overview of the physiology of the motor cortex and the neural mechanisms that govern voluntary movements. Next, we will examine the most common methods for acquiring and preprocessing brain signals, followed by a more in-depth discussion of the specific preprocessing techniques we have selected for our analysis (detailed configurations will be provided in Chapter 4).

In particular, we will focus on spike sorting and threshold-based spike detection, including both single-unit activity (SUA) and multi-unit activity (MUA), as well as a detailed examination of Spiking Band Power (SBP) and its theoretical advantages.

We will then delve into spiking neural networks (SNNs), discussing various neuron structures, foundational paradigms, popular spike encoding methods, and training strategies. Additionally, we will explain how regression can be performed using SNNs, with a focus on our chosen training technique: Spatio-Temporal Backpropagation (STBP).

Finally, we will explore unsupervised and supervised adaptation techniques aimed at improving decoder robustness across multiple sessions. This will include an analysis of the theoretical foundations of unsupervised linear transformations, and a supervised approach which involves fine-tuning one or more layers of the network for a brief period on new session data. Technical details will be presented in Chapter 4.

## 2.1 Anatomy and Physiology of Neuromuscular System

Voluntary movements represent the culmination of a complex interaction between various actors and intricate structures, all working in concert to produce what appears to be an effortless action. Together, they constitute the neuromuscular system.

In the following section, we will provide a general overview of this phenomenon by introducing each key player and offering insight into their interactions. We will discuss the nervous system, skeletal muscles, and neurons, with a particular focus on the genesis of movement within the brain, and the role of the primary motor cortex (M1), where our sensor is placed. This introduction is intended as a broad overview rather than an exhaustive explanation.

### 2.1.1 Central and Peripheral Nervous System

The nervous system is a complex network that coordinates the body's activities by transmitting signals between different parts of the body. It is divided into two main components: the Central Nervous System (CNS) and the Peripheral Nervous System (PNS) [11, p. 171].

The CNS is the command center of the body, comprising the brain and spinal cord. The brain processes sensory information, initiates responses, stores memories, and generates thoughts and emotions. The spinal cord serves as a conduit for signals between the brain and the rest of the body, and it also controls simple reflexes that do not require brain input. The PNS consists of all the nerves that branch out from the brain and spinal cord, extending to other parts of the body, including the limbs and organs. It is further divided into the somatic nervous system, which controls voluntary movements and transmits sensory information, and the autonomic nervous system, which regulates involuntary functions such as heart rate and digestion.

### 2.1.2 The Brain

Within the central nervous system (CNS) lies the brain, a big and energy-demanding organ that serves as the central hub for regulating essential functions such as breathing, heart rate, and other vital processes. Additionally, it manages consciousness, memory and voluntary activities, including speech, walking, and thinking. In this region, movements are conceived, planned, and transmitted via the spinal cord to the muscles, with feedback integrated in a loop that ensures voluntary movements to be smooth and effective. The brain is organized into several key areas, each with distinct functional roles [11, pp. 175–202].

#### Cerebrum

The cerebrum is the largest and most prominent part of the brain, divided into two hemispheres—left and right—that each govern various mental and physical functions. Each hemisphere consists of four distinct lobes. The frontal lobe, located at the front, is crucial for high-level processes such as decision-making, problem-solving, and planning. The parietal lobe, situated on the top of the brain, is essential for integrating sensory information and facilitating spatial orientation. The temporal lobe, found on the sides, plays a vital role in processing auditory information and forming memories. Lastly, the occipital lobe, positioned at the rear, is dedicated to visual processing, converting light into meaningful visual experiences.

#### Cerebellum

Nestled beneath the cerebrum, the cerebellum oversees the coordination of movement and balance. It fine-tunes motor activities, ensuring smooth, precise actions and maintaining equilibrium.

#### Brainstem

The brainstem serves as a vital conduit between the brain and the spinal cord, regulating essential physiological processes necessary for life. The midbrain is responsible for processing visual and auditory stimuli, contributing to sensory perception and reflexes. Acting as a bridge, the pons facilitates communication between different brain regions and plays a role in regulating respiration. At the base of the brainstem, the medulla oblongata controls autonomic functions such as heartbeat and breathing, which are critical for survival.

#### Diencephalon

The diencephalon, nestled deep within the brain, encompasses the thalamus and hypothalamus. The thalamus serves as a relay station for sensory information, while the hypothalamus is instrumental in maintaining homeostasis, managing temperature, hunger, and circadian rhythms.

#### Limbic System

A key player in emotion and memory, the limbic system includes structures such as the hippocampus, crucial for forming new memories, and the amygdala, which processes emotional responses. This system is fundamental to our emotional experience and memory consolidation.

### 2.1.3 The Neurons

The brain, like the rest of the nervous system, is primarily composed of neurons, which are fundamental components of nervous tissue and essential for its function.

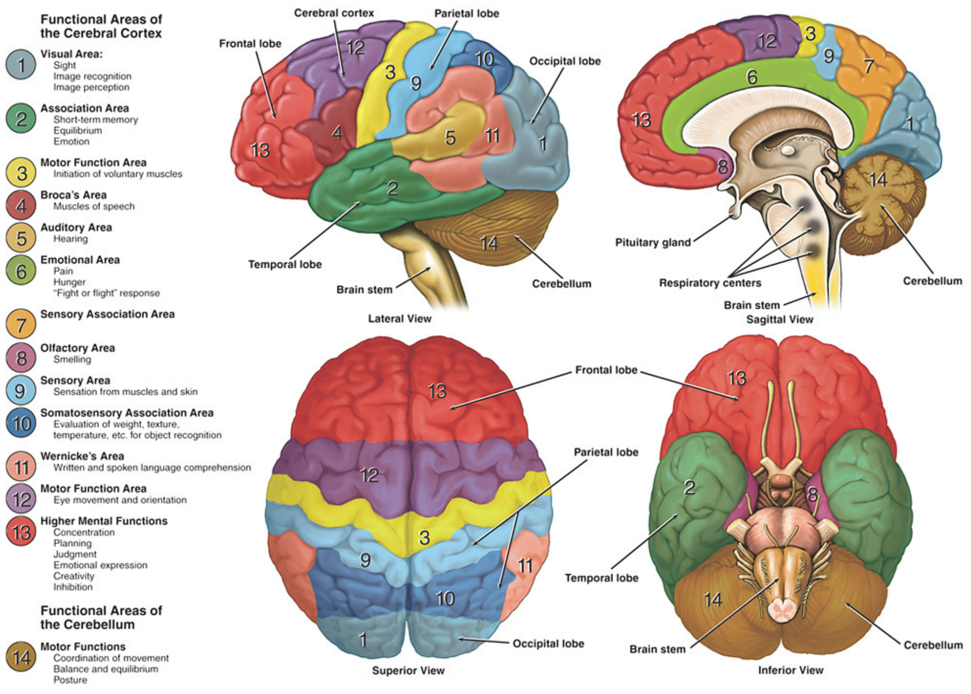

Functional Areas of the Cerebral Cortex
1 Visual Area: Sight, Image recognition, Image perception
2 Association Area: Short-term memory, Equilibrium, Emotion
3 Motor Function Area: Initiation of voluntary muscles
4 Broca's Area: Muscles of speech
5 Auditory Area: Hearing
6 Emotional Area: Pain, Hunger, "Fight or flight" response
7 Sensory Association Area
8 Olfactory Area: Smelling
9 Sensory Area: Sensation from muscles and skin
10 Somatosensory Association Area: Evaluation of weight, texture, temperature, etc. for object recognition
11 Wernicke's Area: Written and spoken language comprehension
12 Motor Function Area: Eye movement and orientation
13 Higher Mental Functions: Concentration, Planning, Judgment, Emotional expression, Creativity, Inhibition
Functional Areas of the Cerebellum
14 Motor Functions: Coordination of movement, Balance and equilibrium, Posture
Cerebral cortex
Parietal lobe
Frontal lobe
Occipital lobe
Temporal lobe
Brain stem
Cerebellum
Lateral View
Pituitary gland
Respiratory centers
Brain stem
Cerebellum
Sagittal View
Frontal lobe
Parietal lobe
Temporal lobe
Occipital lobe
Brain stem
Cerebellum
Superior View
Inferior View

**Figure 2.1:** Overview of brain anatomy. Taken from [12]

Although neurons can be classified into various types based on their shape and function, they all share a common structural framework consisting of three main parts: the cell body, which contains the nucleus and organelles necessary for the neuron's metabolic functions and processes incoming information; the dendrites, which are branch-like extensions that receive signals from other neurons and transmit them to the cell body; and the axon, a long projection that carries electrical impulses away from the cell body to other neurons, muscles, or glands. The axon may be covered by a myelin sheath, enhancing the speed of impulse transmission.

Neurons can be divided into two main categories based on their roles in integrating sensory input and generating appropriate actions. Afferent neurons, also known as sensory neurons, carry sensory information from receptors toward the central nervous system (CNS), allowing the brain to perceive and process external and internal stimuli. Efferent neurons, or motor neurons, transmit motor commands from the CNS to muscles and glands, facilitating movement and physiological responses.

Neurons generate and transmit electrical signals called action potentials (APs). An action potential is a rapid change in the electrical charge across the neuron's membrane that travels along the axon to its terminals. Upon reaching the terminals, neurotransmitters are released to communicate with other neurons or target cells. This process enables the nervous system to efficiently coordinate various physiological functions and responses.

### 2.1.4 The Skeletal Muscle System

Skeletal muscles are critical for movement and are structured to facilitate contraction. Each skeletal muscle is composed of numerous muscle fibers, which are long, cylindrical cells capable of contracting. These fibers are grouped into bundles known as fascicles, and each muscle is enveloped by connective tissue that provides support and protection. Muscle fibers contain myofibrils, the contractile components of the muscle. Myofibrils are organized into sarcomeres, which are the smallest functional units of muscle contraction. The interaction between actin and myosin filaments within the sarcomeres leads to muscle contraction through the sliding filament mechanism.

Control of skeletal muscle contraction is mediated by motor neurons. When a motor neuron sends an action potential, it travels to the muscle fiber at the neuromuscular junction, where the neuron communicates with the muscle. The neurotransmitter acetylcholine is released at this junction, triggering muscle contraction. Subsequently, sensory receptors gather information such as position, resistance, temperature and contact, to then send it to the brain. This feedback loop enables the brain to adjust and fine-tune ongoing movements based on sensory input, ensuring precise control over muscle movements and accurate responses, ranging from simple actions to complex tasks [13].

### 2.1.5 The Genesis of Voluntary Movements

Let us now explore mechanisms and brain regions responsible for managing voluntary movements—which integrate sensory feedback with motor commands to ensure smooth and coordinated actions. This is crucial for our study, in order to understand the central role of the Primary Motor Cortex(M1), that will be the main source of our data. The generation of movement in the brain involves a meticulously coordinated process across several key areas. It begins with the planning and initiation of movement, progresses to execution, and is continuously refined based on real-time sensory feedback.

#### 1. Planning and Initiation of Movement

The process of voluntary movement starts in the premotor cortex and the supplementary motor area (SMA), both of which are situated in the frontal lobe. These

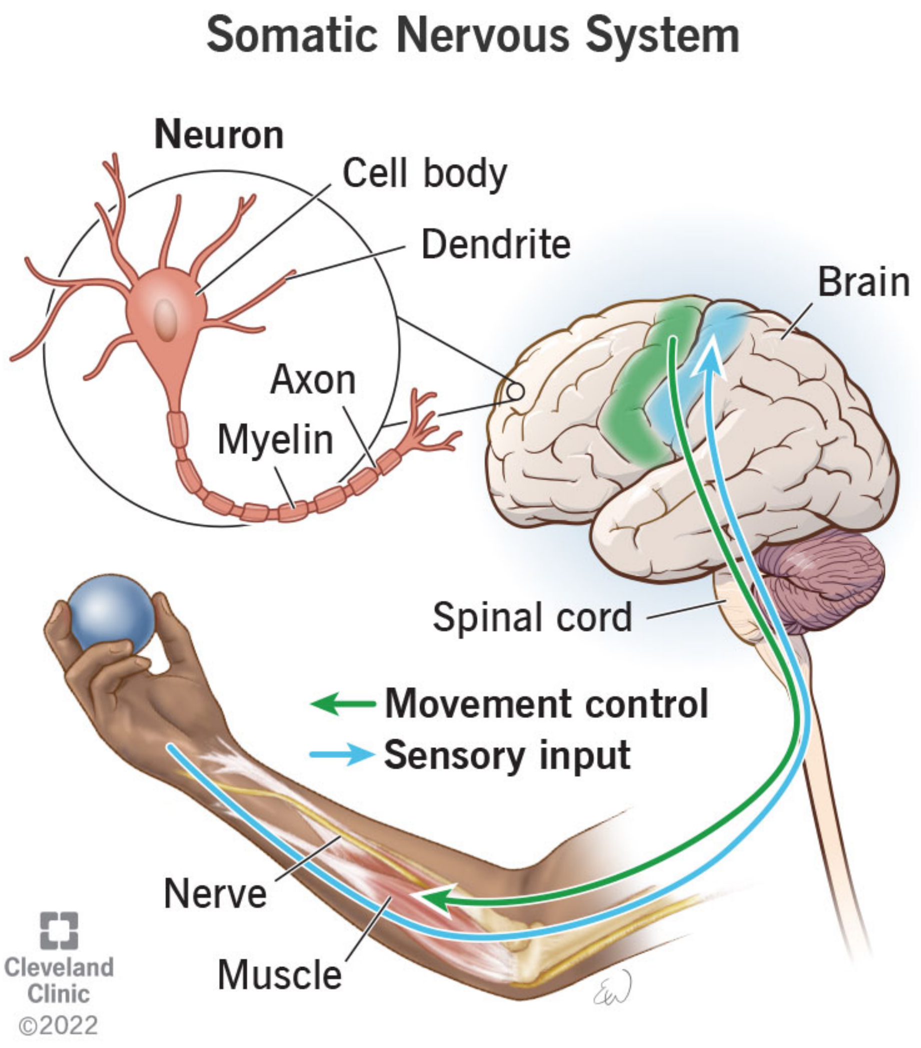


**Figure 2.2:** Simplified representation of the sensorimotor loop. M1 Cortex in green and S1 Cortex in blue. Taken from [14].

areas are involved in the planning and preparation of complex motor actions. The premotor cortex integrates sensory information to create a motor plan, while the SMA contributes to the planning of sequential movements and coordination of movements involving both sides of the body [11, pp. 313–320].

## 2. Execution of Movement

Once a movement plan is formulated, it is transmitted to the primary motor cortex (M1), located in the precentral gyrus of the frontal lobe. The primary motor cortex is responsible for the execution of voluntary movements. It sends motor commands through the corticospinal tract to the spinal cord, where the commands are relayed to the appropriate muscles. The motor cortex is organized somatotopically, meaning that different regions of the cortex correspond to different body parts, creating a motor map of the body.

This map, known as the motor homunculus, is the somatomotor equivalent of the more famous somatosensory homunculus, situated in the Primary Somatosensory Cortex (S1). Both are represented in Figure 2.3. The motor homunculus reflects the relative cortical area devoted to controlling each body part, with larger areas allocated to parts requiring finer motor control, such as the hands and face. This organization enables precise movement control and exhibits plasticity, adapting to changes in motor demand or injury [11, pp. 337–345].

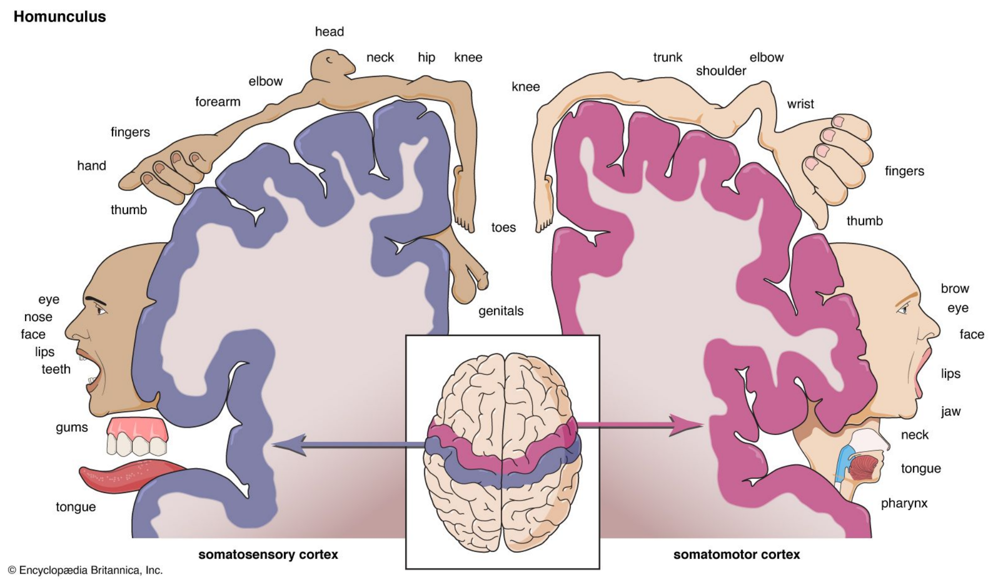


**Figure 2.3:** The two maps present in our brain: the sensory and motor homunculus. Taken from [15]

## 3. Integration of Sensory Feedback

During movement execution, real-time sensory feedback is crucial for fine-tuning and adjusting motor commands. The somatosensory cortex, located in the post-central gyrus of the parietal lobe, processes sensory information related to touch, proprioception, and pressure from various body parts. Sensory signals are sent to the somatosensory cortex, where they are integrated to provide continuous feedback about the movement. This feedback helps in adjusting motor commands to correct any errors and refine the movement [11, pp. 265–268].

**4. Coordination and Refinement**

The cerebellum and basal ganglia are essential for the coordination and refinement of movements. The cerebellum receives input from the motor cortex and the sensory systems, and it is responsible for fine-tuning motor commands to ensure smooth and accurate execution. The basal ganglia, a group of nuclei deep within the brain, helps in regulating the initiation and amplitude of movement by modulating the activity of the motor cortex [11, pp. 297–311].

**5. Continuous Adjustment**

Throughout the execution of a movement, the brainstem assists in regulating basic motor functions and maintaining posture. It integrates information from the cerebellum and sensory systems to adjust muscle tone and stabilize movements [11, pp. 297–311].

In summary, the production of voluntary movements involves a complex interaction between several brain regions. The premotor cortex and supplementary motor area plan and prepare movements, the primary motor cortex executes them, and the somatosensory cortex provides necessary feedback for ongoing adjustments. The cerebellum and basal ganglia further refine and coordinate these movements, ensuring precise and adaptive motor control.

## 2.2 Brain Signal Acquisition Methods for BMIs

Acquiring brain signals is a fundamental aspect of brain-machine interfaces (BMIs) and neuroprosthetics, as it allows for the direct interpretation of neural activity. Various methods exist, each with its own set of advantages and limitations. These methods are typically categorized based on their invasiveness and the nature of the signals they capture.

### 2.2.1 Invasive Methods

Invasive methods involve surgical intervention to place electrodes either on the surface or within the brain, enabling the acquisition of high-fidelity neural signals. Such implants are called intracranial electrodes provide the most direct access to brain activity, capturing high-fidelity signals, including single-neuron action potentials and local field potentials (LFPs) [4]. Surgically implanted, these electrodes offer precise neural activity localization, making them valuable for decoding complex movements such as hand trajectories, individual finger movements, and tactile stimuli within the motor cortex. [6]

#### Electrocorticography (ECoG)

ECoG electrodes are placed on the cortex's surface, requiring a craniotomy but not penetrating the brain tissue. This method balances signal fidelity and invasiveness, offering better spatial resolution than EEG while minimizing risks associated with deeper implantation. ECoG is effective in decoding various movements, ranging from discrete hand gestures to continuous cursor control [6, 4].

#### Stereo-electroencephalography (SEEG) and Deep Brain Stimulation (DBS)

SEEG and DBS electrodes are implanted in deeper brain structures such as the subthalamic nucleus (STN) or ventrolateral thalamus (VL). Primarily designed for epilepsy monitoring and Parkinson's disease treatment, these electrodes demonstrate potential for movement decoding. Researchers utilize SEEG and DBS signals to predict movement onset, identify hand movements, and decode continuous kinetic information related to grasping [4, 6].

### 2.2.2 Non-Invasive Methods

Non-invasive methods, in contrast, do not require surgical intervention for electrode placement. These methods are generally safer but often come with limitations in signal resolution and susceptibility to noise.

#### Electroencephalography (EEG)

EEG is the most prevalent non-invasive technique, involving the placement of electrodes on the scalp to measure electrical activity from neuronal populations.[16, 5, 4], While safe and user-friendly, EEG has limited spatial resolution compared to invasive methods and is vulnerable to noise from muscle activity and other artifacts [5, 16, 4]. Despite these drawbacks, EEG is effective in decoding movements such as cursor control, wheelchair navigation, and prosthetic limb control.[5, 16, 4]

#### Magnetoencephalography (MEG)

MEG measures the magnetic fields generated by electrical currents in active neurons. Offering enhanced spatial resolution compared to EEG, MEG is less prone to signal distortion by the skull and scalp. However, MEG requires specialized equipment and a shielded environment to minimize magnetic interference [5, 4].

#### Functional Near-Infrared Spectroscopy (fNIRS)

fNIRS is a non-invasive optical imaging technique that measures blood oxygenation changes in the brain, reflecting neuronal activity. Although safe, portable, and less sensitive to motion artifacts than fMRI, fNIRS has limited temporal resolution. Nonetheless, fNIRS demonstrates potential in decoding movement-related brain activity [5, 4].

#### Functional Magnetic Resonance Imaging (fMRI)

fMRI is another method used to measure brain hemodynamic responses, utilizing blood-oxygen-level-dependent (BOLD) activity measured with an MRI scanner. While fMRI boasts superior spatial resolution, it suffers from low temporal resolution, with a significant lag between neuronal activity and the BOLD response [4].

### 2.2.3 Advantages of ECoG

In the context of brain-machine interfaces, electrocorticography (ECoG) is often favored due to its balance between invasiveness and signal fidelity. While fully invasive techniques offer higher resolution, they come with increased risks of tissue damage and signal degradation over time. Conversely, non-invasive methods like electroencephalography (EEG) are safer but typically provide lower resolution and less reliable signals.

ECoG serves as a middle ground by delivering high-quality signals for decoding two-dimensional hand kinematics with relatively lower risks, making it a suitable option for brain-machine interface (BMI) and brain-computer interface (BCI) applications. ECoG electrodes are positioned directly on the surface of the brain, capturing neural activity with greater spatial resolution than non-invasive EEG, which facilitates more precise movement decoding [4]. Additionally, ECoG grids can cover a larger cortical area than penetrating microelectrode arrays, providing access to a broader range of motor and sensory information pertinent to two-dimensional kinematics [4].

Moreover, ECoG offers a favorable signal-to-noise ratio, being less susceptible to artifacts such as muscle activity, which often plague EEG recordings [4]. This advantage is further underscored by a study demonstrating that a subject could control a BMI using facial electromyography (EMG), illustrating EEG's vulnerability to such artifacts [4, 17].

Finally, multiple sources emphasize the long-term stability of ECoG compared to intracortical electrodes [2, 18]. This stability arises from ECoG electrodes being placed on the brain's surface, which minimizes tissue damage and inflammatory responses that can lead to signal degradation over time [4, 1, 19]. Chronic ECoG recordings have been shown to last for many years in both animals and humans [4].

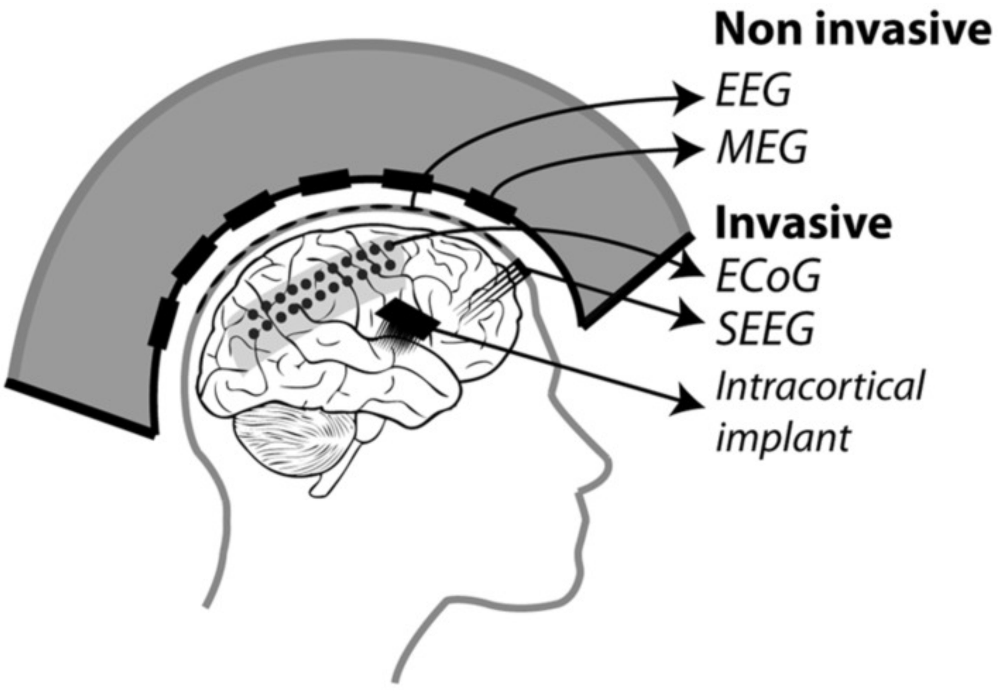


**Figure 2.4:** Overview of brain signal recording methods. Taken from [20]

## 2.3 Signal Cleaning and Processing Techniques (ECoG)

Signal processing techniques play a crucial role in analyzing neural data, particularly for applications like Brain-Machine Interfaces (BMIs). These techniques are essential for interpreting the complex and often noisy signals recorded from the brain, especially in the context of electrocorticography (ECoG) signals.

### 2.3.1 Data Acquisition and Initial Cleaning

Neural ECoG signals, like in our case, are commonly acquired using intracortical microelectrode arrays such as the Utah array. These arrays consist of multiple electrodes that can record the activity of both single neurons, referred to as single-unit activity (SUA), and groups of neurons, known as multi-unit activity (MUA). The recorded signals are usually sampled at high frequencies (e.g., 24.4 kHz or 30 kHz) to capture the fast temporal dynamics of neural activity. To preserve the intricate details of these signals, they are digitized with high resolution, typically 16-bit or more [2, 3].

Immediately after acquisition, the neural signals undergo several preprocessing steps to improve their quality. Pre-amplification is applied to strengthen the signals, and an anti-aliasing low-pass filters may be applied to remove high-frequency noise above the Nyquist frequency. This process is critical for enhancing the signal-to-noise ratio, thereby facilitating more accurate downstream analysis [21, 2, 3].

Moreover, as better explained below, different kind of referencing scheme may be adopted to further enhance signal quality.

### 2.3.2 Noise Reduction, Referencing Schemes, and Our Approach

In neural recordings, reducing noise and removing artifacts are crucial for obtaining high-quality data, especially given the wide array of noise sources, including power line interference (50 or 60 Hz and harmonics), electronic artifacts, and biological noise.

The dataset we use for our study already applies unipolar referencing during the data recording phase. In unipolar referencing, each electrode's potential is measured relative to a distant reference electrode, often located at an electrically neutral site. This distant positioning helps reduce interference from local neural activity. However, because the reference is external and far from the recorded brain region, unipolar referencing is more susceptible to common-mode noise. Common-mode noise originates from external electromagnetic sources or from shared activity across the brain regions, affecting multiple electrodes simultaneously. As a result, unipolar referencing cannot easily differentiate between true neural signals and widespread noise, potentially compromising signal clarity [22].

To improve the signal-to-noise ratio and better isolate neural activity in our analysis, we will implement Common Average Referencing (CAR). In CAR, the average signal across all electrodes is subtracted from each individual electrode's signal. This technique reduces noise that is common to all electrodes while preserving the distinct neural activity at each recording site. The method can be expressed mathematically as:

$$V_{\text{CAR},i}(t) = V_i(t) - \frac{1}{N}\sum_{j=1}^{N} V_j(t) \tag{2.1}$$

where $V_{\text{CAR},i}(t)$ is the CAR-referenced signal for electrode $i$, $V_i(\text{t})$ is the original signal from electrode $i$, and $N$ is the total number of electrodes, this value is calculated at each time step $t$. This approach effectively eliminates common-mode noise by leveraging the assumption that noise is distributed evenly across all electrodes, thus enhancing the neural signals of interest [23, 21, 3].

CAR offers robust noise suppression, making it a preferred method for neural signal analysis. While bipolar referencing is another effective strategy for reducing local noise by measuring the potential difference between neighboring electrodes, CAR provides a more global and simple solution by addressing noise common to the entire electrode array. For this reason, CAR is particularly well-suited to our study's focus on neural decoding, where capturing high-quality signals while keeping it as simple as possible is critical [2].

### 2.3.3 Spike Detection and Spike Sorting (MUA and SUA)

Later on, we can proceed with the real preprocessing: spike detection and sorting are common steps when dealing with ECoG signals and BMIs. These methods are used to detect the activity of individual neurons from electrodes recording population spiking activity.

Spikes are brief, high-frequency events that reflect the output of neurons. To detect these spikes, the raw neural signal is typically band-pass filtered to isolate the frequency band associated with spikes (usually above 300 Hz, or between 500 and 5000 Hz) [2, 3, 24]. Then events in that range of frequency that cross a predefined threshold are identified as spikes (Threshold crossing TC). This threshold is set based on the noise level of the recording and can be determined either manually or using automated algorithms [25, 3, 19].

Following detection, spike sorting is performed to distinguish the detected spikes based on their waveform. From a single channel, detected spikes are grouped in different units. This step is crucial to differentiate between spikes from different neurons on the same electrode. It provides insights into how each neuron, or usually a little sub-population of neurons, contributes to brain activity. Various algorithms are used for this purpose. [2, 3, 19, 24].

To reduce memory usage and computational costs, we will not consider each spike event, but we will perform a firing rate estimation.

### 2.3.4 Firing Rate Estimation (TCR)

Firing rate estimation is a key technique used to summarize spiking activity over time, converting discrete neural events into a simpler representation that makes neural data more manageable for processing. This approach reduces computational complexity by focusing on overall neural activity patterns rather than individual spike events, making it particularly valuable in creating a lightweight decoder for brain-machine interfaces.

We experimented with two widely used methods for firing rate estimation: binning and kernel smoothing. Binning is a straightforward technique where spikes are counted within fixed-duration intervals, or bins, and the firing rate is expressed as the total number of spikes within each bin. It aims to preserve the temporal resolution of neural activity while avoiding the need to process each spike individually. This method is simple to implement and computationally efficient, but its effectiveness depends on choosing the right bin size. A large bin may miss rapid changes in neural dynamics, while a small bin can introduce noise and variability [19].

On the other hand, kernel smoothing produces a continuous estimate of the firing rate by convolving the spike train with a smoothing kernel, such as the Gaussian kernel. The width of the Gaussian kernel, controlled by its standard deviation, determines how much smoothing is applied. A wider kernel smooths out noise but may obscure finer details of neural activity, while a narrower kernel retains more temporal precision at the expense of potentially allowing more noise into the signal [22, 19].

When you combine binning with spike detection you have Threshold Crossing Rate (TCR). In short, it provides a quick way to estimate the firing rate of neurons by counting how often the signal crosses a set threshold, making it a useful tool in detecting spikes from neural data. However, it has limitations in accuracy, especially in noisy recordings or for smaller, less obvious spikes.

In our study, we tested both techniques, as background research, and found that they provided similar performance in terms of decoding accuracy. We chose to adopt TCR because of its computational simplicity and reduced energy usage. It allowed us to achieve a more lightweight and efficient implementation without sacrificing significant accuracy. We will discuss the technical details and the parameters used in this implementation in the chapter 4.

### 2.3.5 Most Common Processing Techniques

For completeness, we outline some of the main preprocessing techniques commonly discussed in the literature. These techniques are essential for enhancing the quality of neural recordings and preparing the data for further analysis. In our study, we will focus on three primary methods: Spiking Band Power (SBP), Multi-Unit Activity (MUA), and Single-Unit Activity (SUA). By utilizing and adapting these three methods, we aim to enhance the accuracy and reliability of our neural signal analyses, tailoring the preprocessing techniques to best fit our experimental goals.

**Entire Spiking Activity (ESA)** : It captures the overall spiking activity without relying on threshold-based detection. ESA is obtained by high-pass filtering the raw signal, full-wave rectifying it, and then applying a low-pass filter. This method is particularly advantageous in chronic recordings with low signal-to-noise ratios, as it is less sensitive to noise and avoids biases associated with threshold-based [3].

**Multi-Unit Activity (MUA)** : It represents the combined spiking activity of a group of neurons recorded on a single electrode. MUA is usually obtained by band-pass filtering the raw signal to isolate the spiking frequency band (e.g., 500 Hz to 5 kHz) and then by computing spike detection. Although it provides a coarser measure of neural activity compared to SUA, MUA still offers valuable information about population-level dynamics [2].

**Single-Unit Activity (SUA)** : It reflects the spiking activity of single neurons. SUA is obtained by first detecting and then sorting spikes to identify those originating from the same neuron. This high level of detail is crucial for studying neural coding and network dynamics [2].

**Local Field Potential (LFP)** : It represents the low-frequency component of the extracellular signal, reflecting the summed synaptic and subthreshold activity of a population of neurons near the electrode. LFPs are extracted by applying a low-pass filter to the raw signal, typically with a cutoff frequency around 300 Hz. LFPs contain valuable information about both the input to a neural population and the local processing within that population [24].

**Spiking Band Power (SBP)** : It can be seen as a variant of ESA. It measures the power within the high-frequency band of the neural signal, typically between 300 Hz and 1 kHz or higher, primarily reflecting the spiking activity of neurons. SBP is a continuous signal, and can be used as a simpler alternative or complement to traditional spike sorting, especially in applications where identifying individual units is challenging or unnecessary. This method is very energy and memory-efficient, and it is particularly effective in decoding tasks even at low signal-to-noise ratios, highlighting its potential for long-term implantable devices [26].

## 2.4 Spiking Band Power Overview

Spiking Band Power (SBP) has recently gained attention in neuroengineering, especially regarding its application in Brain-Machine Interfaces (BMIs). The novelty of our work also involves examining the long-term stability of this preprocessing method. For this reason, this section delves into the definition of SBP, its

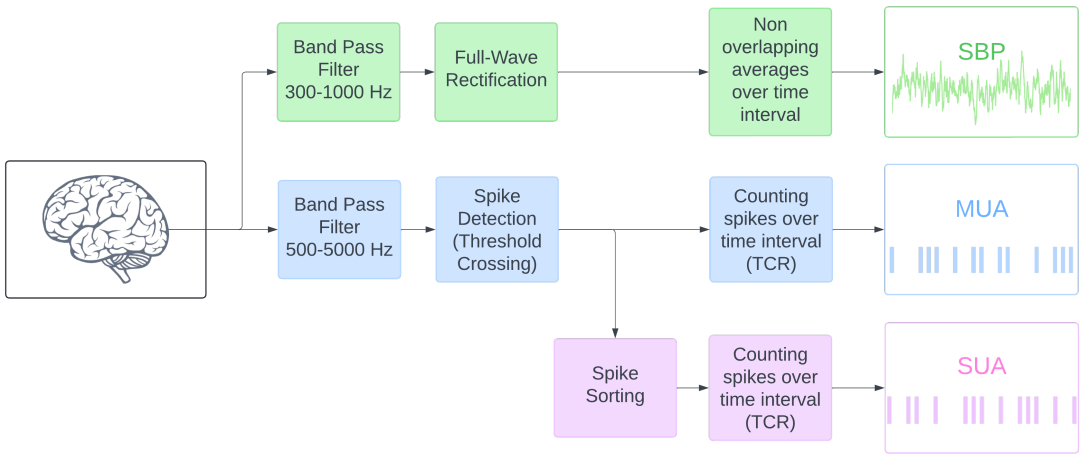

Band Pass Filter 300-1000 Hz
Full-Wave Rectification
Non overlapping averages over time interval
SBP
Band Pass Filter 500-5000 Hz
Spike Detection (Threshold Crossing)
Counting spikes over time interval (TCR)
MUA
Spike Sorting
Counting spikes over time interval (TCR)
SUA

**Figure 2.5:** The main preprocessing steps to obtain SBP, MUA, and SUA.

performance advantages, power efficiency, methodology for extraction, and its applications in various tasks.

### 2.4.1 Understanding Spiking Band Power

SBP quantifies the power concentrated within a specific high-frequency range of recorded neural signals, typically between 300 and 1,000 Hz or higher [26]. This measure is crucial for capturing the energy predominantly associated with neuronal spiking activity. Conceptually, the neural signal can be envisioned as a spectrum of frequencies; SBP emphasizes the higher end, where action potentials, or spikes, are most prevalent. Unlike traditional spike-sorting methods that depend on identifying individual spikes above a specific voltage threshold, SBP quantifies the overall power within the spiking band. This approach is particularly advantageous as it enables the detection and analysis of signals from neurons with weaker amplitudes that might be missed by threshold-based methods.

### 2.4.2 Performance Advantages and Power Efficiency

SBP demonstrates superior performance compared to traditional spike-based decoding methods, such as MUA and SUA, especially in predicting complex motor movements. Its narrower bandwidth does not compromise its excellent spatial specificity, effectively reflecting the activity of high-amplitude neurons situated near the electrode. This characteristic is crucial for accurately capturing neural activity, even at low signal-to-noise ratios (SNRs), allowing SBP to detect neuronal signals that may elude conventional spike sorting techniques [26]. Moreover, the reduced bandwidth requirement of SBP leads to significant power savings for BMI devices. By concentrating on a narrower frequency band, SBP enables lower sampling rates, which directly decreases power consumption in the analog front-end. Additionally, SBP can eliminate the need for complex spike detection and sorting circuits, further enhancing overall power efficiency [26]. This efficiency is particularly advantageous for real-time applications in BMIs, where battery life and computational resources are often constrained.

### 2.4.3 Methodology for SBP Extraction

The extraction of SBP follows a systematic process, primarily based on methodologies outlined in existing literature, with slight modifications planned to align with the specific needs of our study. Practical details of these adjustments will be detailed in Chapter 4.

The extraction process begins with band-pass filtering the raw neural signal to isolate the frequency range associated with spiking activity, typically 300-1,000 Hz. This step effectively removes irrelevant frequency components, focusing the analysis on the spiking band [26, 27]. Next, the filtered signal undergoes full-wave rectification, where both positive and negative phases of the spikes are captured by taking the absolute value [27]. This is the magnitude extraction. After rectification, the signal is smoothed and downsampled using various techniques, including static binning with non-overlapping averages, moving average filters, or Gaussian filters. These methods enhance the signal by smoothing it and highlighting the overall envelope of spiking activity, thus facilitating more accurate analyses of neuronal behavior [27, 26].

### 2.4.4 Applications of SBP

Recent studies illustrate the effectiveness of SBP in various BMI tasks. For instance, SBP has shown performance comparable to or superior to TCR methods in decoding one-dimensional finger movements in both open-loop and closed-loop settings [26]. It has excelled in two-dimensional cursor control tasks, highlighting its potential for more complex control schemes [26]. Additionally, SBP has been successfully utilized as input for SNN decoders, achieving high accuracy in finger velocity prediction while maintaining computational efficiency [27].

### 2.4.5 Conclusions and Future Directions

While SBP shows considerable promise, it is important to note that most research is still in its preliminary stages, necessitating further investigation to fully understand its limitations and optimize its application across various BMIs [26]. We will examine this aspect in our reseach. Additionally, the optimal bandwidth for SBP may vary depending on factors such as brain region, recording electrodes, and the specific task being decoded [27, 26]. In summary, Spiking Band Power (SBP) emerges as a powerful and promising neural feature for BMI applications. It offers a unique blend of high decoding accuracy, spatial specificity, and power efficiency, positioning it as a valuable tool for the development of advanced and clinically viable neural interfaces.

## 2.5 Spiking Neural Networks

Spiking Neural Networks (SNNs) represent a neuromorphic computing architecture inspired by the structure and function of the brain [28]. Unlike conventional artificial neural networks (ANNs), which rely on continuous activation functions and synchronous neurons, SNNs process information using asynchronous discrete spikes, mimicking the way biological neurons communicate. This approach makes them particularly suitable for brain-inspired computing. Additionally, the binary and sparse nature of spikes leads to more efficient computations, offering significant improvements in terms of power consumption and resource usage [29, 28]. Among others, two studies by Carpegna et al. [30, 31] discuss and explore efficiency in hardware implementation, while the recent results by Padovano et al.[32] show how optimisation can further enhance on-chip efficiency while maintaining high prediction quality.

In SNNs there is a loss in encoding capacity due to the binary nature of spikes. However, this limitation can be mitigated by leveraging the temporal dynamics neurons, which makes SNNs particularly effective for handling time series data. Essentially, information can be encoded not just in the presence or absence of a spike, but also in the timing of spikes [33]. The efficiency of SNNs can be fully realized through specialized hardware optimised to simulate such structure. This is an active area of research with platforms like Loihi from Intel and TrueNorth from IBM aiming to push the boundaries of low-power SNN processing [29]. Unfortunately, while this hardware aspect is of great interest, it falls outside the scope of our exploration due to time constraints.

At a structural level, SNNs closely resemble traditional ANNs, with the primary distinction being the behavior of individual neurons. Keeping this in mind, we will first outline the architecture of the most basic neural network, the fully connected neural network (FCNN), and then explore how it evolves within this new neuromorphic paradigm. As a matter of fact, the temporal dynamics of a SNN follows the general paradigm of recurrent neural networks (RNN) [33]. It is important to mention that a lot of different solution have been proposed for ANNs and then adapted to SNNs, like the more modern Transformers [34]. We could explore these topics in much greater detail, and we would love, but given the limitations of time and space, we'll focus on the essentials and relevant information for this thesis, aiming to be exhaustive, clear, and not too boring.

### 2.5.1 Fully Connected Neural Networks

For details about this architecture refer to [35]. A fully connected neural network (FCNN), also known as a dense neural network, is a fundamental type of artificial neural network where each neuron in one layer is connected to every neuron in the following layer. This architecture allows for a comprehensive interaction between the neurons across different layers, enabling the network to learn complex patterns from the data.

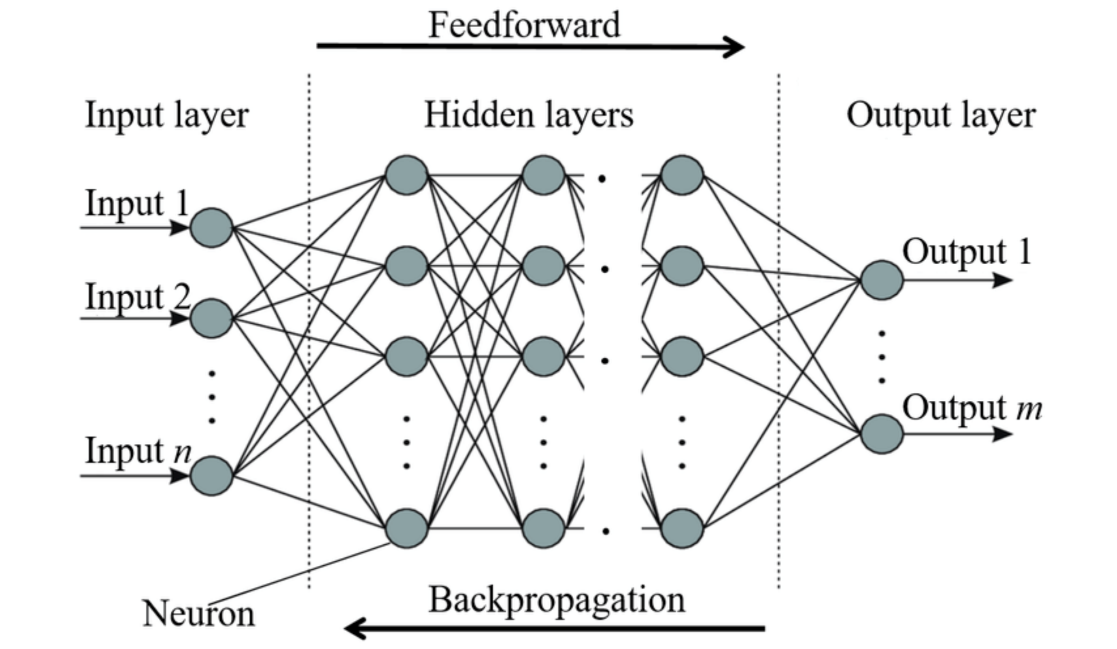


**Figure 2.6:** Structure and main components of a FCNN. Figure taken from [36].

#### Neurons, Layers, Weights, and Biases

Neurons serve as the fundamental computational units in neural networks, emulating the function of biological neurons. Each neuron processes input from other neurons or raw data by computing a weighted sum, adding a bias, and applying an activation function to generate an output, which is then transmitted to the next layer.

The basic architecture of a neural network consists of several layers. The input layer receives the raw data, where each neuron corresponds to a feature of the input;

for example, a 28x28 pixel grayscale image would have 784 neurons. Hidden layers follow, where the majority of computations occur. Neurons in these layers are fully interconnected, and the weights of these connections are adjusted during training to improve data fitting. Finally, the output layer generates the final predictions, which could involve classification or regression tasks. For a classification problem with 10 classes, the output layer would contain 10 neurons, each representing a class, often utilizing functions like softmax to produce interpretable probabilities.

The strength of the connections between neurons is governed by weights, which are learned during training, along with biases that allow for the activation function to be shifted. This combination enhances the network's capacity to accurately model the data by minimizing prediction errors through iterative updates during the training process.

### 2.5.2 Recurrent Neural Networks

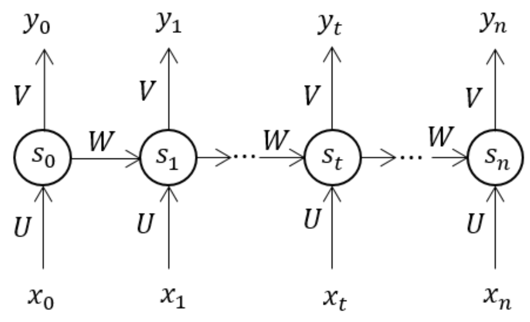


**Figure 2.7:** An RNN unfolded, with $S_0, S_1, ...$ being the internal states of the net at each time step. Taken from [37]

.

While FCNNs excel in handling static data, they fall short when it comes to sequences or time-dependent information. This is where Recurrent Neural Networks (RNNs) come into play. RNNs are specifically designed to process sequential data by incorporating temporal dynamics into their architecture. Unlike FCNNs, RNNs maintain a memory of previous inputs through internal state representations, allowing them to capture patterns across time steps. In Figure 2.7 we can see a graphical representation of an RNN unfolded in time, highlighting the role of previous internal stases in current predictions.

The key feature of RNNs is their ability to loop back connections, enabling information to persist in the network. This recurrent structure allows RNNs to learn from both current and past inputs, making them particularly effective for tasks such as natural language processing, speech recognition, and time series forecasting. By leveraging this memory, RNNs can model temporal dependencies and make predictions based on sequences of varying lengths.

In summary, RNNs enhance the capabilities of neural networks by addressing the limitations of FCNNs when it comes to sequential data, providing a powerful tool for a wide range of applications that require an understanding of temporal relationships. More details can be found in the paper by Laith Alzubaidi et al.[35]

### 2.5.3 Common Spiking Neuron Models

The differences between Artificial Neural Networks (ANNs) and Spiking Neural Networks (SNNs) primarily stem from the structure of their fundamental units: the neurons. For this reason, we will briefly mention the main models used in the spiking neural network domain, here a more comprehensive review [29].

The Hodgkin-Huxley (HH) Model is celebrated for its biological realism, a complex simulation of the dynamics of ion channels that govern neuronal activity. However, its high computational demands can make it impractical for large-scale simulations and deep learning tasks. In contrast, the Izhikevich (IZH) Model offers a balance between biological fidelity and computational efficiency, successfully reproducing a range of neuronal firing patterns, though it is still more demanding than simpler models.

The Leaky Integrate-and-Fire (LIF) Model stands out for its computational efficiency, integrating input currents over time while allowing charge to leak at a defined rate. When the membrane potential exceeds a specific threshold, the neuron fires a spike and resets. The Adaptive Exponential (AdEx) Model builds on the LIF framework by incorporating an adaptation mechanism that makes firing rates dependent on past activity, thus enhancing biological realism without sacrificing computational efficiency. Other models, such as the Integrate-and-Fire (IF) and Non-linear Integrate-and-Fire (NLIF) models, also exist, each with their own trade-offs between biological accuracy and computational complexity.

### 2.5.4 The Appeal of LIF Neurons

The following concepts about LIF neurons and formulas are taken from previous literature [28, 29, 34]. The Leaky Integrate-and-Fire (LIF) neuron model has gained popularity, particularly in deep learning applications, due to its computational simplicity and efficiency. This makes it well-suited for large-scale simulations and deep spiking neural networks (SNNs). The straightforward dynamics of the LIF model align well with gradient-based learning algorithms such as Spatio-Temporal Backpropagation (STBP), enhancing the effectiveness of the training process. While it does not capture the full complexity of biological neurons as models like Hodgkin-Huxley (HH) or Izhikevich (IZH) do, the LIF neuron offers a practical balance between biological realism and computational feasibility. Additionally, its simplicity makes it ideal for implementation on neuromorphic hardware, which often prioritizes energy efficiency.

Ultimately, the choice of neuron model should align with the specific objectives of the application. For scenarios where biological fidelity is crucial and computational cost is less of a concern, HH or IZH models may be preferable. However, for applications that emphasize computational efficiency, scalability, and compatibility with deep learning techniques and neuromorphic platforms, LIF neurons emerge as the most convenient option.

### 2.5.5 The LIF Neuron in Detail

The Leaky Integrate-and-Fire (LIF) neuron provides a simplified but effective representation of biological neurons, balancing computational efficiency with biological plausibility. The functioning of the LIF neuron can be understood through the accumulation of synaptic currents over time. Inputs, representing the electrical signals from other neurons, are integrated within the membrane potential of the neuron. This process is akin to charging a capacitor, where the neuron accumulates the input signals it receives, which are scaled by their respective synaptic weights.

A key feature of the LIF neuron is its leakage mechanism, where the membrane potential gradually decays over time in the absence of sustained inputs. This is a distinctive property that sets it apart from models without decay, and is controlled by a parameter, often represented as $\tau$ or $\beta$, that determines how quickly the neuron "forgets" past inputs. The decay process ensures that the neuron does not accumulate inputs indefinitely and resets itself in the absence of stimulation.

Once the accumulated membrane potential crosses a defined threshold, $V_{th}$, the neuron fires a spike. This firing event transmits a signal to other neurons in subsequent layers, acting as a communication event. After the neuron spikes, the membrane potential resets, which typically involves subtracting the threshold value.

Mathematically, the membrane potential of the LIF neuron in the continuous-time LIF model behaves following this differential equation:

$$\tau_m \frac{du(t)}{dt} = -(u(t) - u_{rest}) + RI(t) \tag{2.2}$$

$$\text{if } u(t) \geq V_{th} \rightarrow \begin{cases} O(t) = 1 \\ u(t+dt) = u(t) - V_{th} \end{cases} \tag{2.3}$$

Where $\tau_m = RC$, $u_{rest} = 0$ and $I(t)$ is the input signal. When the membrane potential surpasses the fixed $V_{th}$, then a spike is emitted and $u(t)$ is reset by subtraction. In Figure 2.8 we can see an example that highlights these dynamics.

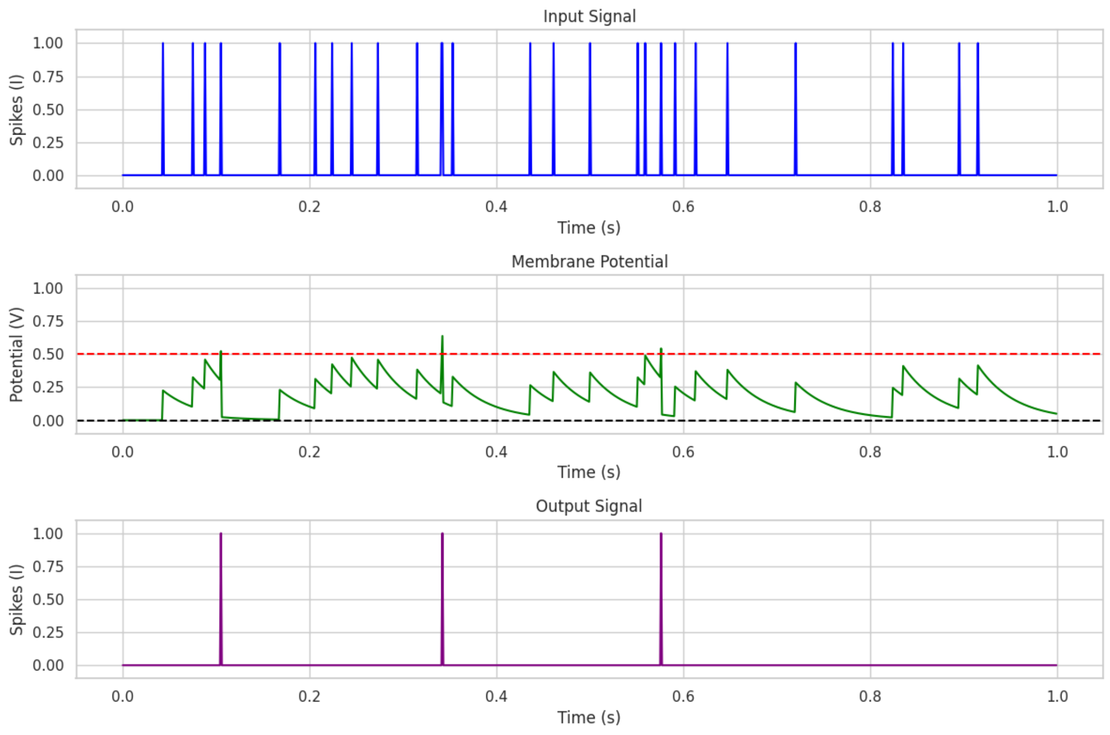


**Figure 2.8:** LIF-neuron response to input spikes and behaviour of the membrane leaking dynamics

While the discrete approximation, implemented in our digital model is represented in the equation below:

$$u(t) = \tau(u(t-1) - s(t-1)V_{th}) + I(t) \tag{2.4}$$

where $u(t)$ represents the membrane potential at time $t$, $\tau$ is the decay factor governing the rate of leakage, $s(t-1)$ is the output spike from the previous time step, and $I(t)$ is the total synaptic input current at time $t$. This input current, $I(t)$, can be described as the sum of the weighted input spikes from connected neurons:

$$I(t) = \sum_i w_i \cdot s_i(t) + b \tag{2.5}$$

Here, $w_i$ denotes the synaptic weight for the $i$-th input, $s_i(t)$ is the spike received from the $i$-th neuron, and $b$ represents the bias term associated with the neuron.

The output spike $s(t)$, which represents whether the neuron fires or not, is a binary decision:

$$s(t) = \begin{cases} 1 & \text{if } u(t) \geq V_{th} \\ 0 & \text{otherwise} \end{cases}$$

Thus, the LIF neuron behaves in a time-dependent manner, accumulating input, leaking potential, and firing once the threshold is crossed, then resetting to repeat the process. This dynamic system allows for the modeling of time-sensitive neuronal behavior, making the LIF neuron particularly suitable for spiking neural networks (SNNs) and applications that require efficient temporal computation.

### LIF Neurons vs. Traditional Perceptrons

While both LIF neurons and perceptrons (traditional neurons in ANNS) serve as foundational units in neural networks, they operate under distinct mechanisms:

| **Feature** | **LIF Neuron** | **Perceptron** |
|---|---|---|
| Signal Type | Spikes (discrete events in time) | Continuous values |
| Time Dynamics | Inherently time-dependent; membrane potential evolves over time, integrating inputs and leaking charge | Typically time-independent; output depends solely on current inputs |
| Output Representation | Spike rate (frequency of spikes) or directly using membrane potential for continuous values | Continuous value obtained by applying an activation function (e.g., sigmoid, ReLU, Softmax) |
| Learning | Spike-Timing-Dependent Plasticity (STDP) or gradient-based methods like STBP | Gradient-based methods, primarily backpropagation |
| Biological Plausibility | More biologically realistic due to spike-based communication and temporal dynamics | Less biologically realistic; primarily an abstraction of weighted summation and synchronous computations |

**Table 2.1:** Comparison between LIF neuron and perceptron

In essence, LIF neurons capture the temporal aspects of neural processing, making them suitable for tasks involving time-series data and event-based applications. Their spike-based nature also aligns well with energy-efficient neuromorphic hardware. In contrast, perceptrons excel in tasks where temporal dynamics are less critical, and continuous input-output mappings suffice.

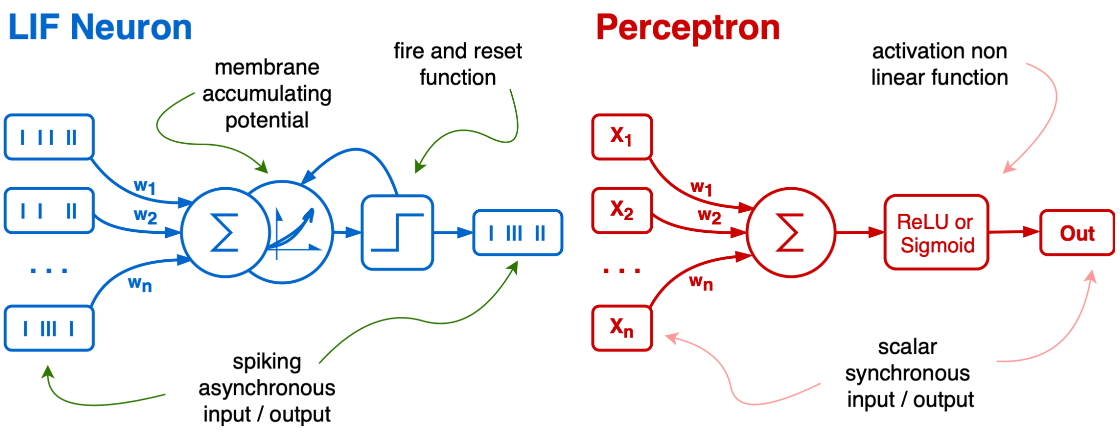

LIF Neuron
membrane accumulating potential
fire and reset function
w1
w2
wn
spiking asynchronous input / output
Perceptron
activation non linear function
X1
X2
Xn
w1
w2
wn
ReLU or Sigmoid
Out
scalar synchronous input / output

**Figure 2.9:** Comparison between LIF neuron and perceptron.

## 2.5.6 Backpropagation for FCNNs

We believe that providing a clear explanation of how the training process works is very important for our work. As a matter of fact, it's crucial to discuss how it is adapted for Spiking Neural Networks (SNNs). Backpropagation is fundamental for training Artificial Neural Networks (ANNs); it's a mathematical process that enables the network to adjust its weights and biases, minimizing error and allowing for gradual learning that extracts meaningful insights from raw data. Without backpropagation, neural networks as we know them would not exist. To introduce Backpropagation we refer to this summary paper [38].

The training process utilizes the chain rule of calculus to propagate errors backward through the network, calculating the gradients (derivatives) of the loss function with respect to each weight and bias. These gradients are then employed to update the parameters. The process can be broken down into four main steps: the forward pass, loss calculation, backward pass (gradient computation), and weight updates. Let us explore each step along with the relevant mathematical concepts.

### 1. Forward Pass

During the forward pass, the network computes the output by passing the input through each layer, using the current weights and biases. For a single layer, the output of neuron $j$ in layer $l$ is given by:

$$z_j^{(l)} = \sum_i w_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)}$$

where: - $w_{ji}^{(l)}$ is the weight connecting neuron $i$ from the previous layer $l-1$ to neuron $j$ in the current layer $l$, - $a_i^{(l-1)}$ is the activation of neuron $i$ from the previous layer, - $b_j^{(l)}$ is the bias of neuron $j$ in the current layer.

The weighted sum $z_j^{(l)}$ is passed through an activation function $\sigma$ (e.g., ReLU, sigmoid, etc.) to produce the output or activation of neuron $j$:

$$a_j^{(l)} = \sigma(z_j^{(l)})$$

This process is repeated across all layers until the output layer, where the network produces a final prediction $\hat{y}$.

### 2. Loss Calculation

After the forward pass, the network's prediction $\hat{y}$ is compared to the actual target $y$ using a loss function $L(\hat{y}, y)$. A common choice for the loss function is the mean squared error (MSE) for regression tasks or the cross-entropy loss for classification tasks.

For example, the MSE loss for a regression problem where the output layer is composed by $n$ neurons is:

$$L(\hat{y}, y) = \frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_j)^2$$

### 3. Backward Pass (Gradient Computation)

The goal of backpropagation is to compute the gradients of the loss function $L$ with respect to the weights and biases. These gradients tell us how much each weight and bias needs to be adjusted to minimize the loss.

For the output layer, the derivative of the loss function with respect to the activation $a_j^{(L)}$ (where $L$ denotes the output layer) is computed first. If MSE is used, the gradient of the loss with respect to the activation $a_j^{(L)}$ is:

$$\frac{\partial L}{\partial a_j^{(L)}} = 2(a_j^{(L)} - y_j)$$

Next, we compute the gradient of the loss with respect to the pre-activation value $z_j^{(L)}$ (before applying the activation function):

$$\frac{\partial L}{\partial z_j^{(L)}} = \frac{\partial L}{\partial a_j^{(L)}} \cdot \sigma'(z_j^{(L)})$$

where $\sigma'(z_j^{(L)})$ is the derivative of the activation function used in the output layer.

For hidden layers, the gradients are computed using the chain rule, propagating the error backward. The error in layer $l$ is related to the error in layer $l+1$ by:

$$\delta_j^{(l)} = \left( \sum_k \delta_k^{(l+1)} w_{kj}^{(l+1)} \right) \cdot \sigma'(z_j^{(l)})$$

Here, $\delta_j^{(l)}$ represents the error term for neuron $j$ in layer $l$, and $\sigma'(z_j^{(l)})$ is the derivative of the activation function for neuron $j$ in that layer.

The gradients of the loss with respect to the weights and biases can now be computed. For a given weight $w_{ji}^{(l)}$ between neurons $i$ and $j$ in layer $l$, the gradient is:

$$\frac{\partial L}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} a_i^{(l-1)}$$

Similarly, the gradient with respect to the bias of neuron $j$ in layer $l$ is:

$$\frac{\partial L}{\partial b_j^{(l)}} = \delta_j^{(l)}$$

These gradients are calculated for all layers, starting from the output layer and propagating backward through the hidden layers.

### 4. Weight Update

Once the gradients have been calculated, the weights and biases are updated to reduce the error. This is typically done using stochastic gradient descent (SGD) or a variant. The update rule for the weights is:

$$w_{ji}^{(l)} = w_{ji}^{(l)} - \eta \frac{\partial L}{\partial w_{ji}^{(l)}}$$

And for the biases:

$$b_j^{(l)} = b_j^{(l)} - \eta \frac{\partial L}{\partial b_j^{(l)}}$$

Here, $\eta$ is the learning rate, which controls how much the weights are adjusted in each iteration. The gradients computed during backpropagation guide the updates, ensuring that the network gradually reduces the loss and improves its predictions.

### 2.5.7 Spatio-Temporal Backpropagation (STBP) for Regression in SNNs

Following the introduction of backpropagation in Fully Connected Neural Networks (FCNNs), we now delve into its adaptation for Spiking Neural Networks (SNNs) in the context of regression tasks, following the works by Wu et al. and Liao et al. [39, 27]. While backpropagation is crucial for adjusting weights and biases in traditional Artificial Neural Networks (ANNs), SNNs' spike-based communication and temporal dynamics require specialized modifications. In this case, the final predictions are continuous values, encoded in the membrane potential of non-spiking Leaky Integrate (LI) neurons in the output layer. The hidden layers, meanwhile, are composed of Leaky Integrate-and-Fire (LIF) neurons, which handle the spiking activity.

Spatio-Temporal Backpropagation (STBP) is a technique that allows efficient training of SNNs by addressing the non-differentiability of spikes through surrogate gradients and incorporating backpropagation through time (BPTT) [40] to manage the network's temporal behavior. In the following, we explore how STBP is employed for SNN regression tasks, using the Mean Squared Error (MSE) as the loss function and the arctangent (arctan) function as the surrogate gradient.

#### Forward Pass in Regression with SNNs

In SNN regression, inputs are processed through layers of LIF neurons, which accumulate and integrate information over time. These neurons spike when their membrane potential exceeds a certain threshold, transmitting information to subsequent layers. However, unlike classification tasks where the output is often a spike rate, in regression, the final layer consists of non-spiking LI neurons. The membrane potential of these neurons directly encodes the continuous prediction values at the final time step.

This approach allows SNNs to harness their unique temporal integration capabilities while still producing smooth, continuous outputs suited to regression tasks.

#### Loss Calculation with MSE

For regression, the Mean Squared Error (MSE) is the most suitable loss function, as it measures the difference between the network's predicted values (represented by the membrane potentials in the non-spiking output layer) and the actual target values. Minimizing this loss ensures that the network's continuous predictions get closer to the true values over time, improving performance on regression tasks.

### Surrogate Gradient with Arctangent Function

One of the central challenges in training SNNs is that spikes are inherently non-differentiable, which prevents the direct use of gradient-based optimization methods. To circumvent this, surrogate gradients are used to approximate the gradient of the spike generation process, allowing backpropagation to function effectively.

Different function can be used as the surrogate gradient, the most common are the the arctangent (arctan) function and the sigmoid, that we can see in Figure 2.10. This approximation allows gradients to be computed during training, even though the actual spike generation is not differentiable.

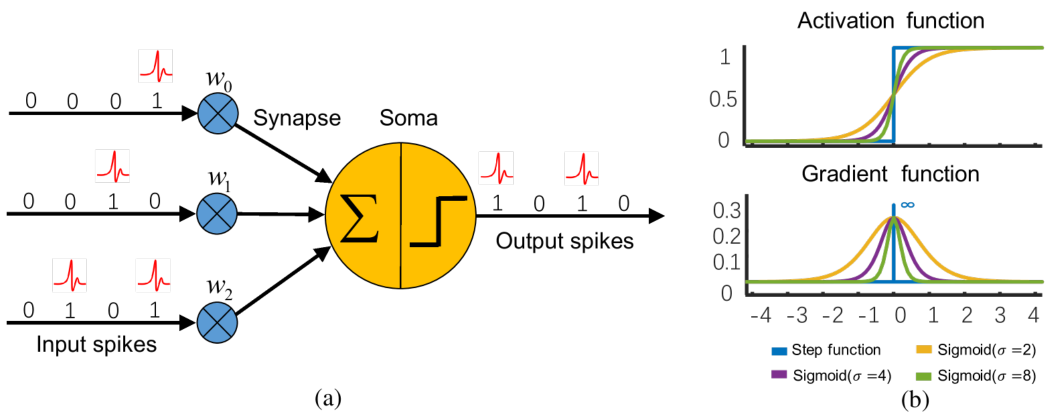
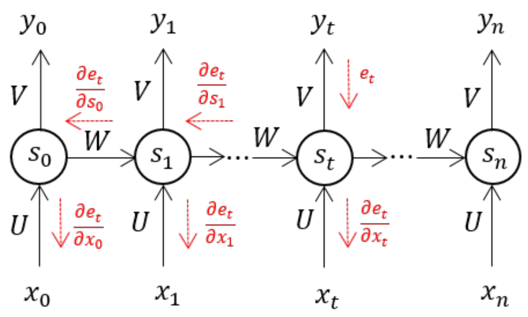


**Figure 2.10:** (a) Diagram of a spiking neuron. (b) The Sigmoid function used as surrogate gradient to approximate Heaviside function. Taken from [41].

### Backward Pass with BPTT in SNNs



$y_0$
$y_1$
$y_t$
$y_n$
$V$
$\frac{\partial e_t}{\partial s_0}$
$\frac{\partial e_t}{\partial s_1}$
$e_t$
$W$
$s_0$
$s_1$
$s_t$
$s_n$
$U$
$\frac{\partial e_t}{\partial x_0}$
$\frac{\partial e_t}{\partial x_1}$
$\frac{\partial e_t}{\partial x_t}$
$x_0$
$x_1$
$x_t$
$x_n$

**Figure 2.11:** Error backpropagation through time in unfolded RNN. Image taken from [42].

The backward pass in Spiking Neural Networks (SNNs) builds upon the principles of standard backpropagation, but it is extended to handle the temporal nature of SNNs through Backpropagation Through Time (BPTT). Unlike traditional feedforward networks, where input data is static, SNNs deal with time-varying data, requiring a more sophisticated approach to process the temporal dynamics of spiking neurons.

To manage this, the dataset is discretized into fixed-length time steps, and these time steps are grouped into short sequences. Each sequence contains a fixed number of samples and is often designed to overlap with adjacent sequences to capture continuous temporal patterns. The overlapping allows the network to better track temporal dependencies across a broader time window and ensures smoother transitions between sequences.

During BPTT, the network is trained on these folded sequences, where each sequence represents a chunk of the input data, discretized into time steps. The network processes the entire sequence, and at the end, the loss is computed by summing the losses at each individual time step within that sequence. This method enables the network to be effectively "unfolded" across time, treating the temporal sequence as a continuous flow of information rather than isolated static inputs.

The core idea of BPTT is to propagate gradients backward not only through the spatial layers of the network but also through the temporal dimension. This allows the model to learn how activations (spikes) at earlier time steps influence future predictions, tackling the temporal credit assignment problem. The gradients calculated from the loss at each time step are accumulated and then used to update the weights and biases of the network, improving its ability to learn temporal patterns.

This approach ensures that the network learns effectively from temporal data, while accounting for the spiking nature of SNNs, where neurons communicate through discrete spikes over time. Additionally, early predictions in each sequence are often discarded, as the network takes time to "warm up" and settle into a stable state, leading to more accurate predictions toward the latter part of each sequence.

By discretizing time steps, folding the dataset into overlapping sequences, and using BPTT, SNNs can learn from complex time-varying data, leveraging both spatial and temporal dynamics in their learning process.

For a regression task, on the final non-spiking layer the gradients are calculated based on the membrane potentials, which represent the network's predictions. For the hidden LIF layers, the surrogate gradients are used to approximate the gradients of the spiking neurons, allowing errors to be propagated backward through the spikes.

This backward pass, combining surrogate gradients and BPTT, ensures that both the temporal dynamics and spatial relationships in the network are considered during training.

### 2.5.8 Optimization Variants: AdamW and Dropout

In our study, following [27], during training of SNN, we introduce two key optimization techniques: AdamW [43], which we use as a variant of the more classical Adam, an adaptive Stochastic Gradient Descent (SGD) technique, and Dropout [44], which we use to prevent overfitting. These methods help ensure stability, efficiency, and generalization during training. Below, we present both methods and explain how they are applied to update the model's weights and biases.

#### AdamW as a Variant of SGD

While Stochastic Gradient Descent (SGD) is a fundamental optimization technique, its simplicity can lead to slow convergence and issues with stability, particularly in deep networks like Spiking Neural Networks (SNNs). To address these limitations, we employ AdamW, a more advanced optimization algorithm that builds upon the strengths of SGD by incorporating adaptive learning rates and improved weight decay handling.

The traditional weight update rule for SGD is as follows:

$$w_{ji}^{(l)} = w_{ji}^{(l)} - \eta \frac{\partial L}{\partial w_{ji}^{(l)}} \tag{2.6}$$

where $\eta$ is the learning rate, and $\frac{\partial L}{\partial w_{ji}^{(l)}}$ is the gradient of the loss $L$ with respect to weight $w_{ji}^{(l)}$ at layer $l$.

In contrast, AdamW improves on this by introducing two key innovations: adaptive learning rates for each parameter, based on estimates of the first and second moments of the gradients and decoupled weight decay, helping prevent overfitting.

The simplified update rule for AdamW is as follows:

$$w_{ji}^{(l)} = w_{ji}^{(l)} - \eta \left( \frac{\alpha \hat{m}_{ji}^{(l)}}{\sqrt{\hat{v}_{ji}^{(l)}} + \epsilon} + \lambda w_{ji}^{(l)} \right) \tag{2.7}$$

where $\hat{m}_{ji}^{(l)}$ and $\hat{v}_{ji}^{(l)}$ represent the estimates of the first and second moments (mean and variance corrected and weighted using the initial parameters $\beta_1$ and $\beta_2$) of the gradient for weight $w_{ji}^{(l)}$, $\alpha$ is another initial parameter, $\epsilon$ is a small constant for numerical stability, and $\lambda$ is the weight decay factor. For a more formal definition refer to the paper by Ilya Loshchilov and Frank Hutter [43].

By using AdamW, we ensure that our model converges faster and with greater stability, while avoiding overfitting by properly regularizing the weights.

#### Dropout to Prevent Overfitting

In addition to weight decay, Dropout is a popular regularization technique that helps prevent overfitting during the training process [44]. Overfitting occurs when the model becomes too specialized in fitting the training data, failing to generalize well to unseen data. Dropout works by randomly deactivating (or "dropping out") a portion of the neurons during each training iteration, forcing the network to learn more robust and general features.

During the forward pass of training, a fraction $p$ of the neurons in each layer are randomly set to zero. This prevents the network from relying too heavily on any single neuron and encourages the network to distribute its learning across the available units. Dropout is typically only applied during training; during inference, all neurons are active, and their outputs are scaled appropriately to account for the dropout that occurred during training.

This technique has been widely effective in preventing overfitting, particularly in deep neural networks like our SNN model, which requires robust generalization due to its complex temporal dynamics.

#### Conclusion

Spatio-Temporal Backpropagation (STBP), in conjunction with surrogate gradients and Backpropagation Through Time (BPTT), enables the efficient training of Spiking Neural Networks (SNNs) for regression tasks. By using the arctangent function as the surrogate gradient, STBP allows smooth gradient computation, ensuring that even non-differentiable spikes contribute to learning. With the Mean Squared Error (MSE) as the loss function, the network can directly optimize its predictions, which are encoded as membrane potentials in the non-spiking LI neurons of the final layer. This approach unlocks the power of SNNs for continuous-value prediction tasks, while addressing the challenges posed by their unique spiking dynamics.

Furthermore, by employing both AdamW and Dropout, we provide a strong foundation for training our SNN model efficiently while avoiding overfitting, ultimately improving its performance across different tasks.

### 2.5.9 Overview of Spike Encoding Methods

A fundamental step in the spiking world is encoding input data into spike trains. It is critical as SNNs can only work with spikes and the choice of the right method influences how effectively the network can learn and perform various tasks. This section provides a formal overview of three principal spike encoding methods—rate encoding, temporal encoding, and phase encoding—and introduces an adaptive encoding approach where the encoding layer is trained alongside the rest of the network.

### Rate Encoding

Rate encoding, also known as firing rate encoding, is one of the most fundamental methods used in SNNs due to its simplicity and direct correlation with traditional artificial neural networks. In this approach, information is represented by the average firing rate of a neuron over a specified time window [29]. The intensity or magnitude of the input signal is translated into the number of spikes generated within this period; higher input values correspond to higher firing rates, while lower input values result in fewer spikes.

Mathematically, the firing rate can be expressed as:

$$r = \frac{n}{T} \tag{2.8}$$

where denotes the number of spikes emitted in the time window . This method is advantageous because it is robust to temporal noise and does not require precise spike timing. It provides a straightforward mechanism to encode continuous input values into spike trains that the SNN can process. However, rate encoding often necessitates longer time windows to accurately estimate the firing rate, which can be a limitation in applications that demand rapid processing or have strict latency requirements.

### Temporal Encoding

Temporal encoding utilizes the precise timing of individual spikes to represent information [34], offering a more efficient use of spikes compared to rate encoding. In temporal encoding, the exact times at which spikes occur carry significant information about the input signal. Two common variations of this method are time-to-first-spike coding and inter-spike interval coding.

In time-to-first-spike coding, the latency between the onset of a stimulus and the first spike emitted by a neuron encodes the input value. A shorter latency indicates a stronger input signal, establishing an inverse relationship between spike timing and input magnitude. This can be mathematically represented as:

$$I \propto \frac{1}{t_{\text{latency}}} \tag{2.9}$$

where is the input value and is the latency to the first spike. Inter-spike interval coding, on the other hand, conveys information through the intervals between consecutive spikes, with variations in these intervals reflecting changes in the input signal.

Temporal encoding is advantageous for tasks requiring rapid information transmission and can enhance computational efficiency by reducing the number of spikes needed. It aligns closely with certain biological neural processes, providing increased biological plausibility. However, this method demands precise control over spike timing and can be more susceptible to noise and variability, posing challenges in implementation and robustness.

### Phase Encoding

Phase encoding represents information through the phase relationship between neuronal spikes and an underlying oscillatory cycle, such as theta or gamma rhythms commonly observed in neural systems [28]. In this method, spikes are timed relative to a global oscillation, and the phase angle at which a neuron fires conveys the input information.

The phase angle at which a spike occurs can be calculated using the expression:

$$\phi = 2\pi \left( \frac{t \mod T_{\text{osc}}}{T_{\text{osc}}} \right) \tag{2.10}$$

where is the spike time, is the period of the oscillation, and *mod* denotes the modulo operation. Phase encoding facilitates synchronization across different neural populations and can efficiently represent complex, high-dimensional data by leveraging the temporal structure of oscillations.

This method is particularly beneficial in neural systems where oscillatory activity plays a significant role in information processing and communication. However, implementing phase encoding in artificial systems can be complex, as it depends on maintaining consistent and precise oscillatory cycles. Additionally, it may require sophisticated mechanisms to generate and synchronize these oscillations within the network.

### Trainable Encoding

A recent approach to spike encoding involves integrating the encoding process directly into the training of the SNN, allowing the network to learn how to convert continuous input data into spike trains optimally [29]. In this method, raw input data are fed directly into the SNN, and the first hidden layer serves as encoding layer, learning how to perform the encoding during the training process.

This encoding layer consists of spiking neurons with trainable parameters, such as synaptic weights, membrane potentials, and threshold values. As the network undergoes training, these parameters are adjusted to enable the neurons in the encoding layer to emit spike trains that represent the input data effectively for the task at hand. This adaptive encoding process is governed by the same learning rules and optimization algorithms applied to the rest of the network.

The key advantage of this trainable encoding layer is that it allows the encoding scheme to be tailored specifically to the data distribution and the requirements of the task, potentially leading to better performance than static, predefined encoding methods. By learning the encoding in conjunction with the rest of the network parameters, the SNN can discover novel and efficient ways to represent inputs, enhancing adaptability and generalization capabilities.

This approach reduces the need for manual preprocessing or the design of separate encoding mechanisms. It is particularly effective in complex tasks such as image recognition, speech processing, or any application where the optimal encoding of input data is not readily apparent. By leveraging the network's capacity to learn from data, the encoding layer can adapt to various input modalities and distributions, making the SNN more robust to changes in the input environment.

#### Implementation in Our Model

Selecting an appropriate spike encoding method is crucial for the performance and efficiency of spiking neural networks. Rate encoding offers simplicity and robustness but may be limited in speed and efficiency. Temporal encoding provides rapid information transmission and efficient spike usage but requires precise timing control and may be sensitive to noise. Phase encoding facilitates synchronization and efficient representation of complex data but relies on consistent oscillatory mechanisms that can be challenging to implement.

The introduction of a learnable encoding layer represents an interesting advancement in spike encoding methodologies. By incorporating the encoding process into the network's training, this approach allows the SNN to actively learn the most effective way to represent input data as spike trains. This end-to-end learning enhances the network's adaptability, performance, and flexibility, potentially surpassing the capabilities of static encoding schemes and enhancing the long-term stability of the model. Our hypothesis it that this approach, combined with coherent preprocessing methods (like SBP) can improve long-term stability of decoders.

## 2.6 Adaptation Techniques for Temporal Robustness

Neural recordings, such as Electrocorticography (ECoG) data, can vary significantly across different recording sessions due to various factors, including changes in the brain's physiological state, sensor placements, or environmental conditions. These variations can degrade the performance of machine learning models if not addressed properly, especially when the goal is to generalize a decoder across multiple sessions or adapt it over time.

In previous sections, we discussed the training methods necessary to develop an efficient SNN-based regression model, which predicts the 2D kinematics of hand movements from neural signals. However, to ensure that the model remains effective over time, it is necessary to introduce adaptation techniques that allow the model to handle shifts in the data distribution without requiring constant retraining. This chapter focuses on two primary strategies for adaptation: unsupervised techniques, which adjust subsequent sessions without needing labeled ground truth, and supervised techniques, which involve selectively fine-tuning parts of the model to adapt to temporal changes.

In the following sections, we will present both approaches, starting with the unsupervised techniques that leverage linear transformations and then moving on to the supervised fine-tuning of specific model components.

### 2.6.1 Unsupervised Adaptation Techniques

Unsupervised adaptation techniques are particularly advantageous in situations where labeled data from new recording sessions is limited. These methods enable the model to adjust to variations in the input distribution without requiring explicit information about the output. In our study, the objective was to keep the model static and to adapt new electrocorticography (ECoG) data to it, without the need for access to the two-dimensional hand kinematics that we aim to predict.

The literature indicates that the covariance matrix of the motor cortex (M1) remains relatively stable over extended periods [45], while the behavior of individual neurons can exhibit considerable variability from session to session. This variability can arise from several factors, including sensor misplacement and synaptic plasticity, as discussed in Section 1.3.1.

Linear transformations of data that leverage the stability of the covariance matrix through time may represent an effective strategy for unsupervised adaptation across sessions. We address this challenge through the application of two linear transformation techniques. One technique utilises principal component analysis (PCA) to extract and align the principal components from different sessions [46], while the other involves a straightforward online normalisation. Both approaches are calculated using a little portion of the new session to then align the remaining data with the original one.

It is important to highlight that our approach leverages PCA-based alignment in a different context compared to existing literature. Previous studies [46] have successfully applied this method to sessions containing identical movements performed in the same sequences, making it a supervised approach. In contrast, our aim is to generalize this alignment technique to accommodate more diverse sessions with varying movements, thus transitioning to an unsupervised framework.

### 2.6.2 Aligning Neural Data with PCA

This section discusses methodologies for aligning covariance matrices of neural recordings from different sessions through the application of PCA, emphasizing the alignment of principal components (PCs) as a more efficient alternative to Canonical Correlation Analysis (CCA), which would deal with datapoints directly, instead of just covariance matrices. The aim of this approach is to align a inference session to a reference session, referred as $X_{\text{inf}}$ and $X_{\text{ref}}$.

#### Using PCA to Extract Stable Neural Manifolds

The main idea involves utilizing PCA to extract stable neural manifolds from each recording session. These manifolds are defined as latent spaces that encapsulate the essential dynamics of neural activity, particularly for decoding movement-related information [46, 45]. The process begins with the calculation of the covariance matrix, which reflects how the activity of different neurons co-varies across the session. From this covariance matrix, the eigenvectors (PCs) are extracted, revealing the directions of maximum variance in the neural data. Corresponding eigenvalues indicate how much variance is accounted for by each PC. Then it is possible to align the sessions through their manifolds expressed as PCs.

Importantly, the approach discussed here does not aim to reduce the dimensionality of the data. Although PCA is frequently applied for dimensionality reduction, the focus is instead on identifying and aligning the PCs that represent the underlying dynamics of neural activity without discarding any information. This methodology allows for a complete representation of the data while leveraging the stable characteristics of the neural manifolds for subsequent analysis. In a second refinement step, as novelty, we include also the eigenvalues in the process, in order to not only rotate the covariance matrices, but also to rescale them.

#### Covariance Matrix Decomposition: Leveraging PCA and SVD

Given the symmetry of the covariance matrix derived from neural signals, Principal Component Analysis (PCA) is used to decompose the neural data matrix $X$ from each session into principal components and eigenvalues. Specifically, the PCA of the reference session data $X_{ref}$ produces:

$$W_{ref}, \Lambda_{ref} = \text{PCA}(X_{ref}) \tag{2.11}$$

Where $W_{ref}$ is the matrix of principal components (eigenvectors) and $\Lambda_{ref}$ is the diagonal matrix of eigenvalues, representing the variance captured by each principal component. This decomposition captures the primary orthonormal directions of variation in the data, making the data easier to align between sessions. PCA was used in practice for this decomposition because it is simple, needs only covariance matrix, which can be easily computed online, and directly orders the components by variance explained, simplifying alignment.

While PCA was our method of choice, the relationship between PCA and Singular Value Decomposition (SVD) provides a theoretical foundation for the alignment process. For symmetric matrices like the covariance matrix, SVD yields the same principal components and singular values. The SVD of $C_{ref}$, covariance matrix of $X_{\text{ref}}$ would be:

$$C_{ref} = U_{ref} \times \Sigma_{ref} \times V_{ref}^{T} \tag{2.12}$$

Where $U_{ref}$ contains the singular vectors (equivalent to the principal components in PCA), and $\Sigma_{ref}$ is the diagonal matrix of singular values of the covariance matrix, which are the eigenvalues of the original dataset. Thus, in practice, PCA directly provides both the principal components $W$ and the singular values of the original session (through the square root of the eigenvalues), making SVD and PCA practically interchangeable in this case.

Furthermore, this equivalence will allow us to interpret the singular values obtained via PCA as the proper scaling factors when aligning sessions, and refine this alignment approach.

### Aligning Sessions through PCA

Following the extraction of the PCs and eigenvalues for each session, attention turns to aligning these components to facilitate meaningful analysis. Instead of employing the older CCA approach on single-unit activity (SUA) data from the motor cortex (M1) as outlined in the literature [45], a new method based on PCA has been proposed, which enhances computational efficiency, as noted in [46].

The online adaptation process consists of two main steps. First, the PCs of the new session are determined using a little portion of its data. Subsequently, these PCs serve as a linear transformation to align the new session's data with that of the reference session. This alignment can be mathematically expressed as:

$$x_{\text{aligned}} = W_{\text{ref}} \times W_{\text{inf}}^{T} \times x_{\text{inf}} \tag{2.13}$$

The advantage of utilizing PCA over CCA is highlighted in the literature, particularly in the context of the proposed alignment technique. PCA matrices are orthonormal, simplifying the inversion process to a straightforward transposition. This eliminates the computational complexity associated with decomposing matrices, a requirement in CCA. As emphasized in the paper by Zanghieri et al. [46].

### 2.6.3 Supervised Adaptation Techniques

In addition to the unsupervised adaptation strategies previously explored, supervised adaptation provides a direct method to handle variations in input data across sessions. We decided to complement unsupervised adaptations with a more traditional approach: fine-tuning the pre-trained model using a small set of labeled data from the new session. Depending on the extent of the data shift, fine-tuning can target either the entire network or specific layers. By incorporating session-specific labeled data, the model recalibrates its parameters, enhancing its ability to generalize to new conditions while retaining knowledge from prior sessions.

This flexibility allows the adaptation process to be tailored according to the extent of data shifts. The specifics of these supervised fine-tuning methods, including layer selection and the amount of new labeled data required, will be discussed in further detail in the methods section.

#### Fine-Tuning Strategies

Fine-tuning can either involve the entire model or selectively focus on specific layers. Recalibrating all parameters is beneficial when input data across sessions differs significantly. By updating the entire model, all layers adapt to reflect the new data distribution, improving the model's ability to generalize. This comprehensive adjustment is suitable for scenarios where session-to-session variations are broad. In many cases, the variations may be more localized. In our context, this translates to fine-tune only some components of the model, complementing the unsupervised adaptations techniques already implemented. This selective fine-tuning not only enhances efficiency but also aligns with the need for a lightweight, simple, and resource-efficient solution, especially in embedded hardware environments.

While this theoretical advantage is promising, real-world performance must be validated not only with software experiments, performed in our study, but also through hardware implementation and testing to fully understand the impact on energy and memory efficiency.

#### Theoretical Considerations for Fine-Tuning the First Layer

In our context, fine-tuning the first layer, which processes SBP-preprocessed neuronal signals, offers a key advantage. This layer directly interacts with the input signals, making it sensitive to session-specific changes and enabling effective adaptation while maintaining the stability of higher-level features.

Our approach has another potential benefit. The model is entirely trained using STBP, a supervised training method that incorporates techniques from ANNs. However, future improvements could explore training the first layer—responsible for encoding spikes—using unsupervised methods. This change could enable unsupervised adaptation to new sessions without relying on labeled data, offering greater flexibility in the adaptation process. Unfortunately, due to time constraints, this possibility was not fully explored in this work but remains a promising direction for future research. By investigating unsupervised training strategies for the encoding layer, we could enhance adaptability to new sessions while reducing the need for labeled data.

### 2.6.4 Conclusion and Future Directions

In this section, we reviewed unsupervised and supervised adaptation techniques aimed at improving the robustness of Spiking Neural Networks (SNNs) for Brain-Computer Interface (BCI) applications.

Unsupervised methods as online normalisation and PCA, were explored for aligning neural data across sessions without the need for labeled data. By aligning principal components, instead of employing the more computationally intensive Canonical Correlation Analysis (CCA), PCA offers a more efficient means of compensating for distributional shifts in neural manifolds. However, further evaluation is required to confirm its effectiveness in contexts where neural signals show significant variability. In the literature, PCA-based alignment has typically been applied to the same specific movements across sessions, ensuring consistency in the underlying neural dynamics. In our study, however, the movements involved are similar in nature (e.g., reaching for a target) but not identical to previous sessions. This difference introduces greater variability in neural activity, making it necessary to assess whether PCA can still effectively capture and align the underlying manifold in such scenarios. It is possible that a simple alignment technique like online normalisation calculated over a little portion of the new session may be preferable.

On the supervised side, we examined fine-tuning different layers, focusing on the theoretical reasons behind the choice of the first hidden layer of our model. This component is responsible for encoding preprocessed neural signals like Spike Band Power (SBP) or interprets Threshold Crossing Rate (TCR). While Spike-Time-Dependent Backpropagation (STBP) enables adaptation, it can be computationally intensive.

In this thesis, we will further investigate these strategies for BCI systems, to provide a foundation for improving unsupervides adaptation to dynamic neural environments.

# Chapter 3

# Related Works

In this chapter, we aim to provide a comprehensive review of the current state of the art in motor BMIs, in particular 2D hand velocity decoding from M1 cortex. By doing so we analysed various key research papers, and special emphasis has been placed on contributions that adopted the same dataset [47] we used. By synthesizing these insights, we will uncover the trends, limitations, and key innovations that have shaped the development of motor decoders.

Alongside this review, we will explore the rising potential of Spiking Neural Networks (SNNs), a neuromorphic architecture inspired by the brain's natural spike-based communication. Unlike traditional ANNs, SNNs promise significant improvements in energy efficiency and power consumption, making them particularly attractive for BMIs. We will investigate the scale of these improvements and assess their practical implications for BMI systems.

Lastly, we will delve into the preprocessing techniques used to enhance neural signal quality and decoding accuracy. A special mention will be given to Entire Spiking Analysis (ESA), and Spike Band Power (SBP), which have emerged as powerful methods in the field. This discussion will help clarify the most effective approaches for optimizing neural signal analysis, offering insights into how these techniques can further improve the performance of motor BMIs.

## 3.1 State-of-the-Art Hand Velocity Decoding in BMIs

Literature highlights several approaches for decoding hand velocity in Brain-Machine Interfaces (BMIs), ranging from established methods like Kalman Filters to emerging techniques utilizing spiking neural networks. This overview provides a detailed exploration of these decoders, emphasizing their strengths, limitations, and potential for advancing BMI technology.

### 3.1.1 Traditional Filters: The Foundation

Kalman Filters (KFs), particularly velocity Kalman filters (vKFs), are widely used for continuous state decoding in BMIs. They estimate hand velocity from neural features like spike counts within a defined time bin. However, KFs are linear decoders, which may not optimally capture the non-linear relationships between neural activity and hand kinematics. Wiener Filters (WFs), similar to KFs, have been widely used, but they share the same linear characteristics that might exhibit suboptimal performance due to the non-linear nature of neural signals.

Unscented Kalman Filters (UKFs) and Wiener Cascade Filters (WCFs) respectively address the linearity limitation of KFs and WFs by incorporating non-linear transformations and extensions. While they have demonstrated superior performance in decoding hand velocity, they still rely on the assumption of stationary Gaussian noise, which may not always hold true for neural signals, as stated in the paper by Ahmadi et al. [3].

### 3.1.2 Deep Learning Decoders: Embracing Complexity

Despite a relevant increase in power, complexity and energy consumption, deep learning's capacity to model complex, non-linear relationships has led to its increasing adoption in BMI decoding, replacing lighter but under performing traditional filters. Long Short Term Memory recurrent neural networks (LSTMs) excel at capturing temporal dependencies in sequential data, making them suitable for decoding hand kinematics from time-varying neural signals. Studies have shown LSTMs outperforming KF-based decoders for hand kinematic decoding from Local Field Potentials (LFPs) [2], and their superior performance to other recurrent approaches like GRU when using MUA signals [48, 49]. Moreover, like reported by Ahmadi et al. [19] LSTM quality can be further enhanced with ad hoc preprocessing methods, like Bayesian Adaptive Kernel Smoother (BAKS) applied to MUA. Further researchs [3] suggest that Quasi Recurrent NNs (QRNNs), coupled with Entire Spiking Activity (ESA) as input, achieve significantly higher decoding performance than other decoders, including KFs, UKFs, LSTMs, and other recurrent neural networks, as shown in Figure 3.1.

This figure and the related study are very important for our work, as ESA shows similarities to SBP, and coupled with the QRNN represents the state-of-the-art for prediction quality over our dataset [47].

Finally, another study by Ahmadi et al. from 2019 stated that also Temporal Convolutional Networks (TCNs), in pair with raw LFP can offer a valuable end-to-end approach, automatically learning features from raw signals and decoding hand kinematics [50].

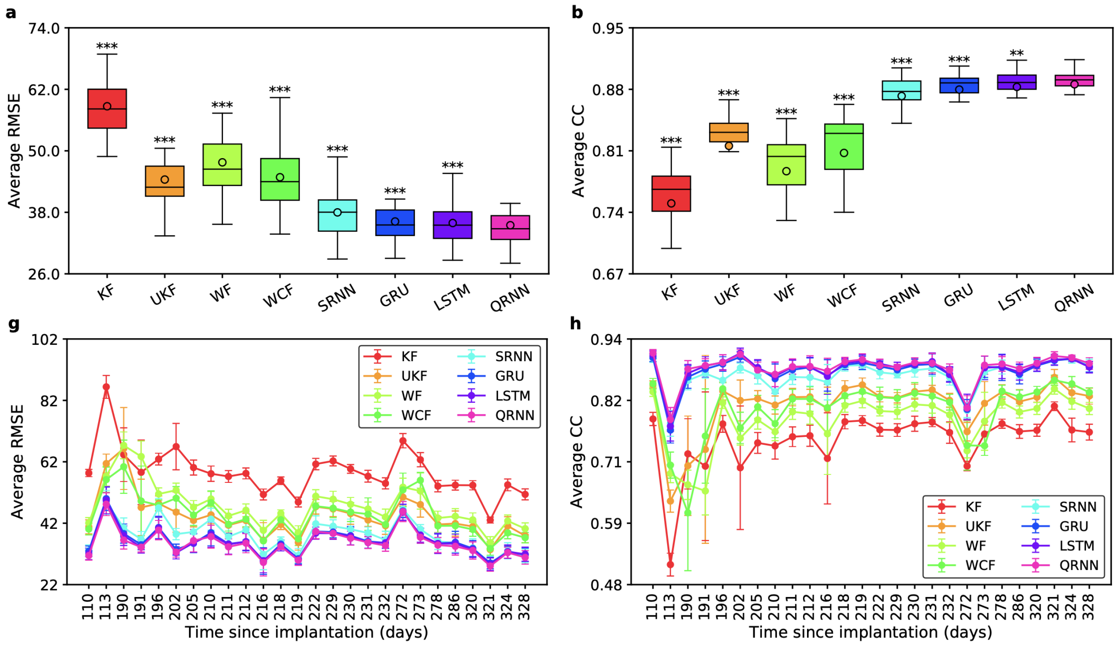


**Figure 3.1:** In this figure from the work by Ahmadi et al. [3] we can see a comparison (CC and RMSE) of 8 different decoders over 26 sessions, spanning on 218 day.

### 3.1.3 Spiking Neural Networks: High-Performing and Energy-Efficient

SNNs present a compelling alternative to traditional ANNs for BMI decoding, particularly in power-constrained implantable devices. While being comparable to filters for energy consumption, they are theoretically capable of ANN state-of-the-art performances. Unlike ANNs, SNNs rely on the timing of sparse spikes to process information, which significantly reduces the number of computations and the amount of data movement. This event-driven nature makes SNNs inherently energy-efficient. A key study [27] for our research implemented an SNN decoder for finger velocity using spatio-temporal backpropagation (STBP). This SNN achieved accuracy comparable to state-of-the-art ANN decoders, while requiring only 6.8% of the computational operations and 9.4% of the memory accesses, highlighting its potential for low-power implantable BMIs. In another study [51], Chen et al. demonstrated that an SNN implementation on the Loihi neuromorphic chip [52] consumed 50 times less power than an equivalent ANN method.

## 3.2 Long-Term Stability in Brain Signal Decoding

A key challenge in Brain-Machine Interfaces (BMIs) is the long-term stability of decoding performance, particularly when relying on signals that degrade over time. Literature discuss several approaches and findings related to achieving stable decoding performance in BMIs.

### 3.2.1 From Single-Unit Activity to Spike Band Power

The choice of the method to preprocess raw signals is fundamental. It needs to retain enough information to ensure high performance and high adaptability through time, while not overloading memory or energy resources of the decoder. Ahmadi et al. in different publications [3, 2, 19] highlighted the characteristics and differences between SUA, MUA, LFPs and ESA.

Single-Unit Activity (SUA) proved to be able to provide short-term high accuracy, while its performance tends to decline over time. In contrast, Multi-Unit Activity (MUA) offers simpler processing and improved stability, though its decoding accuracy is typically lower than SUA. Local Field Potentials (LFPs), which reflect the summed synaptic activity, have demonstrated promising long-term stability, even in the absence of spike activity. Because LFPs can be sampled at lower rates, they consume less power, making them an appealing option for implantable brain-machine interfaces (BMIs). However, LFP decoding accuracy is generally lower compared to SUA.

Finally, Entire Spiking Activity (ESA), a continuous signal that captures population spiking activity, has emerged as a promising alternative. It exhibits long-term stability similar to LFPs and surpasses MUA in performance. As previously discussed, ESA-based decoders, particularly when combined with deep learning techniques like QRNNs, can achieve superior decoding accuracy, even outperforming SUA-based decoders.

An alternative approach by Nason et al. [26] is constituted by Spike Band Power (SBP), particularly within the 250-3,000 Hz range. It has been successfully integrated with spike counts in human clinical trials, enhancing decoding performance and potentially extending the operational lifespan of implanted arrays. This method has also been used with SNN [27], giving promising results. Nevertheless its long-term stability has yet to be proven.

### 3.2.2 Adaptive Strategies for Long-Term Stability

We need the right decoder architecture to properly leverage the chosen preprocess method and enhance long-term stability. Traditional decoders, like Kalman filters, might not be ideal for non-stationary neural signals, as previously stated, their performance can degrade as the relationship between neural activity and behavior changes over time, moreover their have overall lower performances than more advanced approaches. State-of-the-art decoders rely on complex machine learning architectures such LSTM and QRNN. Yet, once again, the power and resources needed to train and perform predictions on those models make them unsuitable for real world scenarios. Furthermore, the fine tuning process is usually supervised, slow, data intensive and computationally expensive [7].

Some unsupervised Learning Algorithms, like the recurrent exponential-family harmonium (rEFH), have shown superior performance compared to Kalman filter-based approaches. These methods learn from the statistical structure of neural data without relying on explicit kinematic labels, nor to the hypothesis of gaussian stationary noise or linearity, potentially offering greater robustness to signal variations. However, they still fall short in performance when compared to Artificial Neural Networks (ANNs) [25].

Changing perspective, recent studies on the stable latent dynamics within neural population activity have shown promising results for long-term decoding stability [45], while developing strategies to perfom computations online with minimal energy consumption [46]. By aligning the latent dynamics, researchers have achieved accurate predictions of behavioral features, even with neuron turnover in recordings spanning up to 2 years. This suggests that stable behavioral execution might be rooted in the consistent dynamics of neural populations, rather than in the activity of individual neurons, which are more prone to variability over time.

While population-based decoding approaches, like MUA, ESA, SBP and LFPs, have shown improved stability compared to methods relying on SUA, linear transformations aligning stable latent dynamics could be leveraged to explore unsupervised adaptations to real world scenarios.

researches on transfer and continual learning for SNNs are also giving promising results [53, 54], but further studies have to be conducted in this field to address the behaviour of SNN on fine-tuning and long-term stability, while keeping the big advantage of energy efficiency.

Our strategy focuses on minimizing fine-tuning efforts in the first hidden layer. We will create a pipeline that starts with SBP preprocessing, followed by unsupervised online linear transformations to align latent dynamics, and concludes with our SNN. This approach takes advantage of the first hidden layer's natural ability to encode spikes [27], resulting in a lightweight encoder that can be trained alongside the rest of the network and hopefully easily fine-tuned to adapt to data shifts.

# Chapter 4

# Methods

This chapter presents the methodologies used to decode two-dimensional hand kinematics from neural signals recorded in the primary motor cortex (M1) within a brain-computer interface framework. We focus on M1 due to its crucial role in voluntary motor control and its predictive capabilities for movement intentions, supported by empirical evidence and practical considerations related to data availability.

We detail the preprocessing methods applied to the raw neural data—Single-Unit Activity (SUA), Multi-Unit Activity (MUA), and Spiking Band Power (SBP)—each selected for its ability to capture neural activity pertinent to decoding motor intentions. The preprocessing procedures encompass filtering techniques, spike detection and sorting, and data binning, preparing the neural signals for input into our Spiking Neural Network (SNN) models. Common steps like data folding for temporal contextualization are also included.

To address session-to-session variability in neural recordings, we employ unsupervised adaptation techniques using normalization and Principal Component Analysis (PCA), and supervised fine-tuning, enhancing the stability and performance of the SNN models across different sessions. Finally, we describe the architecture and configuration of the SNN model, including its hyperparameters and layer structure. This model predicts 2D hand velocity from neural inputs, allowing us to explore performance trade-offs between model complexity and efficiency.

## 4.1 The choice of M1 over S1

We have made the design choice to focus on neural signals from the primary motor cortex (M1) for decoding 2D hand kinematics in our brain-computer interface (BCI) due to M1's critical role in generating and controlling voluntary movements. M1 is the most reliable source of motor-related information [2], making it ideal for accurately decoding intended hand movements.

In our design choice, the future implementations in real world scenario will rely on visual feedback of the user. This closed-loop interaction should harness the brain's natural plasticity, allowing it to integrate the BCI into its motor control schema and thereby enhance the system's accuracy and intuitiveness.

In summary, our choice to utilize M1 signals and real-time visual feedback within a closed-loop design aims to ensure robust, real-time control over assistive devices, facilitating the brain's adaptation and optimizing the BCI's long-term performance.

In designing a brain-computer interface (BCI) for decoding 2D hand kinematics, the decision to focus on signals from the primary motor cortex (M1) rather than the primary somatosensory cortex (S1) is grounded in several key factors:

**M1's Critical Role in Motor Control** M1 is the primary cortical region responsible for generating and controlling voluntary movements. It encodes detailed information about hand kinematics, making it an ideal source for decoding motor intentions [1, 4]. M1's ability to predict future movements, typically 100-150 ms ahead, is particularly valuable for real-time BCI applications, allowing for smoother and more responsive control [55].

**Empirical Success of M1-Based Decoding** Numerous studies have demonstrated that decoding hand kinematics from M1 alone is sufficient for effective BCI control, without the need to incorporate sensory feedback from S1. This success underscores M1's reliability and robustness as a source of motor-related signals [2, 55, 7, 56].

**Complexity of S1 Integration** While S1 provides important sensory feedback, its role in predicting future movements is less direct than that of M1. S1 primarily encodes current sensory feedback rather than future kinematics, which can complicate the decoding process. Integrating S1 into the BCI system would require more sophisticated algorithms to manage the dynamic interplay between motor commands and sensory information, adding significant complexity to the initial development phase [55, 56, 4].

**Brain Plasticity and Adaptation** The brain's plasticity, particularly within M1, supports the hypothesis that the brain can adapt to the BCI over time. Interacting with a BCI induces functional changes in M1, enabling the brain to refine its control signals for more effective communication with the BCI decoder [4, 56, 2]. This adaptive capacity is crucial for the long-term success and stability of the BCI, as it allows the brain to incorporate the decoder into its motor schema, enhancing intuitive control.

## 4.2 Selected Preprocessing Methods

In our study, three preprocessing methods were chosen to enhance the interpretation of neural data and ensure comparability with existing studies. These methods include Single-Unit Activity (SUA), Multi-Unit Activity (MUA), and Spiking Band Power (SBP). Each method was selected for its unique advantages and its role in providing a comprehensive view of neural activity.

**Single-Unit Activity (SUA):** The dataset chosen already preprocessed sessions extracting SUA. This method, by spike detection and sorting, captures the activity of isolated neurons and is critical for understanding the firing patterns of individual neurons. In our dataset (for the 30 sessions that we are interested in) the results is to have 480 distinct features, originated by sorting each original channel in 5 new sorted units. Each of them is trying to capture the spiking activity of a single neuron. This method provides detailed insights into neuronal behavior and it is commonly used by studies focused on neural coding and network dynamics [47].

**Multi-Unit Activity (MUA):** To complement the SUA data and ensure coherence with related studies, particularly [48], we implemented Multi-Unit Activity (MUA). MUA represents the combined spiking activity of multiple neurons recorded on a single electrode. For our analysis, MUA was obtained by binning the SUA data into the original recording channels. This binning process aggregates the spiking activity of several neurons, summing them together, providing a broader view of neural population dynamics. By aligning our methodology with that of Marta Bono [48], we aim to facilitate a meaningful comparison and validate our findings within the context of existing research.

**Spiking Band Power (SBP):** The third preprocessing method employed is Spiking Band Power (SBP), which measures the power within the high-frequency band of the neural signal. SBP is a valuable technique for analyzing spiking activity and offers several advantages in the context of ECoG data, better explained in the dedicated section 2.4. We are selecting this method due to its potential for long-term stability, low energy consumption, and reduced complexity.

## 4.3 Preprocessing Details

In this section, we outline the preprocessing steps essential for transforming raw Electroencephalography (ECoG) data into usable formats for our model, specifically focusing on Multi-Unit Activity (MUA), Single-Unit Activity (SUA), and Spike Band Power (SBP). We will detail the practical methodologies known and employed in our study, including the parameters and configurations that guide the conversion process from raw data to these derived signals. Following this, we will discuss the common preprocessing steps applied uniformly across MUA, SUA, and SBP, facilitating efficient model training and performance.

### 4.3.1 MUA and SUA Practical Implementation

The preprocessing pipeline for multi-unit activity (MUA) and single-unit activity (SUA) signals follows a well-defined sequence. The neural recordings were sampled at 24.4 kHz and filtered using a causal IIR bandpass filter (fourth-order Butterworth) with a passband of 500 Hz to 5000 Hz. After filtering, spikes were detected and sorted.

For both MUA and SUA, spike detection begins by calculating the absolute value of the filtered neural signal and identifying events that crossed a threshold, typically set between 3.5 to 4.0 times the signal's standard deviation. The spike detection process marks the points in time where these threshold crossings occur, and the corresponding timestamps are recorded.

The key difference between MUA and SUA preprocessing is the additional spike sorting step in SUA. After threshold crossings are detected, SUA applies spike sorting on the waveforms of the spikes detected —consisting of 64 samples (about 2.6 ms) around the timestamps. These waveforms are first aligned to their highest point (the waveform peak) to ensure consistency in comparison, and then reduced using PCA. This makes it easier to classify the waveforms into distinct groups.

Once the waveforms have been simplified, they are sorted into different units using pre-defined templates created by the operator. These templates act as references, helping to assign each spike to a specific unit based on its waveform shape. Depending on the recording session, spikes could be sorted into up to two or four distinct units per channel, allowing for the identification of individual neurons or sub-populations of neurons. Additionally, an unsorted "noise" or "hash" unit was maintained for each channel, containing events that crossed the threshold but did not match any defined unit template. Finally, units with spike rates below 0.5 Hz were discarded to exclude inactive or irrelevant neurons from further analysis. All this process is well described in the literature [25, 47].

For both MUA and SUA, the final preprocessing step involves binning the detected spikes into non-overlapping 4-millisecond intervals, as shown in figure 4.1, and count the detected spikes in that interval of time. This time-discrete representation of neural firing rates can be referred to as Threshold Crossing Rate (TCR). The "Threshold Crossing" phase refers to the detection of spikes, while the "Rate" aspect involves counting and summing the spikes within each time window. In MUA preprocessing, all detected spikes are retained without differentiation between units, providing a broader view of population-level neural activity. In

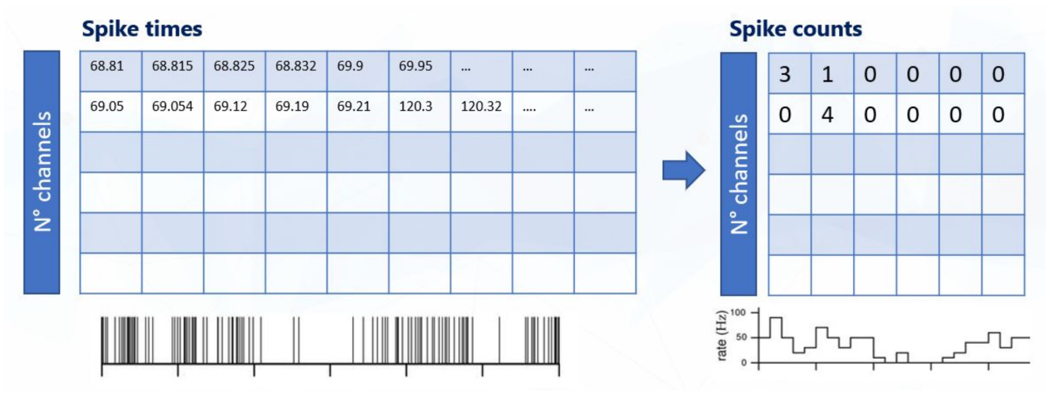


**Figure 4.1:** MUA and SUA binning and summing spikes in 4 milliseconds time intervals. Figure taken from [48].

contrast, SUA preprocessing benefits from spike sorting, allowing for the isolation of specific neuronal units.

### 4.3.2 SBP Practical Implementation

This section describes the implementation of two Spiking Band Power (SBP) preprocessing variations designed for use with Spiking Neural Networks (SNN), *SBP1000* and *SBP2000*. They share many common steps, yet they differ primarily in their band-pass filtering frequency.

To mitigate common noise, including residual power line interference, Common Average Referencing (CAR) is applied over unipolar referencing already implemented during the recording phase. This technique enhances the visibility of localized neural activity by subtracting the average signal across all electrodes from each individual channel, effectively isolating the relevant neural signals, more details in 2.3.2.

A 2nd order digital Butterworth band-pass filter with finite response is then employed to isolate the frequency range associated with spiking activity. Specifically, *SBP1000* focuses on the range of 300-1000 Hz, whereas *SBP2000* extends this range to 300-2000 Hz. The broader bandwidth utilized in the second is expected to enhance decoding performance by capturing higher frequency spiking activity.

Next, the signal magnitude is extracted by calculating the absolute value, which rectifies the signal and captures the envelope of spiking activity. Following this, the signal undergoes downsampling (10x) and averaging in non-overlapping windows of 4 milliseconds. This averaging process smooths the signal, reduces its dimensionality, and aligns the data rate with the 250 Hz sampling rate of the hand position data.

### 4.3.3 Common Preprocessing Steps

In preparing the neural data for input into our model, we employ a systematic preprocessing approach that applies uniformly to three primary methodologies: Multi-Unit Activity (MUA), Single-Unit Activity (SUA), and Spike Band Power (SBP). After initial computation, all three methods yield time steps of 4 ms, ensuring consistency across the dataset.

#### Binning and Summation for Dataset Reduction

After reducing samples to 4 ms time steps, further processing is necessary to enhance model efficiency and manageability. We achieve this by summing the samples into non-overlapping windows, typically to 32 ms in duration. The specific choice of window size is determined by the parameters selected for the model. This binning process effectively condenses the dataset, allowing for the retention of critical features while reducing the overall volume of input data. By summing the data over these larger intervals, we minimize noise and capture the underlying trends in neural activity, which are essential for effective model learning.

#### Data Folding for Temporal Contextualization

To enable the model to leverage the temporal dependencies inherent in the data, we implement data folding. This technique involves grouping time steps into larger, overlapping windows, where the overlap is defined as the window size minus one. For instance, with a selected window size of 10 samples, the resulting overlap would be 9 samples. The length of windows is fixed, and change model to model. A visual representation of the final dataset is in figure 4.2. This structure is particularly advantageous for BPTT, allowing the model to learn effectively from sequential data. By performing error backpropagation through each window, we capture the dynamic interactions present across time, thereby enhancing the model's ability to learn from the temporal context of the data.

## 4.4 Unsupervised Adaptations with Normalization and PCA

This section provides an in-depth explanation of theoretical foundation and methodologies for unsupervised adaptation aimed at reducing session-to-session variability in neural signals recorded from the primary motor cortex (M1). The goal is to improve the long-term performance of a spiking neural network (SNN) trained to predict 2D hand velocity from M1 signals. In literature these methods rely on supervised techniques that require the same repeated movements sequence across

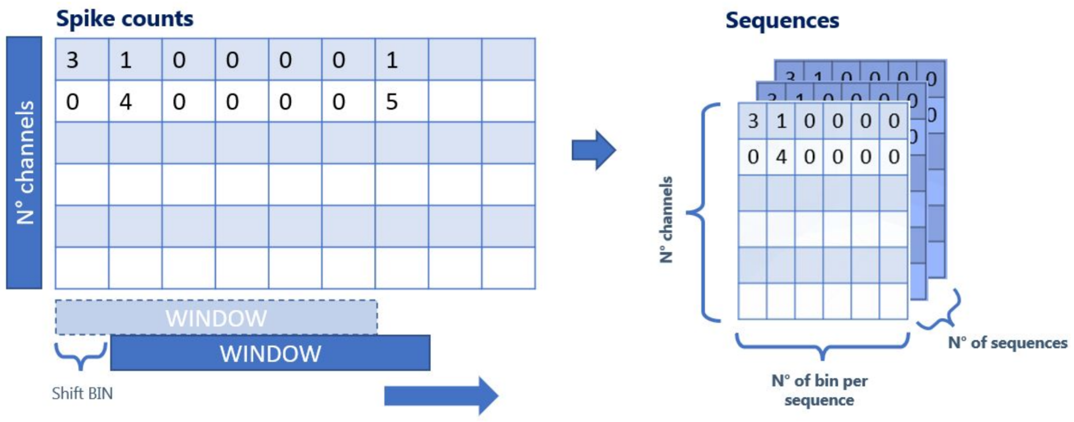


**Figure 4.2:** Data folding into overlapping windows or "Sequences". Figure is taken from [48].

sessions, but in this work, we attempt to generalise this alignment strategies for unsupervised approaches capable of adapting to sessions with varying movement patterns.

The adaptation techniques include Principal Component Analysis (PCA) and various forms of normalization, applied to align neural data. We'll refer to the reference session as $X_{ref}$ and to the new session, that will be aligned to the reference one, as $X_{inf}$.

### 4.4.1 Session Alignment Strategies based on Centering and Normalisation

Session-to-session variability in neural data poses significant challenges for neural decoding and analysis. To mitigate this variability, we implemented several alignment strategies that account for differences in both the rotation and scaling of neural data across sessions. These strategies involve steps such as centering and normalization, applied in both dynamic (online) and static (offline) contexts.

By comparing online and offline normalization, as well as centering versus normalization, we aim to determine the most effective strategy for reducing session-to-session variability. Online normalization may offer advantages by adapting to real-time changes in neural activity, while offline normalization provides a consistent baseline. Similarly, normalization may offer benefits over centering by adjusting the dynamic range of the data, potentially improving the performance of alignment techniques sensitive to input variance.

This integrated approach allows us to investigate whether continuous, real-time adjustments are more effective than static transformations in reducing variability, and whether adjusting the dynamic range of the data through normalization provides additional benefits over centering alone.

### Centering and Normalization

The first strategy in aligning neural signals across sessions is to standardize the neural activity within each session. We explored two primary methods: centering and standard scaling (normalization). Centering involves subtracting the mean activity of each neuron over time, ensuring that each channel has a zero mean while maintaining its original variance. This method is expressed as:

$$X_{\text{centered}} = X - \mu_X$$

where $\mu_X$ represents the mean activity of each input channel across the session. By centering the data, we retain the covariance relationships between neurons, which is essential for subsequent alignment techniques that rely on these relationships.

Normalization, or standard scaling, extends centering by also scaling each channel's variance to one. This process is defined by:

$$X_{\text{normalized}} = \frac{X - \mu_X}{\sigma_X}$$

where $\sigma_X$ denotes the standard deviation of each channel. Normalization eliminates differences in the scale of neural activity between channels, which is critical for alignment techniques such as Principal Component Analysis (PCA) that behaves differently in the two scenarios.

The choice between centering and normalization depends on the specific requirements of the alignment strategy. If preserving the original variance structure is important, centering may be preferred. If standardizing the variance across channels is necessary, normalization provides additional benefits by adjusting the dynamic range of the data.

### Online and Offline Normalisation

To further address session-to-session variability, we compared the impact of online (dynamic) and offline (static) normalisation. In the online approach, parameters such as the mean ($\mu$) and standard deviation ($\sigma$) are computed dynamically over the first few minutes of each inference session. These parameters are then applied during inference:

$$X_{\text{inf}} = \frac{X_{\text{inf}} - \mu_{\text{inf}}}{\sigma_{\text{inf}}}$$

This method allows for real-time adjustments to the data preprocessing, accommodating any shifts in neural activity that may occur at the start of a new session.

Offline normalization uses fixed parameters derived from a reference session, applying them uniformly to subsequent sessions without real-time updates. This static transformation assumes that the variability between sessions can be accounted for by a consistent set of normalization parameters.

## 4.4.2 Session Alignment Strategies based on PCA

We also explore the effect of normalization versus centering. While centering aligns the mean of each input channel, normalization further standardizes variance. This allows us to investigate whether adjusting the dynamic range of the data through normalization provides additional benefits.

PCA-based alignment is then applied to address rotational differences between sessions. The transformation aligns the directions of maximum variance in the feature space using the equation:

$$X_{aligned} = W_{ref} \times W_{inf}^T \times X_{inf}$$

Incorporating both rotational alignment and scaling adjustments, PCA-based alignment can be further refined by incorporating singular values (square roots of eigenvalues) derived from the covariance matrix.

### Novel Approach: Incorporating Eigenvalues

We introduce a novel approach that extends the traditional PCA method by incorporating not only the rotation indicated by the orthonormal basis formed by the principal components but also the absolute values of the singular values, which are the square roots of the eigenvalues. The final alignment approach can be expressed mathematically as:

$$x' = W_{\mathrm{ref}} \Sigma_{\mathrm{ref}} \Sigma_{\mathrm{inf}}^{-1} W_{\mathrm{inf}}^T x$$

In this formulation, $\Sigma$ represents the diagonal matrix of singular values. This inclusion of singular values allows for both alignment and scaling, ensuring that the variance in the new session is appropriately adjusted relative to the reference session. By taking this comprehensive approach, we enhance the fidelity of the alignment process, which may lead to improved performance in subsequent analyses or predictive modeling tasks.

### 4.4.3 Principal Component Reordering and Sign Alignment

To refine the alignment further, we reordered the principal components of the inference session ($W_{inf}$) to match the reference session ($W_{ref}$) based on their similarity, in case, we also reorder the singular values accordingly . The similarity matrix was computed as:

$$M = W_{inf} \times W_{ref}^T$$

From this matrix, we selected the largest values row by row, ensuring that each principal component in $W_{inf}$ was reordered to match the most similar component in $W_{ref}$. To prevent duplicates, we excluded previously selected components from consideration.

After reordering, we addressed potential sign inversions. While two principal components can point in the same direction, they may have opposite signs. We corrected these inversions by ensuring that the dot product between each reordered PC in $W_{inf}$ and its counterpart in $W_{ref}$ was positive, flipping the sign if necessary. This operation is not needed with singular values, because we are in fact dealing with their absolute values, and the centering process guarantees that data variance is symmetrical respect to the sign.

The final reordered and sign-aligned PC matrix for the inference session was:

$$W_{ord} = \text{reordering}(W_{inf}, W_{ref}^T)$$

This process is derived from established literature: [46].

## 4.5 Supervised Adaptation with Fine-Tuning

In our approach to handling session-to-session variability in spiking neural networks (SNNs) trained to predict 2D hand velocity from M1 signals, supervised adaptation via fine-tuning plays a key role. Specifically, we utilize Spatio-Temporal Backpropagation (STBP) to adjust the synaptic weights and biases while keeping the parameters of the leaky integrate-and-fire (LIF) neurons fixed. The primary focus is on weighting the connections between layers to allow the network to adapt effectively to the current session. However, all supervised adaptation steps are preceded by an unsupervised online normalization phase, applied dynamically during the first few minutes of each new session to account for neural variability across recordings.

We focus on three key strategies: retraining the model from scratch, extensive fine-tuning across all layers, and targeted fine-tuning of specific layers, particularly the first layer. Each of these strategies will be evaluated in terms of their ability to maintain long-term model performance and stability, with a particular emphasis on computational efficiency.

By leveraging unsupervised adaptation through online normalization, we aim to reduce the session-to-session variability in the input signals, enhancing the effectiveness of all subsequent supervised adaptation techniques. Our ultimate goal is to demonstrate that fine-tuning the first layer of the model, in conjunction with online normalization, can yield significant improvements in performance while minimizing the need for extensive retraining.

Given the nature of SNNs, it is uncertain whether the model retains representations from past sessions during fine-tuning. While our ongoing research will not address this issue, it is not critical in our context, as we primarily seek optimal performance in the current session. The model does not need to retain past behavior, as its primary function is to generalize effectively for the present and future sessions.

### 4.5.1 Two Baselines

#### Retraining the Model from Scratch for Each Session

In this experiment, we explore the performance of retraining the SNN from scratch for each new session. This approach involves training a fresh model for every session, ensuring that the model is fully specialized for the current neural data. While this method may provide the best possible fit for each session, it is highly resource-intensive and lacks any transfer learning capability from previous sessions.

This experiment serves as a baseline, allowing us to measure the upper limits of performance without the use of any prior session information or fine-tuning strategies. The results will provide a benchmark to compare against other adaptation methods.

#### Extensive Fine-Tuning Across the Entire Model

In the second experimental condition, we apply extensive fine-tuning across all layers of the SNN model. Here, the model is first initialized using weights learned from previous sessions, and fine-tuning is applied to all layers using the new session data. By fine-tuning the entire model, we aim to adapt to the session-specific data while preserving some of the representations learned in prior sessions. This experiment allows us to test how knowledge embedded from past sessions increase or shows comparable performances

### 4.5.2 Fine-Tuning single layers

After the initial baseline evaluation, we fine-tune the model using only a small portion of the new session data. This allows us to investigate the model's ability to generalize with minimal training data. We will experiment with fine-tuning different layers of the network to understand the impact of adjusting specific parts of the model.

The focus of this experiment is to explore the efficiency of fine-tuning when only the first layer or select layers are updated. By evaluating the model's performance after fine-tuning only a fraction of the layers, we can assess which layers are most important for adapting to new session data and determine whether lower layers, such as the first layer responsible for initial neural feature extraction, are sufficient to handle the majority of the session-to-session variability.

### 4.5.3 Efficiency of First-Layer Fine-Tuning

Our primary hypothesis is that fine-tuning the first layer of the model, after applying online normalization, will provide significant performance improvements with minimal computational cost. The first layer encodes the SBP input data and extracts fundamental neural features, making it a critical component of the network. By fine-tuning this layer, we hypothesize that the model can effectively adapt to session-specific differences while preserving higher-level representations.

Through systematic evaluation, we will determine whether first-layer fine-tuning is sufficient to achieve high performance on new sessions. This approach has the potential to significantly reduce the computational resources required for adapting the model, making it a more practical solution for real-time neural decoding applications.

## 4.6 Model Architecture and Configuration

This section provides an overview of the hyperparameters, configuration and architecture of the model we evaluated. The model is regression-based, designed to predict 2D velocity in Cartesian coordinates, specifically the left-hand kinematics of a Non-Human Primate (NHP). Consequently, the output layer of the model consists of two Leaky Integrate (LI) neurons, while the preceding layers consist of varying numbers of Leaky Integrate-and-Fire (LIF) neurons. These layers are fully connected and incorporate both weight and bias matrices. The model is trained using Spatio-Temporal Back Propagation (STBP), a strategy inspired by BPTT, borrowed from traditional Recurrent Neural Networks (RNNs). Further details are provided in Chapter 2. Moreover the model is enhanced with Dropout and AdamW.

A paper by Liao et al. [27] helped us in fixing correct hyperparameters for our task, as it is addressing a similar issue. Its purpose was to predict finger velocity on one axis for two distinct groups of fingers, of the right arm of a non human primate (NHP) using SNN. While the objective is different, as we wanted to predict 2D kinematics of the left hand, it showed enough similarities.

Hyperparameters are summarized in Table 4.1. Below is a brief explanation of key hyperparameters:

| **Hyperparameters** | |
|---|---|
| training loss | MSE |
| reset mechanism | by subtraction |
| surrogate gradient | $y = \arctan(x)$ |
| output neurons | 2 |
| input neurons | 96 (MUA and SBP) or 480 (SUA) |
| batch size | 256 |
| epochs | 24 |
| learning rate ( $\eta$ ) | 2e-3 |
| dropout | 0.3 |
| windows size (timesteps) | 10 |
| window overlap (timesteps) | 9 |
| warmup (timesteps) | 2 |
| timestep (ms) | 32 |
| $V_{\text{threshold}}$ | 0.4 |
| $\tau_{\text{hidden}}$ | 0.6 |
| $\tau_{\text{out}}$ | 0.6 |
| hidden layers | 3 |
| neurons per hidden layer | 256, 256, 256 |
| use biases | True |
| offset (ms) | 0 |

**Table 4.1:** Hyperparameters of our model.

- **Batch size**: The number of samples into which the training dataset is divided. This determines how many samples are processed together in a single forward and backward pass through the network, allowing for the calculation of loss and the optimization of weights and biases.

- **Epochs**: The number of times the entire dataset is passed through the network during training, providing multiple opportunities for the model to learn from the data.

- **Window size**: The number of timesteps considered when folding the data for BPTT. For more details on this process, refer to Section 4.3.3.
- **Overlap**: The number of timesteps shared between adjacent windows, ensuring continuity between successive data segments.
- **Warmup**: The number of initial timesteps in a window that are excluded when calculating the loss, allowing the model to stabilize before making predictions.
- **Membrane potential threshold** ($V_{\text{th}}$): The value at which a neuron emits a spike. Once this threshold is reached, the neuron resets its membrane potential by subtracting the spike value.
- **Membrane Decay** ($\tau_{\text{hidden}}, \tau_{\text{out}}$): These constants control the rate at which the membrane potential decays in the absence of input. $\tau_{\text{hidden}}$ applies to hidden layers made of LIF neurons, while $\tau_{\text{out}}$ is used for the output layer composed of LI neurons.
- **Biases**: The inclusion of biases in layer connections is optional and can be toggled based on the network's requirements.
- **Offset**: A parameter introduced to account for any potential lag between the input signal and the recorded velocities, hypothesizing that this lag may be due to delays in the experimental data.

# Chapter 5

# Results

In this section, we describe the dataset adopted, followed by each experiment. They are presented in a logical sequence to outline our research process. We began by testing the proposed model in a single-session scenario, assessing regression performance and comparing the effects of different preprocessing methods: SUA, MUA, SBP1000, and SBP2000. Next, we evaluated the multi-session performance of selected methods in both online and offline scenarios, using various supervised and unsupervised adaptation approaches. This leveraged the natural ability of SNNs to encode continuous signals (SBP) into spikes and learn this process during training, thanks to the first hidden layer, called Trainable Encoding Layer (TEL).

These experiments aim to enhance both computational efficiency and usability, making our solution more practical for real-world applications. Our goal was to address data shifts caused by both natural variations and tool-induced changes, ensuring system stability. At the same time, we prioritized minimizing user inconvenience—avoiding long retraining sessions and excessive power consumption. Our results demonstrate that with minimal retraining effort, performance can be restored and even improved.

## 5.1 Dataset

The dataset for this study was obtained using two adult male rhesus macaque monkeys (*Macaca mulatta*), named Indy and Loco, housed in the laboratory of Dr. Sabes [47]. At the time of data collection, Indy was 11 years old with a weight of 12 kg, and Loco was 9 years old weighing 14.5 kg.

We selected this dataset to ensure alignment with previous research conducted by Marta Bono [48], which explored transfer learning using the same dataset as part of the same European project. Additionally, this dataset is open-source and well-suited to our regression task. The dataset, composed of continuous reaching movements toward visual targets in a virtual reality environment, features a relatively simple task, making it ideal for evaluating the performance of our decoder. The simplicity of the reaching task offers a controlled environment for testing, while serving as a foundational step toward future implementation in real-world scenarios. Lastly, the availability of extensive literature on this dataset further supports its suitability for our study.

In Figure 5.1 we can see the whole process, from acquisition to decoding of hand kinematics.

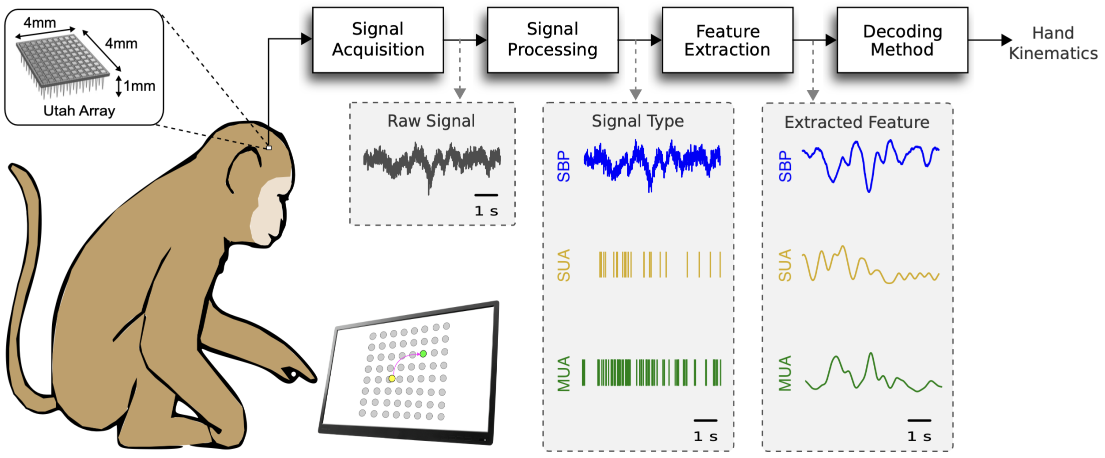


**Figure 5.1:** Complete pipeline from acquisition to decoding of 2D hand kinematics of Indy (NHP).

### 5.1.1 Data Acquisition and Preprocessing (SUA)

Each monkey underwent chronic implantation of two 96-channel silicon microelectrode arrays (Blackrock Microsystems, Salt Lake City, UT) into their sensorimotor cortex. For Indy, the implants were positioned in the right hemisphere, while for Loco, they were placed in the left hemisphere. The arrays targeted specific regions: one was implanted in the primary motor cortex (M1, Brodmann area 4) with 1.0 mm shanks coated in platinum (nominal impedance of $400k\Omega$), and the other in the primary somatosensory cortex (S1, Brodmann area 1) with 1.5 mm shanks coated in sputtered iridium oxide (nominal impedance of $50k\Omega$). The S1 arrays were designed differently because they were used for electrical stimulation in separate experiments. Both arrays were aimed at areas representing the shoulder and upper arm.

Neural signals were recorded using an RZ2 BioAmp Processor paired with a PZ2 Preamplifier (Tucker-Davis Technologies, Alachua, FL). The data were sampled at a rate of 24.4 kHz and filtered using a causal Infinite Impulse Response (IIR) bandpass filter (fourth-order Butterworth filter with a passband of 500 Hz to 5,000 Hz). Spike detection and sorting were conducted online using custom software written in C++. The process involved calculating the absolute value of the filtered neural signals and identifying events that crossed a threshold—typically set at 3.5 to 4.0 times the standard deviation. Extracted waveforms (64 samples, approximately 2.6 ms) were aligned to their peak values. Principal Component Analysis (PCA) was then employed to reduce dimensionality, and waveforms were assigned to single units based on operator-defined templates. This method has been used to obtain SUA recordings.

In some recording sessions, up to two units per channel were assigned, while in others, up to four units were designated. An unsorted "noise" or "hash" unit was maintained for each channel to include events that crossed the threshold but did not match any templates. This process resulted in 480units for sessions selected of Indy.

### 5.1.2 Session Selection and Preprocessing (MUA and SBP)

For this study, we focused on Indy and selected 26 sessions for analysis. These sessions were drawn from an initial set of 30 that, in addition to SUA, contained also raw electrocorticography (ECoG) data, which was crucial for extracting Spiking Band Power (SBP), a novel preprocessing method developed to analyze neural signals without relying on traditional spike sorting and detection. The already extracted single-unit activity (SUA) signals were retained, while multi-unit activity (MUA) signals were obtained by aggregating the 480 resulting SUA units back into 96 channels.

We concentrated on Indy and these 30 sessions because raw data was only available for this subset. However, four sessions were excluded from the analysis due to specific considerations:

- Session 'indy_20160630_01': This session did not employ unipolar referencing during data acquisition, which is critical for eliminating power line noise. Attempting to apply unipolar referencing post hoc would not yield results consistent with the other sessions, potentially compromising data integrity.
- Sessions 'indy_20161013_03', 'indy_20161206_02', 'indy_20170124_01': These sessions were characterized by excessive noise levels, rendering the models incapable of performing reliable inference (CC $\approx 0$). The performance drop may be due to issues with the chronic implantation hardware, increased noise or procedural inconsistencies between sessions. Unfortunately, we do not have visibility of the specific procedures performed to record these sessions and the possible consequences. Exclude those was the simplest solution.

By excluding these four sessions, we ensured that the dataset comprised high-quality recordings suitable for our analysis. The final selection of 26 sessions allowed for consistent application of the SBP method across all data.

Before applying the decoding algorithms, several further preprocessing steps were undertaken to ensure data quality and consistency. Discrete derivatives where applied to the kinematic data to downsample and align them with the neural data binning, facilitating synchronized analysis. SBP, MUA and SUA where extracted from the selected sessions and then formatted into the final dataset, as specified in Section 4.2. Following parameters specified in Table 4.1.

### 5.1.3 Behavioral Task

The monkeys were trained to perform continuous reaching movements toward visual targets within a virtual reality environment. Indy used his left arm, while Loco used his right arm. Visual cues were projected onto a mirror, creating the illusion that they appeared in the plane of the reaching hand. An opaque barrier prevented the monkeys from seeing their actual arms, enhancing the reliance on proprioceptive feedback.

The workspace was defined in the horizontal (transverse) plane just below shoulder level. Coordinate axes were established such that the intersection of the transverse and lateral planes defined the x-axis (with positive values to the right of the subject), and the intersection of the transverse and sagittal planes defined the y-axis (with positive values rostral to the subject).

Targets were presented either in an 8-by-8 square grid or an 8-by-17 rectangular grid for Indy, with an inter-target spacing of 15 mm. Each target was a circle with a visual radius of 5 mm. To successfully acquire a target, the monkey had to position the fingertip within a square acceptance zone measuring 7.5 mm by 7.5 mm, centered on the target, and maintain that position for 450 ms. The acceptance zones of adjacent targets were non-overlapping but directly adjacent to each other.

After a target was acquired, a new target was immediately presented without any inter-trial interval, although a 200 ms “lockout interval” followed target acquisition during which no new target could be acquired. In most sessions, targets were selected randomly with replacement from the set of possible locations, meaning the same target could potentially be presented consecutively. In some sessions, the current target was excluded from immediate reselection to prevent repetition.

Fingertip position was monitored using a six-axis electromagnetic position sensor (Polhemus Liberty, Colchester, VT) operating at 250 Hz. The recorded position data were non-causally low-pass filtered using a fourth-order Butterworth filter with a cutoff frequency of 10 Hz to minimize sensor noise. Velocity were calculated by discretely differentiating the x and y-axis position data.

## 5.2 Metrics

For all the experiments, two principal metrics are used, to evaluate the quality of predictions respect to the ground truth: Pearson Correlation Coefficient (CC) and Root Mean Square Error (RMSE).

### 5.2.1 Pearson Correlation Coefficient (CC)

The Pearson correlation coefficient is a statistical measure that quantifies the strength and direction of the linear relationship between two continuous variables. It is commonly used in data analysis to determine how closely two variables are related. The coefficient, typically denoted by $r$, ranges from -1 (perfect negative linear correlation) to 1 (perfect positive linear correlation). 0 means that there is no correlation.

Pearson correlation assumes the data is normally distributed and measures only linear associations. It is widely used in fields like statistics, machine learning, and finance to evaluate relationships between variables.

The formula for the Pearson correlation coefficient is:

$$\text{CC} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Where: - $x_i$ and $y_i$ are predicted and true velocities, at instant $i$. - $\bar{x}$ and $\bar{y}$ are the means of the $x$ and $y$ data points, respectively. - $n$ is the number of time steps.

This measures the strength and direction of the linear relationship between two variables, in our case $x_i$ are predictions, and $y_i$ the real velocities

### 5.2.2 Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) is a commonly used metric to measure the accuracy of a model's predictions, particularly in regression tasks. It calculates the average magnitude of the errors between predicted values and actual values, giving higher weight to larger errors due to the squaring of differences. RMSE is particularly useful when you want to assess how well a model fits the data, as it indicates how close the predicted values are to the actual ones.

The RMSE is always non-negative, and a lower RMSE value indicates a better fit of the model. Since RMSE has the same units as the dependent variable, it provides an intuitive sense of the prediction error.

The RMSE is calculated using the formula:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2}$$

Where $n$ is the number of time steps, $y_i$ represents the actual values, and $x_i$ represents the predicted values.

### 5.2.3 Why We Used Both RMSE and CC

In evaluating the performance of a predictive model, it is crucial to assess both the accuracy of the predictions and the strength of the relationship between the predicted and actual values. To achieve this, we use two complementary metrics: the Root Mean Square Error (RMSE) and the Pearson correlation coefficient.

RMSE provides a direct measure of how far the model's predictions deviate from the actual values. This metric focuses on the magnitude of errors and highlights large discrepancies by penalizing them more heavily. A lower RMSE indicates better prediction accuracy, with zero representing a perfect match between predictions and actual values. However, RMSE alone captures only the overall error.

On the other hand, CC measures the strength and direction of the linear relationship between the predicted and actual values. While RMSE focuses on prediction error, Pearson correlation evaluates how well the predicted and actual values follow the same trend. By using both RMSE and the Pearson correlation coefficient, we gain a more comprehensive understanding of model performance.

#### A Dummy Example

In Figure 5.2 we can see a simple example to show how the two metrics behave in two different scenarios. In the first case we want to confront the blue signal -the original one- with the orange one, identical but for a single point, shifted by 12 units. In this case RMSE = 1.2 and CC = 0.690.

In the second case we want to confront the blue signal with the green one, which retains the same relationship between points, but are all shifted by 8 units. In this case RMSE = 8.0 and CC $\approx$ 1.

In the first case CC is significantly smaller, because there is an inconsistency when dealing with the relationship between points. In the second case CC is almost 1, indicating a perfect correlation, even if all point are far from the real values. That's because the trend is perfectly preserved. If we focus on RMSE we can observe the exact opposite: just one point is a very little change for the overall signal, but if all points shift, that's considerably more.

## 5.3 Single-Session Evaluation of SNN across MUA, SUA and SBP

We first tested our SNN over the 4 preprocessing methods (MUA, SUA, SBP1000, SBP2000). To build our model we used Python library SnnTorch presented here

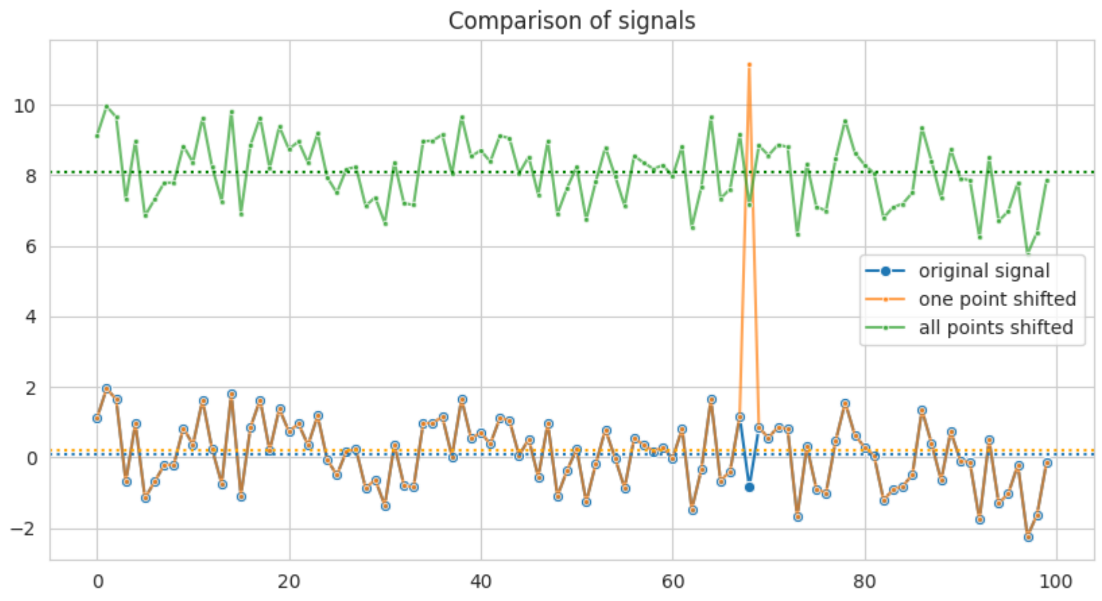


**Figure 5.2:** Example plot of three signals for CC and RMSE comparison.

[34]. More details are present in the models section of the thesis 4.6.

We analysed the performances of our model according to our metrics (CC and RMSE) across 4 different preprocessing methods on a single session: *indy_20160622_01*. We chose this one because it is the first to have both processed (SUA) and raw data available (needed to extract SBP ). Later, we selected the most promising preprocessing method, and we investigated the behaviour of the net: the spiking dynamics, the loss evolution on training, and the evolution of CC and RMSE on the test set after each training epoch.

As general preprocessing steps (after the extraction of MUA, SUA, SBP1000 and SBP2000) we first downsampled the resulting dataset to 32 ms timesteps (by summing together original samples, distant 4ms each), then folded it in windows of 10 timesteps (320 ms per window), with 9 overlapping samples. This is better described in Section 4.3. Later, we divide the resulting dataset in two subsets: Training Set (TRS) which is the 80% of the original dataset, and Test Set (TSS) which is the remaining 20%. Finally, we center and standardize (z-score) the TRS and then TSS, using std and mean obtained by TRS.

### 5.3.1 Comparison between preprocessing methods

At first, we compared CC and RMSE metrics over the 4 preprocessing methods. To do so, we preprocessed the session according to the 4 methods, then we followed the steps described above. Afterwards, we could proceed training on TRS and evaluating on TSS the model for all the 4 methods, and to ensure a more stable result, we repeat the experiment for 10 times over each method (total of 40 times).

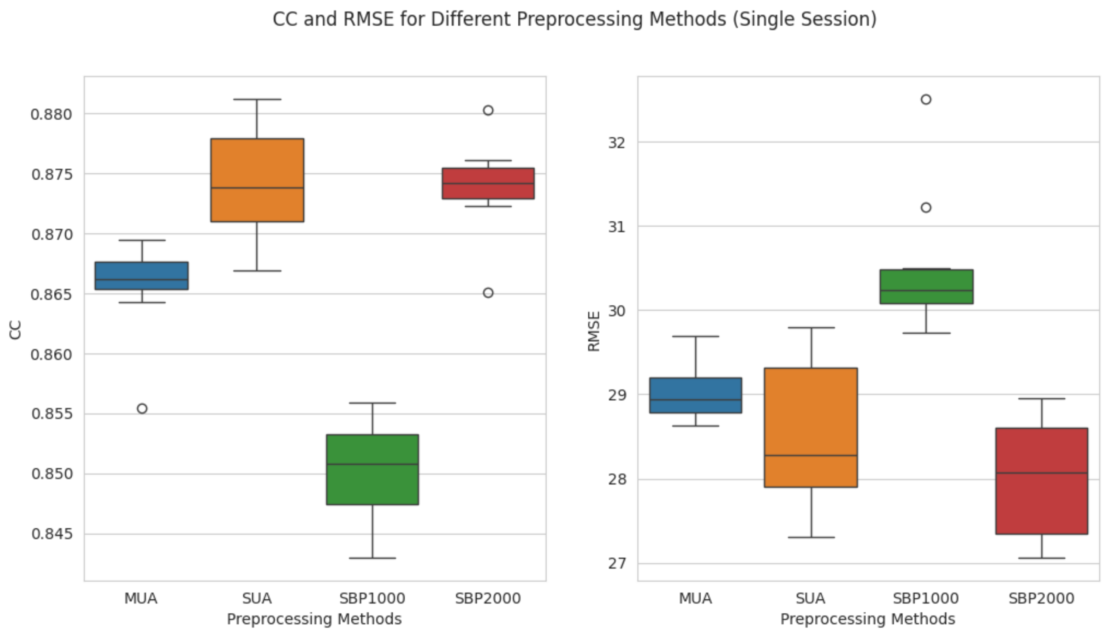


**Figure 5.3:** Single session comparison between MUA, SUA, SBP1000 and SBP2000.

From the results obtained and shown in Figure 5.3, we concluded that the most promising method is SBP2000, only SUA achieved comparable results. We eventually chose SBP because SUA theoretically needs more resources both during preprocessing and training, having 480 channels as input of the model, which translates in 480 input neurons, instead of just 96. A strategy could be to compute a dimensionality reduction, but that would end up in rising both algorithm complexity and resources demand, while potentially decrease prediction quality. For all of these reasons, from now on we decided to continue the single-session evaluation on SBP2000.

To conclude this comparison, in Figure 5.4 we can see the evolution of the prediction quality obtained during training for the different preprocessing methods. The model in this case was trained for 24 epochs on TRS, and at the end of each epoch we measured the accuracy reached by evaluating it on TSS. This experiment was performed one time for each preprocessing method. We can notice the superiority of SBP2000 and how, with all the methods, the model reached stability, and did not overfit. This was further proved by confronting the curve of the loss on both training and test set over the epochs.

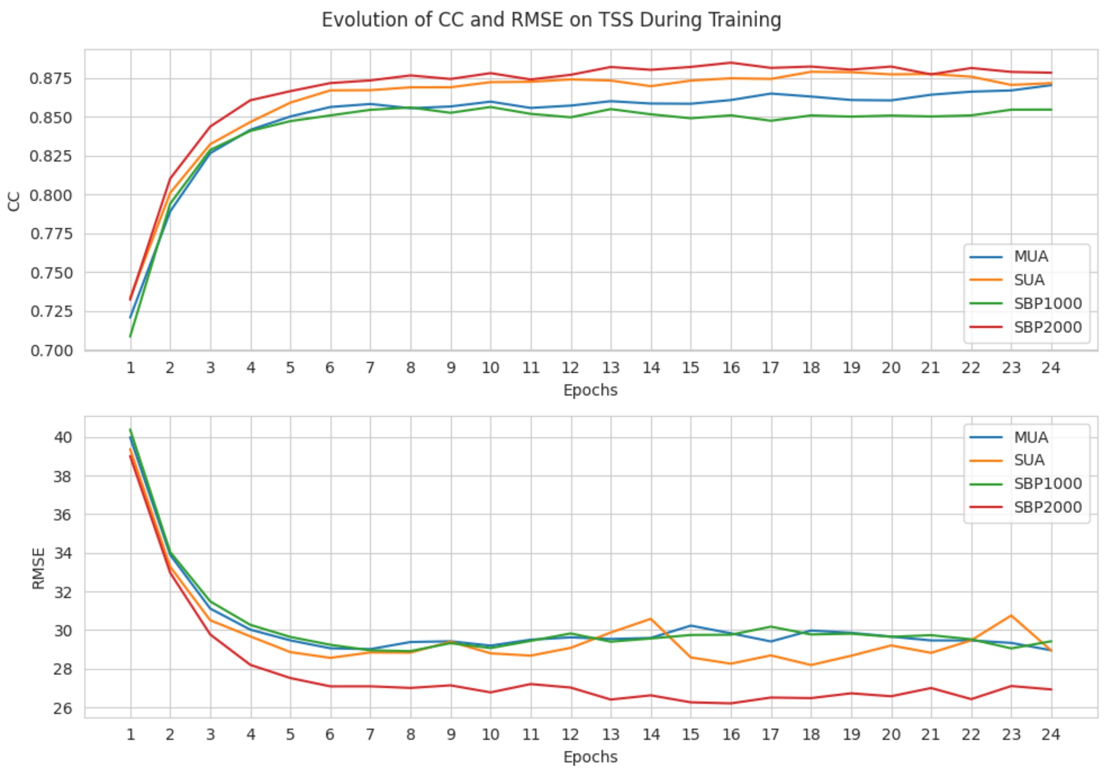


**Figure 5.4:** Evolution of CC and RMSE over epochs while evaluating on TSS and training on TRS.

## 5.3.2 Spiking activity analysis

To show the Spiking activity of our model, we tried with all the different preprocessing methods, and all the results where quite similar, so we decided to show the graphs for the model trained with SBP2000. The spikes are the result of the inference on TRS (while there are no significant difference with TSS). Figure 5.5 shows how many neurons (y-axis) fire with a given frequency (x-axis).

One important aspect when training a Spiking Neural Network (SNN) is evaluating dead neurons and reasons why. A neuron is considered "dead" if, during inference, it never fires. This implies that the weights and biases have been calibrated in such a way that the neuron remains silent all the time, which ultimately leads to a potential loss of information capability, and a waste of computational resources. Furthermore, once a neuron becomes inactive during training, it is often very difficult to reactivate it. This is because our training process does not update the weights and biases of dead neurons, as they no longer emit spikes and are consequently excluded from the backpropagation process.

However, a model able to retain good prediction quality and characterised by

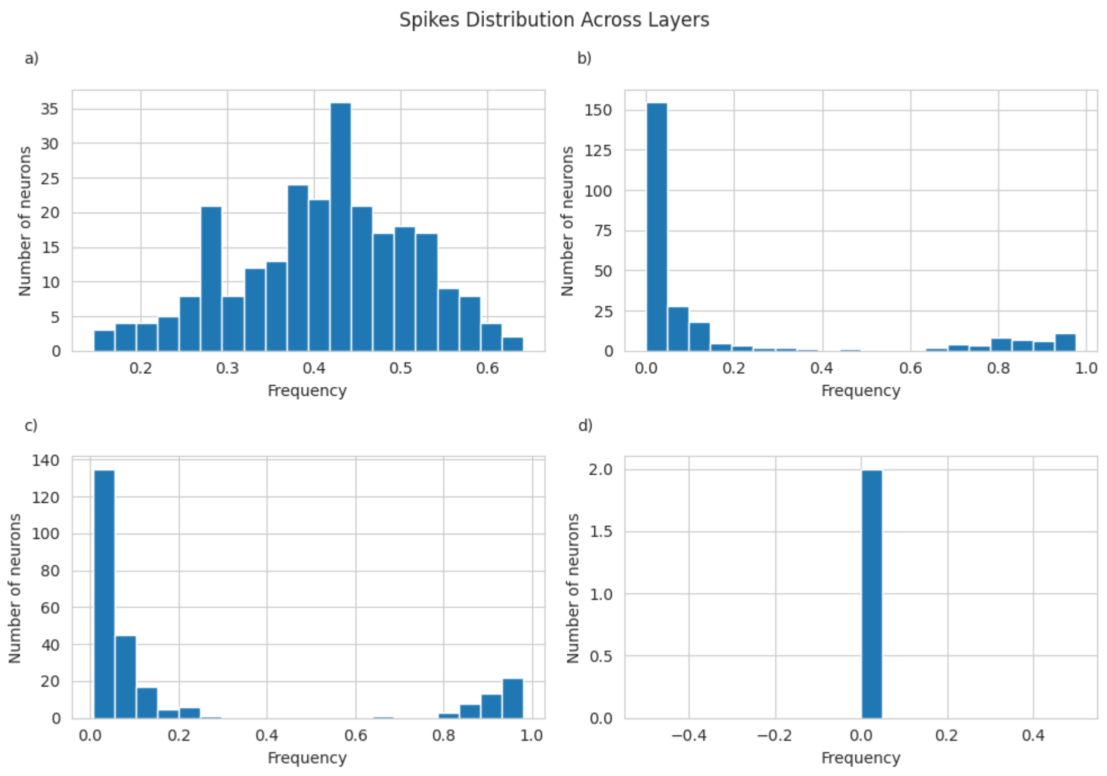


**Figure 5.5:** (a) First hidden layer. (b) Second hidden layer. (c) Third. (d) Output non spiking layer (2 LI-Neurons).

low spiking frequency combined with a moderate amount of silent neurons can also be a very desirable outcome. As a matter of fact, this could also mean less memory movements and more sparse calculations, ultimately increasing the efficiency of the model [27, 33, 34]. This last was our case, confirmed by scores of both CC and RMSE.

Furthermore, we want to spend few words about the first hidden layer performance. It showed almost no dead neurons, exhibiting strong neuronal activity. We chose to display the graph for SBP2000 also because of this particular layer, as it is responsible for encoding the continuous signals of the 96 channels into discrete spikes, we will later call it Encoding Trainable Layer (TEL) for this reason. This encoding method is quite a novelty, as it differs from classical encoding techniques. In the following experiments we tried to better understand and explain the implications.

Finally, we can confirm that the final layer behaved exactly as expected: both of its neurons never fired. This is intentional, as they are LI neurons, designed to accumulate membrane potential without resetting, which is used to predict velocity.

### 5.3.3 Training Loss analysis

Loss during training represents the model's performance, quantifying how far its predictions are from the actual target values. As training progresses, the goal is to minimize the loss, indicating that the model is learning and improving.

Typically, in well-designed models, loss decreases over time, showing that the network is adjusting its parameters (weights and biases) to better capture the underlying patterns in the data. A sharp decline in loss early in training suggests rapid learning, while a plateau or slow decline may indicate that the model is approaching its optimal state or encountering challenges like overfitting or underfitting.

Monitoring the loss trend is crucial to ensure that the model is learning effectively. If the loss doesn't decrease or even increases, it may signal issues like improper learning rates, vanishing gradients, or poor model architecture.

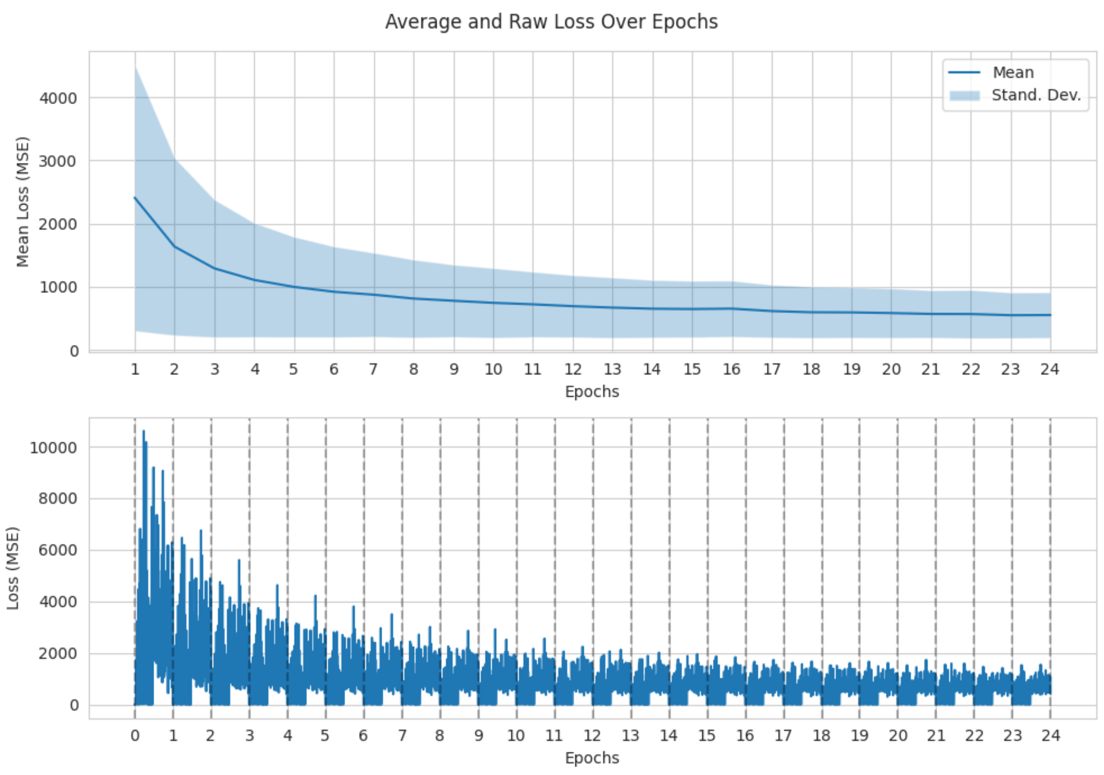


**Figure 5.6:** Training loss mean and std for each session (above). Real training loss batch by batch (below). Loss is MSE, data is preprocessed with SBP.

In our case, the training loss is decreasing as expected with each epoch; however, the standard deviation is higher than expected. We can derive an intuitive explanation for this phenomenon: in Figure 5.6 (below) we can see the loss calculated batch by batch, where the vertical lines highlight the epochs. It is evident a repetitive pattern, where the first half of each epoch exhibits notably low peaks. To better understand the cause, let's focus on the last epoch.

In Figure 5.6 we can see the average trend of the loss (above) during training, and the real loss (below), calculated batch per batch. The vertical lines are there to highlight the start and end of each epoch, showing a repetitive pattern.

In Figure 5.7, we compare velocity with loss, and it becomes evident that speed significantly influences the loss value. The plot shows the clear correlation between a zero-speed sample and a very low loss. For clarity, we plotted only one direction, but the same happened with both.

The current session *(indy_20160622_01)* contains many static samples in the first half, where the speed is zero for both directions of the 2D velocity of the fingers. This peculiarity however is not problematic, as the model successfully trains and makes accurate predictions regardless.

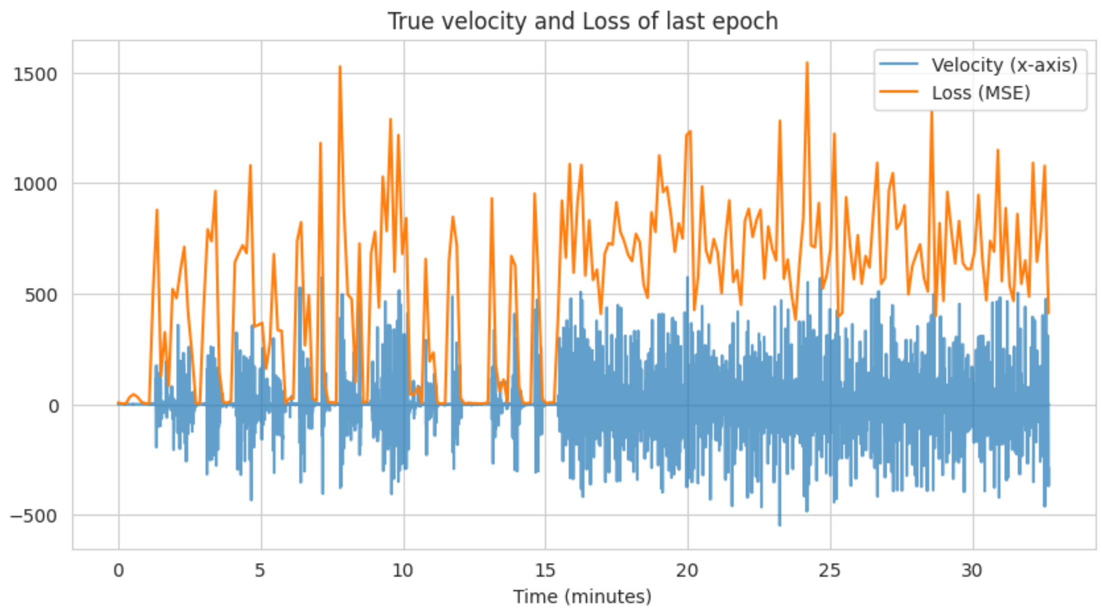


**Figure 5.7:** Comparison between training loss and velocity of the finger.

## 5.4 Unsupervised adaptations: Online Normalisation and PCA Alignment

From this session on, we present the results of experiments conducted on MUA and SBP (SBP2000) neural data over 26 sessions, spanning 223 days. The reference session (RS), from which the model was trained, is the oldest available session,

| | **CC** | **RMSE** |
|---|---|---|
| MUA | $0.8656 \pm 0.0037$ | $29.0276 \pm 0.3305$ |
| SUA | $0.8743 \pm 0.0043$ | $28.4999 \pm 0.8352$ |
| SBP1000 | $0.8502 \pm 0.0040$ | $30.4996 \pm 0.7746$ |
| SBP2000 | $0.8739 \pm 0.0036$ | $28.0236 \pm 0.6778$ |

**Table 5.1:** CC and RMSE for MUA, SUA, SBP1000 and SBP2000 (mean and std).

*indy_20160622_01*, then the behaviour of decoder and preprocessing methods was tested on the following 25 inference sessions (ISs). The primary goal was to assess how different data alignment strategies could affect the long-term performance of our spiking neural network.

In the following experiments, we explored unsupervised methods to align the 25 ISs to the RS. All parameters calculated online (during inference) were extracted from the first 2 minutes of each session, while the performance evaluation metrics (CC and RMSE) were computed on the remaining data. For offline counterexamples, the parameters are fixed, calculated from the first 80% of the RS and the model was always trained on 80% of the RS too.

For simplicity, from now on we will consider normalisation (which include centering by mean subtraction) as a linear transformation, even if translations can be classified as either linear or affine depending on the theoretical framework. We will refer to reference session and inference sessions data also as $X_{\text{ref}}$ and $X_{\text{inf}}$, while the mean and standard deviation will be $\mu_{\text{ref}}$ and $\sigma_{\text{ref}}$ if calculated offline on the first 80% of RS, and $\mu_{\text{inf}}$ and $\sigma_{\text{inf}}$ if calculated on the first 2 minutes of the current inference session.

During the following dissertation, we decided to focus only on MUA and SBP2000 (simply called SBP hereafter). SUA was excluded from this analysis for practical reasons: the complexity of extracting SUA from raw data (spike detection and sorting), coupled with the changes required to the model's structure (480 input neurons instead of 96), would result in a significant loss of computational efficiency while making the preprocessing step unnecessary more complex. Moreover, from results of single session analysis, it seems to not obtain better results than SBP2000.

### 5.4.1 Experiment 0: No Alignment

In this preliminary experiment, after training the model on RS, we evaluated its performances on both MUA and SBP data across ISs without applying any prior adaptation, nor supervised or unsupervised. This setup allowed us to observe the model's ability to generalize across sessions when no adjustments were made to account for session-to-session variability.

This experiment captured a very important difference between MUA and SBP, and showed how the model behaves when dealing with continuous SBP data, highlighting the necessity for applying further transformations.

Figure 5.8, shows us the scatter plots and the linear regression lines for CC and RMSE across the 25 ISs. MUA data, extracted through threshold crossing technique —and so intrinsically scaled and centered— started from a decent $CC \approx 0.65$, and then linearly declined through time, showing a consistent loss in performance, that we tried to compensate in the next experiments. On the other hand, the behaviour of our model fed with SBP was way more critical: it was not able to generalise at all on ISs. Its regression line had no decrease or increase trend, while steadily showing a $CC \approx 0$.

#### The Role of The First Hidden Layer

We highlighted this critical result to discuss the role of the first hidden layer in interacting with continuous input data, like SBP, and to explain why we refer to it as a trainable encoding layer (TEL). As a matter of fact, we designed our decoder pipeline to extract the SBP data with minimal operations from raw neural signals, without explicitly encoding spikes, contrary to methods like MUA or SUA. This architectural choice left the responsibility of encoding spikes —and learning how to do it— to our model, in particular to the first hidden layer, like explained in Section 2.5.9. This came with many advantages, as we will explain later in this section.

Without centering nor scaling input channels of ISs, TEL failed its encoding duty, leading to a model that could not predict velocity at all. This was evident from the correlation coefficient (CC), which remained close to zero for all the sessions. Our supposition for this outcome was that our method would have required data to be normalised, before being used in inference. That's because TEL may need a consistently similar dynamic range to allow its 256 LIF-neurons to decay and rise above the spiking threshold $V_{th}$ and translate continuous inputs into spikes in a coherent way. To better explain this concept, in Figure 5.9 we plotted the different response of a LIF neuron when the same signal, a simple sine function, is normalised or not.

### 5.4.2 Experiment 1: Impact of Online Vs. Offline Normalisation

In this experiment our aim was to investigates the behaviour of the decoder when different kind of input signals, MUA and SBP, were normalised before being fed to the net. Both online and offline normalisation has been tested on the 25 ISs, While the model was first trained on the 80% of the RS, preprocessed and normalised

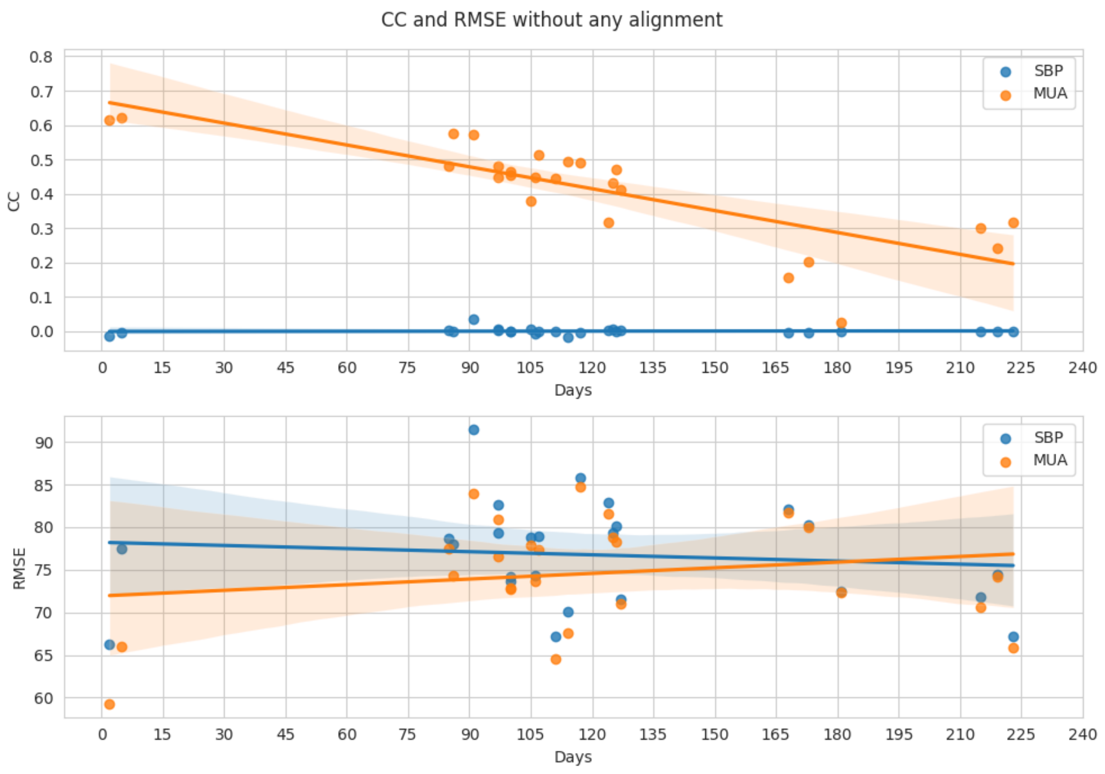


**Figure 5.8:** Linear regression analysis for SBP and MUA without alignment (CC and RMSE).

accordingly. In Figure 5.10 we plotted a first summary of the 4 tested combinations: offline and online normalised MUA and SBP, for both CC and RMSE. We decided to plot scatterplots and linear regression lines along with their respective confidence intervals (95%). This approach allowed us to still display individual sessions while focusing on overall trends. This is particularly important in our study, as we are interested in understanding the long-term behavior of the network.

When working with $\mu_{\text{ref}}$ and $\sigma_{\text{ref}}$, the primary goal was to assess how the performance metrics evolved compared to the previous experiment. We observed a marked change in the model's behavior using SBP. The RMSE trend is not very informative, as its mean was too large to provide meaningful insights. However, the correlation coefficient (CC) consistently remained well above zero, indicating that despite its limitations offline normalization enabled TEL to better encode input signals. Meanwhile, the metrics related to MUA only showed marginal improvements.

After the offline setup, in which centering and scaling can be considered standard preprocessing steps, as they are static for all sessions, and always referring to the

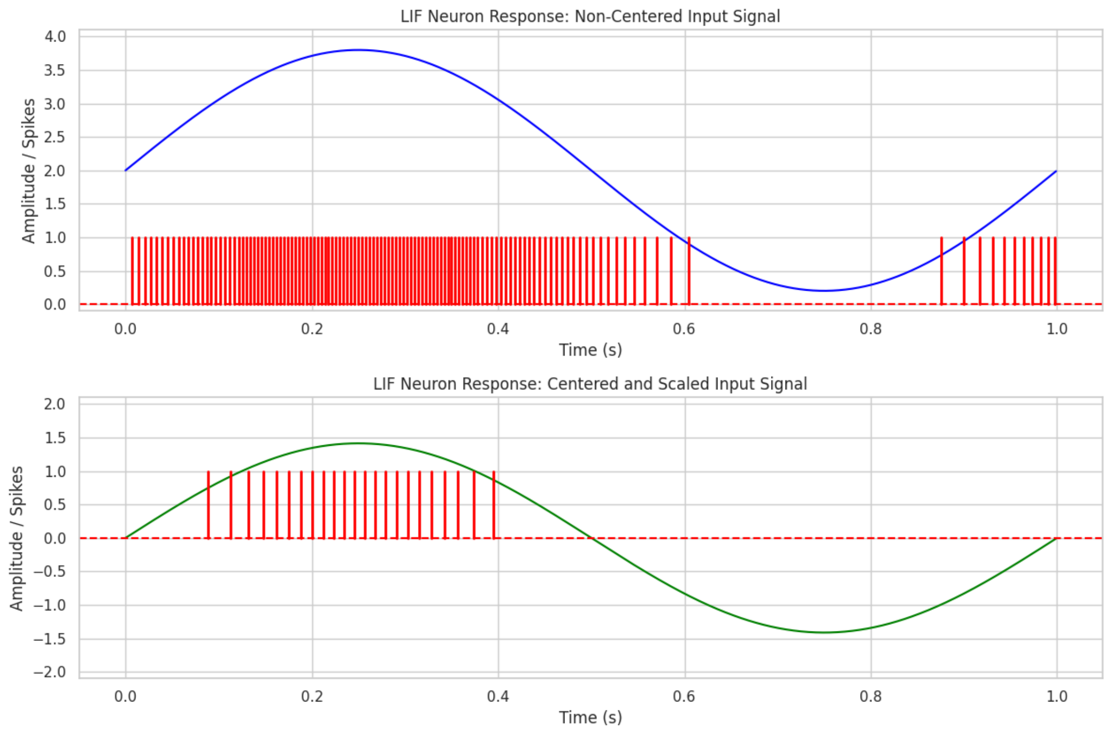


**Figure 5.9:** The different LIF neuron response to a signal with and without normalisation.

RS, we tested online normalisation, using $\mu_{\text{inf}}$ and $\sigma_{\text{inf}}$ calculate on the first 2 minutes of each IS. The primary objective was to quantify the improvement given by this first and very simple unsupervised adaptation technique.

This is the simple equation used during online normalisation (ON):

$$X_{\text{inf, norm}} = \frac{X_{\text{inf}} - \mu_{\text{inf}}}{\sigma_{\text{inf}}} \tag{5.1}$$

We observed a significant improvement in prediction quality for both MUA and SBP, marked by a substantial increase in CC and a clear reduction in RMSE, in both the overall trends and individual sessions. To examine the average metric scores accross the 25 ISs, we can refer to the values presented in Tables 5.3 and 5.4. SBP achieved the best performance, reflected by a CC of approximately 0.75 and a RMSE of about 55.25, with a CC increase of approximately 121% and a RMSE reduction of around 30%. Nevertheless, the groundbreaking aspect of this results lies in the trend of the CC for SBP inference on ISs.

To assess the trend of CC for SBP, we conducted a linear regression analysis. Despite acknowledging the limited statistical significance of our method, primarily

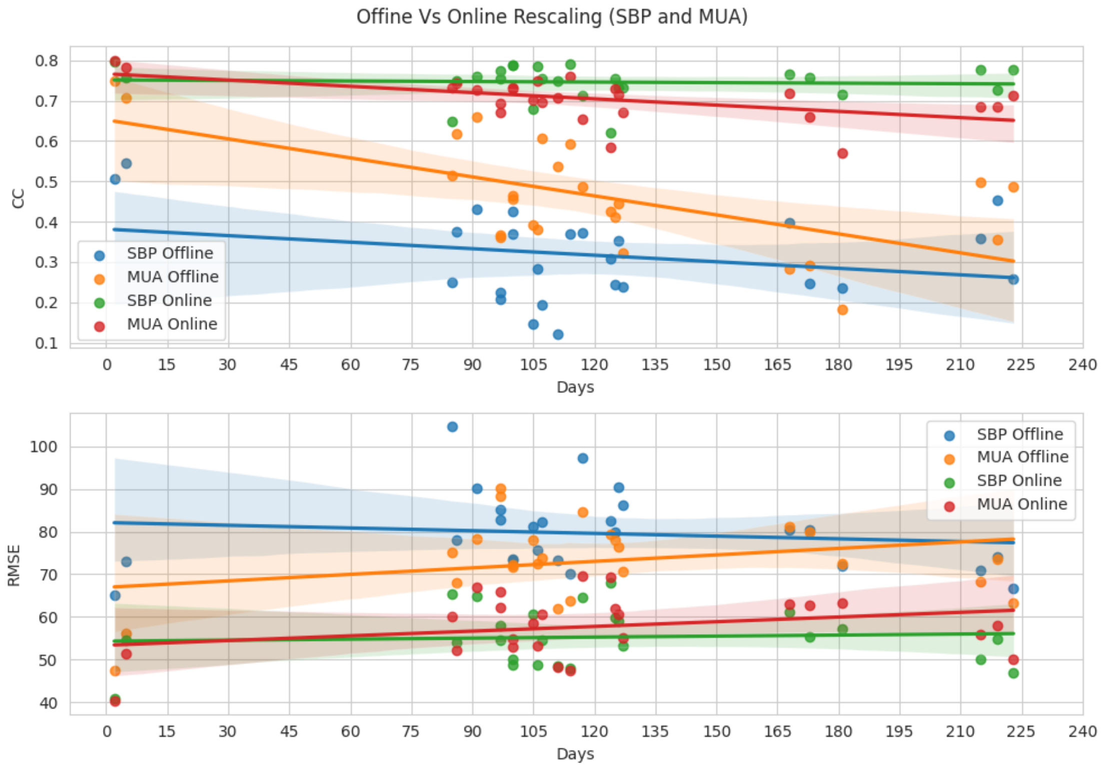


**Figure 5.10:** Linear regression analysis for SBP and MUA with online or offline normalisation (CC and RMSE).

due to a very low R$^2$ value when analyzing session data over time—likely affected by various uncontrollable factors such as noise and other sources of variability not accounted for by linear regression—we chose to present the slope and p-value due to the simplicity of the approach. This allows us to explore, though not definitively prove, potential trends over time.

In this case, the slope of CC remained consistently close to zero, suggesting no significant decline in prediction quality over time, while the p-value was around 0.8, well above the conventional 0.05 threshold. This supports the null hypothesis for the slope parameter, confirming the stability of the SBP model's performance. In contrast, the CC results for the MUA data revealed that, despite the substantial performance boost thanks to ON, MUA performance exhibited a noticeable decline over time. This was confirmed by the p-value (approximately 0.005) of its negative slope, that would lead to the rejection of the null hypothesis for the parameter.

### Statistical Comparison between MUA and SBP

To better capture differences between MUA and SBP when applying ON, we plotted again in Figure 5.11 the two preprocessing method trends, along with the distributions of sessions over performances measured with both CC and RMSE. It shows even more clearly the consistent superiority of SBP respecting to MUA in terms of both mean performances and stability over time.

Additionally, we chose to provide a more statistical validation of the above statement, by testing the null hypothesis: MUA $\approx$ SBP. We employed both the Wilcoxon signed-rank test and the paired t-test to ensure a comprehensive analysis of the statistical significance of the differences. The Wilcoxon test, being non-parametric, does not assume normal distribution, making it suitable for data that may not meet this assumption. In contrast, the paired t-test, which assumes normality, is more powerful when this condition holds. By using both tests, we increased the robustness of our findings, ensuring valid results regardless of the data distribution. This translated into testing $\mu_{\text{MUA}} = \mu_{\text{SBP}}$ for t-test, and $\text{M}_{\text{MUA}} = \text{M}_{\text{SBP}}$ (Median) for Wilcoxon test. For this purpose, Table 5.2 offers further insights.

The p-values obtained were all well below the p-value of 0.05 (statistical significance) for both CC and RMSE. These results confirmed that the difference between performances of SBP and MUA is statistically significant.

| **Test** | **MUA Vs SBP (P-value)** |
|---|---|
| Wilcoxon test for CC | 0.0002 |
| t-test for CC | 0.0001 |
| Wilcoxon test for RMSE | 0.0023 |
| t-test for RMSE | 0.0016 |

**Table 5.2:** Statistical testing results of the Null Hp: MUA $\approx$ SBP, for both CC and RMSE.

### Long-Term Stability: a Synergy between TEL, ON and SBP

The high variability in SBP between sessions—due to physiological changes and noise—demanded continuous adaptation to maintain prediction accuracy. ON helped compensate for these fluctuations. Indeed, both MUA and SBP benefited from this online adaptation, but while the increase in performance of MUA, even if substantial, was not enough to produce a stable trend through time, SBP not only outperformed MUA, but also showed a stable prediction CC across all the 25 ISs ( $\approx$ 220 days).

We believe that the reason why ON's impact was particularly effective lies in

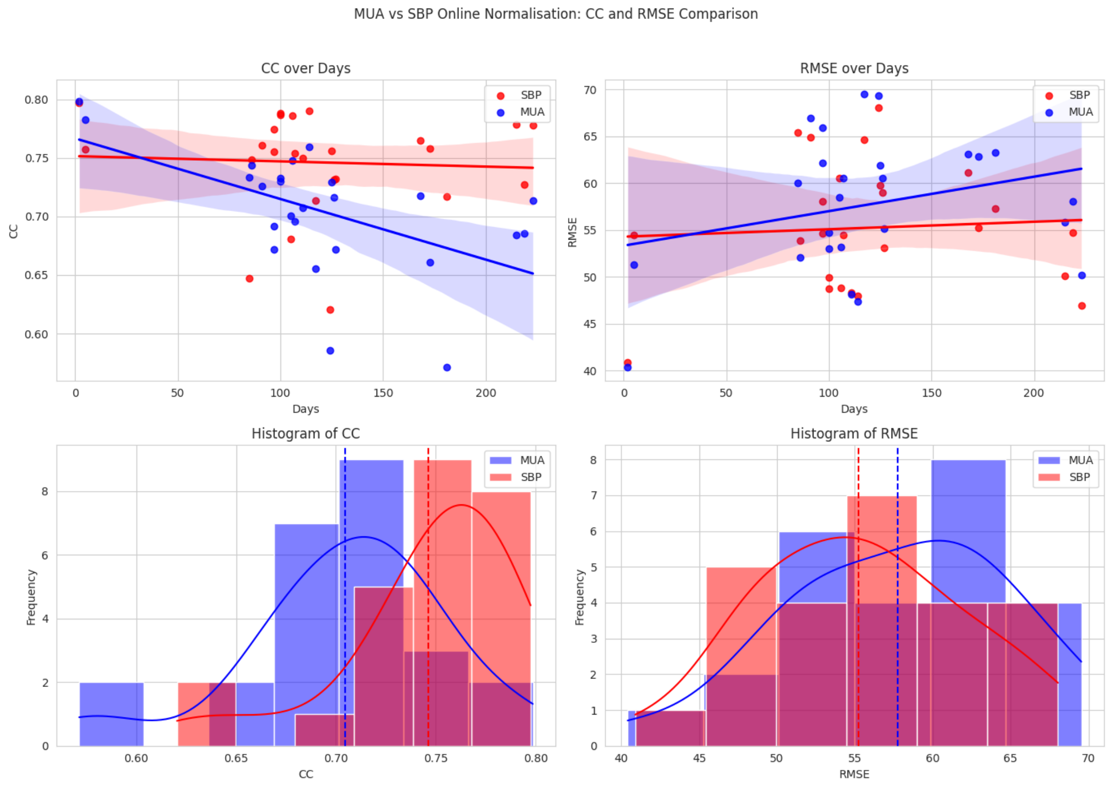


**Figure 5.11:** Linear regression plots (above). Distribution of sessions over metric scores (below). Online normalisation comparison between MUA and SBP (CC and RMSE)

the synergy between the structure of our model, and in particular the presence of TEL and the nature of SBP. This continuous signal is obtained with minimal preprocessing steps, and retains all the information filtered out by static TCR approaches (used with MUA). For this reason it needed TEL to be dynamically encoded in spikes. At the same time, this allowed TEL to be effectively trained on RS, and then to express its whole potential in extracting spikes on the following sessions.

In contrast, MUA data underwent static spike detection through threshold crossing before being processed by the model. This preprocessing step effectively normalized data in a fixed manner, reducing the need for, and the possibility of further adaptations. As a result, the model's ability to respond dynamically to shifts in data is limited for MUA, making the benefits of ON less pronounced.

In summary, the novel application of SBP as a preprocessing method over extended periods of time, combined with ON, proved to yield remarkable performance and, of greater importance, to achieve long-term stability, at least over the spanning time of the inference sessions.

### 5.4.3 Experiment 2: Online Normalisation Vs. Online Centering

After comparing online and offline normalisation in the previous experiment, the focus here is to explore whether rescaling during inference yields better performance than centering alone for maintaining model consistency across sessions.

For each preprocessing method the models was trained on 80% of RS as usual, then tested on ISs. Figure 5.12 shows the results, for both single sessions and trends through time.

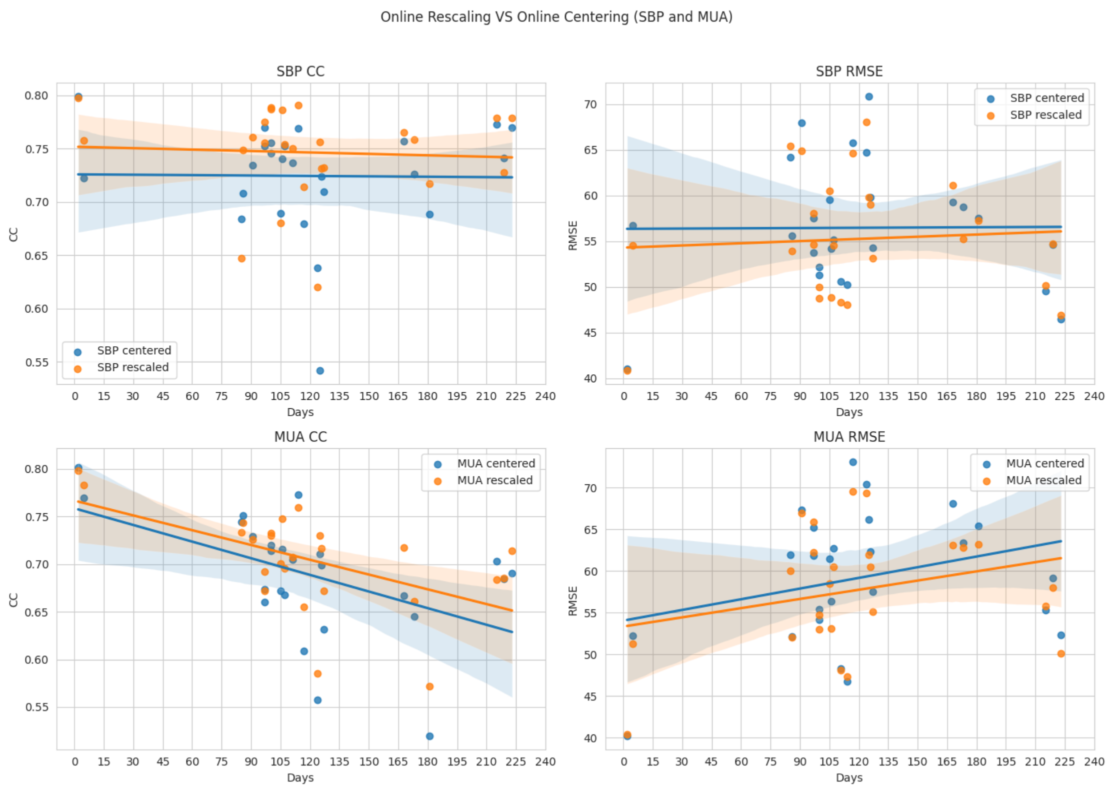


**Figure 5.12:** Linear regression analysis for SBP and MUA. Online normalisation Vs online centering (CC and RMSE).

The experiment revealed that for both SBP and MUA, scaling was important in preserving high prediction quality. The results from Table 5.3 further underscore this assertion, showing the best performance when using scaling and centering together. SBP showed the greatest improvement, with nearly a 3% increase in CC and a 2% reduction in RMSE, underscoring the importance of rescaling during inference. MUA exhibited smaller but consistent improvements, with a change of approximately 2% in both CC and RMSE.

In summary, these findings demonstrated that original ON (centering and scaling) consistently outperformed online centering alone in aligning MUA and SBP data across sessions, suggesting that rescaling in addition to centering is essential for enhancing prediction accuracy across sessions.

From this point on, we have experimented with only SBP. As a matter of fact, the stable behaviour obtained through ON is a crucial achievement, but it still showed lower prediction quality respect to the model trained and evaluated on a single session. For this reason, in the following experiments we have tried different online linear transformations to further increase long-term prediction accuracy.

| **Metrics** | **Offline SBP Norm** | **Online SBP Norm** | **Online SBP Centered** |
|---|---|---|---|
| **CC** | 0.3376 ± 0.1477 | 0.7477 ± 0.0427 | 0.7284 ± 0.0544 |
| **RMSE** | 77.6579 ± 13.0987 | 54.5107 ± 7.4272 | 55.485 ± 8.1905 |
| **Slope CC** | -0.0005 | -0.0 | -0.0 |
| **P-value CC** | 0.1928 | 0.7919 | 0.9507 |
| **Slope RMSE** | -0.0212 | 0.008 | 0.001 |
| **P-value RMSE** | 0.5549 | 0.7581 | 0.9708 |

**Table 5.3:** Performance metrics for SBP (centered and rescaled)

| **Metrics** | **Offline MUA Norm** | **Online MUA Norm** | **Online MUA Centered** |
|---|---|---|---|
| **CC** | 0.4811 ± 0.1583 | 0.7112 ± 0.0598 | 0.6967 ± 0.0735 |
| **RMSE** | 71.1587 ± 13.105 | 56.6054 ± 9.0444 | 57.8689 ± 10.0183 |
| **Slope CC** | -0.0016 | -0.0005 | -0.0006 |
| **P-value CC** | 0.0009 | 0.0049 | 0.0116 |
| **Slope RMSE** | 0.0508 | 0.0368 | 0.0427 |
| **P-value RMSE** | 0.1625 | 0.1811 | 0.1512 |

**Table 5.4:** Performance metrics for MUA (centered and rescaled)

### 5.4.4 Experiment 3: PCA Adaptation: Rotation and Ordering

In this experiment, we focused on a PCA-based rotational alignment strategy: our aim was to work with covariance matrices between sessions and using PCA to extract PCs.. We used them to align the sessions themselves. The goal was to evaluate if the same linear rotation needed to align covariance matrices could be effectively applied to align $X_{\text{inf}}$ to $X_{\text{ref}}$, and increase the model prediction quality across sessions, leading eventually to a better long-term stability. In this first linear transformation, we used this equation:

$$X_{\text{aligned}} = W_{\text{ref}} W_{\text{inf}}^T X_{\text{inf}}$$

Where $X_{\text{aligned}}$ is the current IS rotated like the RS, $W_{\text{ref}}$ is the orthonormal matrix made by PCs of the RS on the columns and $W_{\text{inf}}$ is the orthonormal matrix for the IS, calculated online using the first 2 minutes of the IS. Finally, $X_{\text{inf}}$ is the IS data before alignment. More details can be found in Section 4.4.

The model is always trained on 80% of $X_{\text{ref, norm}}$, and, after each different linear transformation, before inference, each resulting $X_{\text{aligned}}$ is rescaled again, using $\sigma_{\text{aligned}}$ computed on its first 2 minutes, so that each channel presents unit variance, as during training. This is done to ensure consistency between experiments, allowing to always use the same model. Here the two variant of this first PCA-based linear transformation:

1. **Online Normalisation → Rotation**: ISs were normalised using $\mu_{\text{inf}}$ and $\sigma_{\text{inf}}$, then $W_{\text{ref}}$ and $W_{\text{inf}}$ were calculated from 80% of $X_{\text{ref, norm}}$ and the first 2 minutes of $X_{\text{inf, norm}}$. Finally PCA rotation was applied.

2. **Centering → Rotation**: Data was only centered resulting in $X_{\text{ref, cent}}$ and $X_{\text{inf, cent}}$, then $W_{\text{ref}}$ and $W_{\text{inf}}$ were calculated as above, and eventually $X_{\text{inf, cent}}$ was aligned.

The experiment was repeated for all the ISs, both with and without reordering the PCs of $W_{inf}$ according to $W_{ref}$. More details below and in the Section 4.4.3.

The results indicate that the most effective approach is to scale before PCA, followed by reordering PCs. This finding is clearly illustrated in Figure 5.13.

However, the overall prediction quality remained far from satisfactory, especially when compared to the results of simple ON on the same setup. Therefore, this approach seems to not be the most appropriate for unsupervised adaptation.

### To Scale or Not to Scale

The primary objective of this experiment was to assess the quality of predictions made by a model trained on data from RS and applied to data from IS, with the datasets aligned using a PCA-based rotation. Although the brain signals are homogeneous and come from the same source, and thus the features are likely on similar scales, it is still important to consider whether scaling the data could affect the alignment process.

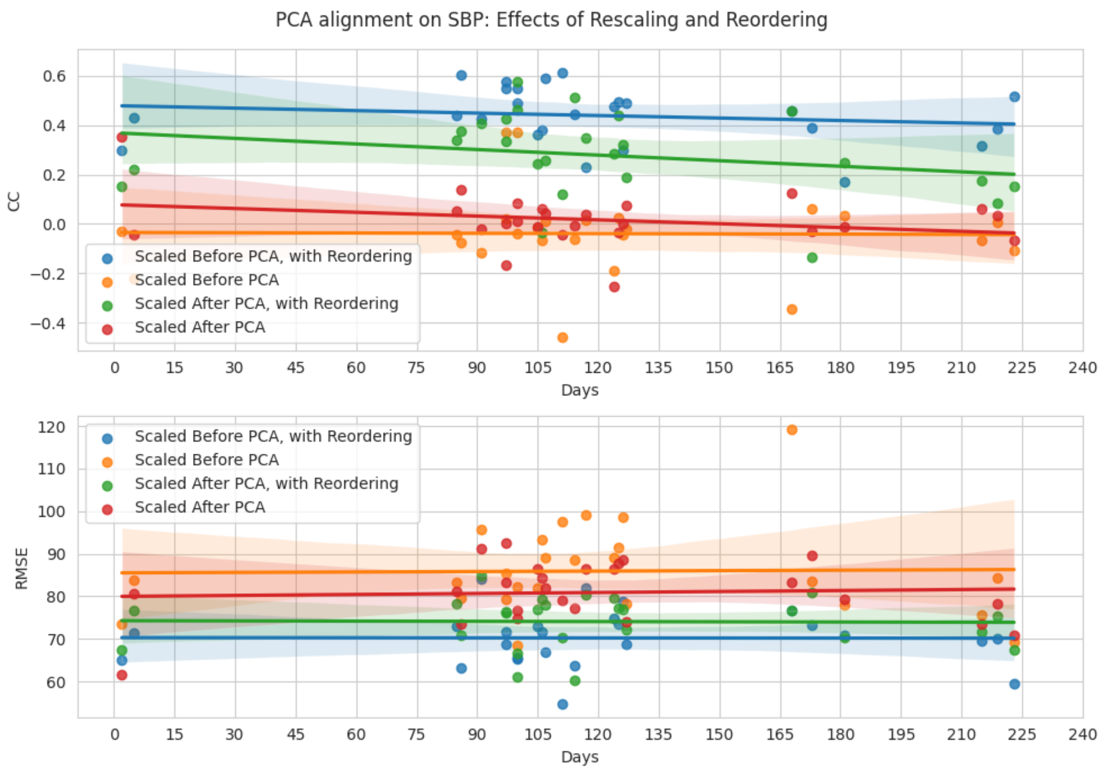


**Figure 5.13:** Linear regression analysis for 4 combinations of normalisation, PCA and reordering on SBP.

When PCA is applied without scaling, the natural variance structure of the data remains intact, with features that have slightly higher variances contributing more to PCs. This could be beneficial if the variance patterns in the brain signals were preserved between sessions, as it should allow the alignment to retain these dominant features. As a result, the model's predictions on the IS may benefit from this natural variance-driven structure, improving its ability to generalize.

On the other hand, scaling the data before PCA ensures that all features contribute equally, preventing small differences in variance from dominating PCs. This could be particularly helpful when aligning datasets across sessions, as it should minimize the impact of any subtle variance differences, leading to a more balanced alignment between the sessions.

Testing both approaches allowed us to explore whether preserving the natural variance or equalizing the feature variances yielded better alignment and, consequently, more accurate predictions on the second session. This comparison is critical for determining which method offers the best model performance and ensures that the alignment process effectively handles the underlying brain signal structure.

### The Role of Reordering PCs

The effect of reordering can be seen in Figure 5.14. We first calculated the similarity matrix $M$ using this equation:

$$M = W_{\text{ref}}^T W_{\text{inf}}$$

Where each value is the dot product between a PC from reference session and a PC from inference session. The dot product of two vectors $\mathbf{u}$ and $\mathbf{v}$ is given by $\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}||\mathbf{v}| \cos(\theta)$, where $\theta$ is the angle between them. Geometrically, it represents how much one vector aligns with the other. If the vectors have unit magnitude, as in our case, the dot product simplifies to $\mathbf{u} \cdot \mathbf{v} = \cos(\theta)$, which directly gives the cosine of the angle between the vectors. We used these values squared as similarity score.

Finally we reordered the PCs of the inference session by sorting the similarity matrix row by row in descending order, and correcting the signs when needed. More details in Section 4.4.3.

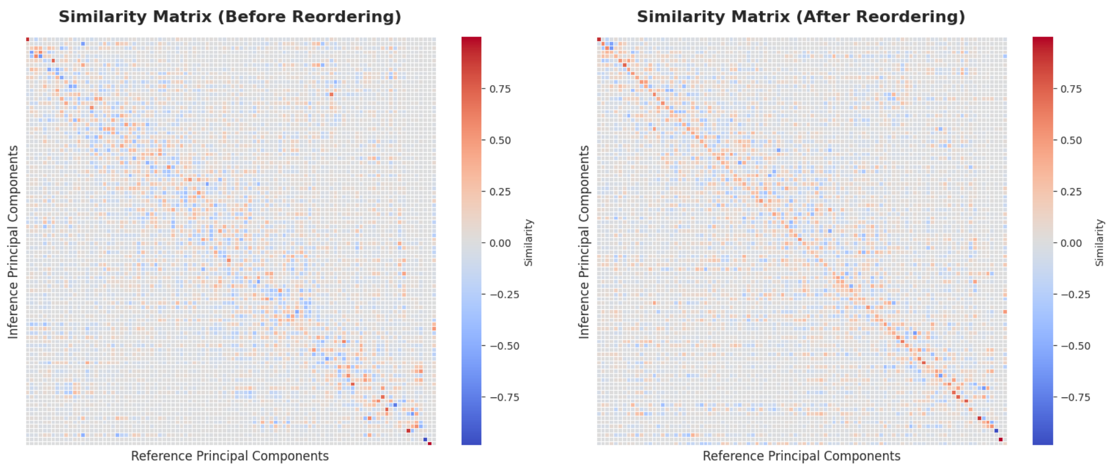


**Figure 5.14:** Similarity matrix of $W_{\text{ref}}$ and $W_{\text{inf}}$, before and after reordering.

### 5.4.5 Experiment 4: PCA Adaptation: Rotation and Scaling with Singular Values

The fourth experiment was our last attempt to improve long-term stability using a more refined method than simply ON. Here we evaluated the effect of using both principal components ($W$) and singular values (square roots of eigenvalues, $\Sigma$) during the alignment process. The equation describing this linear transformation is the following one:

$$X_{\text{aligned}} = W_{\text{ref}}\Sigma_{\text{ref}}\Sigma_{\text{inf}}^{-1}W_{\text{inf}}^{T}X_{\text{inf}}$$

For more details, refer to Section 4.4.

Singular values reflect the standard deviation along each principal component, and using them for scaling should improve alignment by accounting for both rotational and scaling discrepancies across sessions.

As for the previous experiment, the model was trained on 80% of $X_{\text{ref, norm}}$, and after each linear transformation, $X_{\text{inf, aligned}}$ was rescaled using $\sigma_{\text{aligned}}$, calculated from its first 2 minutes, ensuring that each channel maintained unit variance, just like during training. Two alignment methods were tested:

1. **Centering → Rotation & Scaling**: Data is only centered, PCA is applied to both rotate and scale.
2. **Normalisation → Rotation & Scaling**: Data is normalised before PCA rotation and scaling.

In both cases, principal components $W_{inf}$ and singular values $\Sigma_{inf}$ from the inference session were reordered to match the reference session $W_{ref}$ and $\Sigma_{ref}$.

Even after implementing both rotation and scaling, the results were far from being comparable with ON. As we can see from Figure 5.15, the trend is almost stable or just slightly decreasing in time, but the overall performances are substantially worse.

### 5.4.6 Why PCA-based adaptations failed

The experiments aimed to assess various alignment strategies performances across multiple sessions. In the end, Online normalization (ON) consistently yielded superior results on our setup, achieving our goal for long-term stability through unsupervised adaptation, without retraining the model.

This method is particularly effective with data preprocessed with spiking band power (SBP), and encoded thourgh a trainable encoding layer (TEL) made by LIF-neurons. Offline normalization, by contrast, led to poor results, demonstrating that real-time adjustments are essential for long-term model stability. All subsequent experiments, using PCA-based linear transformations, underperformed compared to simple online normalization.

Thus, the clear conclusion from all these methods is that ON with SBP and TEL is the most effective approach to handle session variability and maintain model accuracy over time. We can see the results in details in Figure 5.16, and in Table 5.5.

We can now confidently assert that the results presented in the literature, discussed in 2.6.2, which were derived from sessions involving identical movements

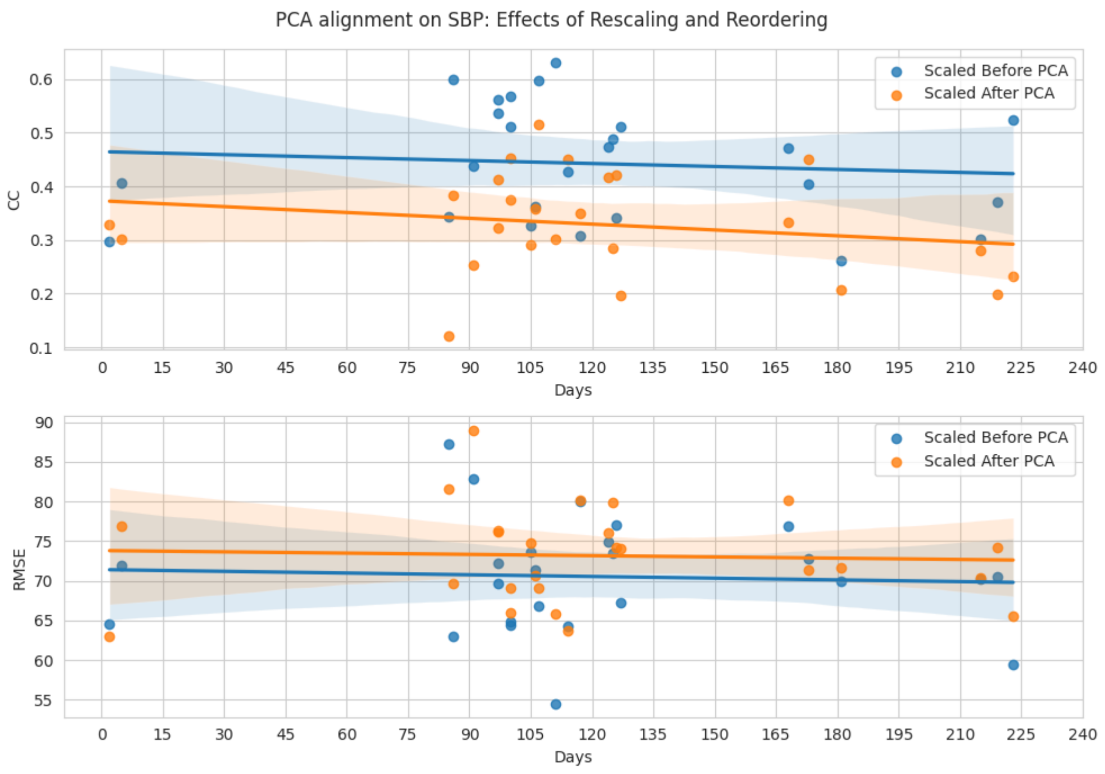


**Figure 5.15:** Linear regression analysis for PCA adaptation (rotation and scaling) on SBP

performed in the same sequence, cannot be generalized to broader unsupervised scenarios like ours. As a matter of fact, we opted not to manually select the same movements across different sessions, as that would have introduced a supervised approach, out from our research scope and already proven to be successful. However, we hope that our experiments and methodologies can serve as a valuable resource for aligning datasets with a more similar structure, more coherent with literature.

## 5.5 Supervised Adaptation for Long-Term Stability

In this section, we explored supervised adaptation strategies to enhance the long-term stability of our Spiking Neural Network (SNN)-based decoder across multiple sessions spanning 223 days. The primary objective was to assess how different retraining approaches could affect the decoder's performance in predicting 2D hand velocity from SBP signals. We focused on the correlation coefficient (CC)

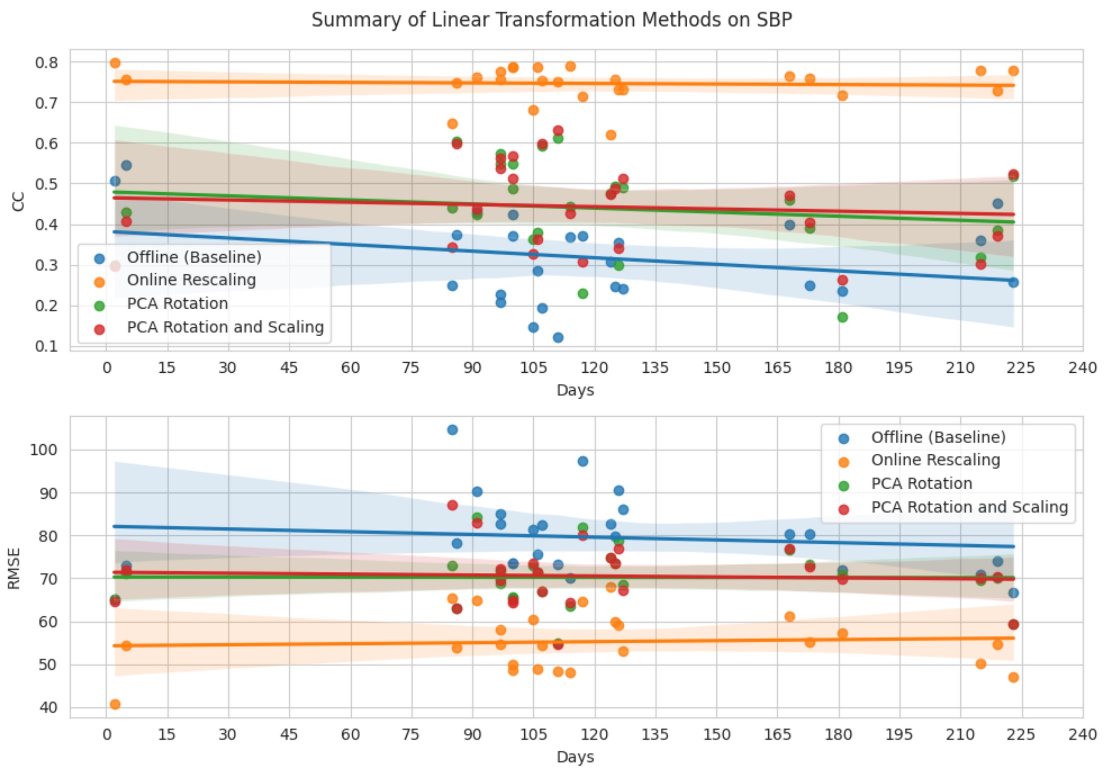


**Figure 5.16:** Summary of linear transformation methods for SBP

| Metrics | Offline Normalisation | Online Normalisation | PCA Rotation | PCA Rotation and Scaling |
|---|---|---|---|---|
| **Mean CC** | 0.3168 ± 0.1071 | 0.7462 ± 0.043 | 0.4392 ± 0.1138 | 0.4426 ± 0.1062 |
| **Mean RMSE** | 79.5725 ± 9.1177 | 55.2512 ± 6.5658 | 70.2241 ± 6.4536 | 70.5426 ± 7.0981 |
| **Slope CC** | -0.0005 | -0.0000 | -0.0003 | -0.0002 |
| **P-value CC** | 0.1928 | 0.7919 | 0.4537 | 0.6599 |
| **Slope RMSE** | -0.0212 | 0.008 | -0.0007 | -0.0072 |
| **P-value RMSE** | 0.5549 | 0.7581 | 0.9791 | 0.7972 |

**Table 5.5:** Performance metrics for SBP (offline and online normalised, with and without PCA)

and root mean square error (RMSE) as evaluation metrics. Building upon the unsupervised adaptation results achieved through online normalization (ON), we aimed to demonstrate the effectiveness of minimal supervised retraining, particularly on the first hidden layer (TEL) of the SNN, to address the temporal evolution of neural firing dynamics in the Primary Motor Cortex (M1) as well as other disruptive actors, like noise and gliar scars.

The initial model was trained on the first session, *indy_20160622_01*, using 80% of the data for training. Then the model was adapted session-by-session, in sequence. The amount of data and number of epochs used to apply ON and during retraining varied between experiments. Our aim was to investigate how different retraining strategies would impact long-term stability and whether minimal retraining of specific layers could maintain or improve decoder performances. To train or fine tune the model, we always used STBP, with AdamW and Dropout, updating only weights and biases.

### 5.5.1 Experiment 1: Extensive Retraining Across Sessions (Whole Model)

The objective of this experiment was to determine the upper bound for CC and the lower bound for RMSE, and to evaluate the impact of prior knowledge on model performance. Two models were trained for 24 epochs on 80% of the data from each normalized session, with evaluation on the remaining 20%. This process was repeated sequentially for all 26 sessions. One model, called "Retrained", was trained from scratch for each session, while the other, referred to as the “Resumed”, was initialized at each session with the weights and biases from the previous session. This setup should allow the resumed model to adapt its parameters to new data while retaining knowledge from earlier sessions.

Both models demonstrated high CC and low RMSE across all sessions, with the resumed model consistently outperforming the retrained model, as shown in Figure 5.17. The results obtained are comparable to the state-of-the-art, presented in Section 3.1.

The figure plots CC and RMSE over time, highlighting the trends across days since the second session, the first is excluded as identical to the single session experiment, and to be consistent with the rest of the tests. While the resumed model maintained stable performance, even increasing for CC, the retrained model showed a slight but constant decline over time. However, this was not proven to be statistically significant, and could be determined by different variables, like noise or the variable length of each session.

The mean and standard deviations for CC and RMSE are summarized in Table 5.6, further confirming the superior performance of the resumed model throughout the sessions and emphasizing the advantages of leveraging prior knowledge. The resumed model results are almost comparable with the state-of-the-art for ANNs tested on the same dataset 3.1.

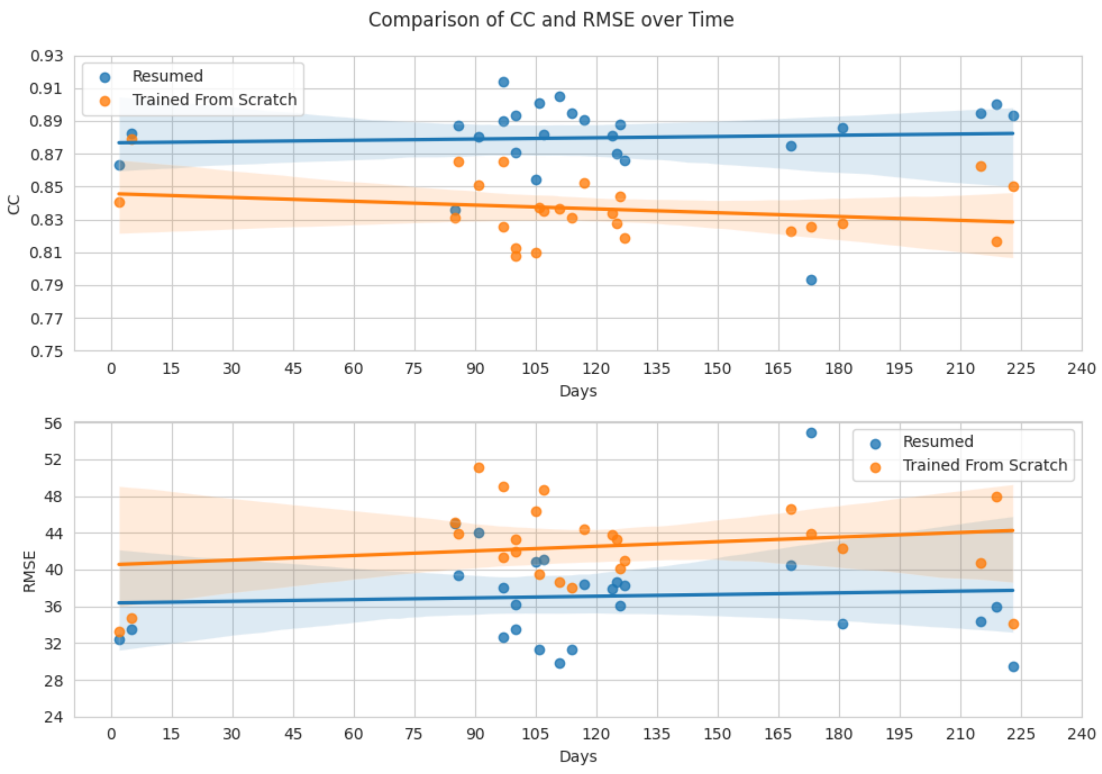


**Figure 5.17:** Performance trends (CC and RMSE) for resumed and retrained models across sessions.

| Metric | Resumed Model | Retrained Model |
|---|---|---|
| CC | 0.8797 ± 0.0242 | 0.8364 ± 0.0183 |
| RMSE | 37.1033 ± 5.4681 | 42.5367 ± 4.5032 |

**Table 5.6:** Mean and standard deviation of CC and RMSE for the resumed and retrained models across sessions with SBP.

### 5.5.2 Experiment 2: Short-Term Adaptations (Single Layers)

To assess whether minimal retraining could maintain or improve performance while reducing computational demand, we performed short-term adaptations on the entire model and on single layers. Our ultimate objective was to evaluate performance increase when fine-tuning for a very short time on the first layer (TEL). The starting hypothesis was that adapting this layer, which is responsible for encoding spikes when fed with continuous signals (SBP), could effectively address for shifts in data domain through time.

Initially, we trained the model on the first 80% of RS normalised, then, for each IS, we took the first 2 minutes, and adapted the model on those, before evaluating the performances on the rest of the ISs.

The complete adaptation consisted in computing $\mu_{\text{inf}}$ and $\sigma_{\text{inf}}$ on the first 2 minutes, apply ON, and then fine tune for just 1 epoch on the same samples. At each IS, weights and biases are resumed from the previous session.

The retrained parameters were only weights and biases of connections before each LIF-neuron layer. We investigated the impact of retraining the entire model or individual layers —either connections before the first, second, or third LIF layer— to determine if focusing on specific components of the model yielded comparable performance gains.

Despite the limited duration of fine tuning, the decoder maintained reasonable performance levels. Moreover, CC exhibited a positive trend over time, as shown in Figure 5.18, indicating that even minimal retraining allowed the model to adjust effectively to the evolving neural dynamics. The RMSE values remained relatively stable or slightly decreased, suggesting improved decoding accuracy too.

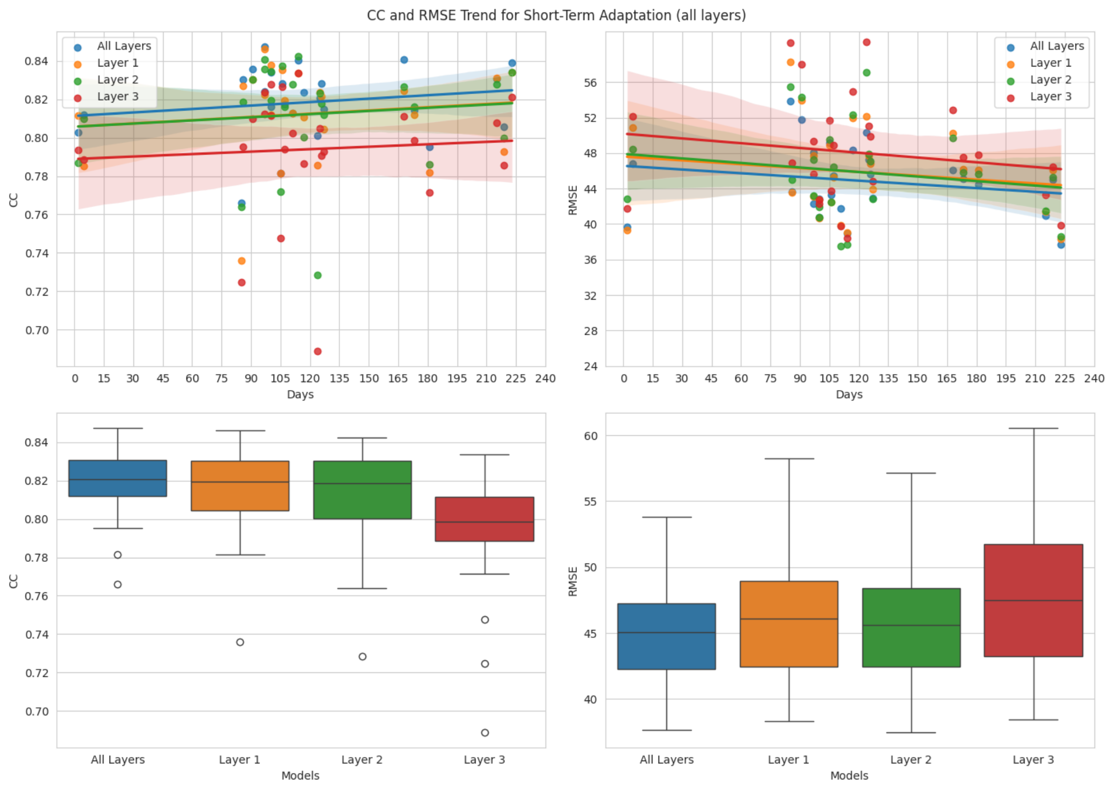


**Figure 5.18:** CC and RMSE over time for short-term adaptations with SBP. Retraining all layers (Experiment 3) and individual layers (Experiment 4)

The mean and standard deviation of CC and RMSE for this experiment are summarized in Table 5.8, confirming the effectiveness of minimal retraining.

The results indicated that retraining the connections before the first or second LIF layer yielded performance metrics nearly identical, this is confirmed by Table 5.7. Moreover the difference between retraining the whole model and the first layer are minimal, with an CC increment of 0.7% and an RMSE decrement of 2% on average.

| **Test** | **Layer 1 vs layer 2 (P-value)** |
|---|---|
| Wilcoxon test for CC | 0.7310 |
| t-test for CC | 0.9480 |
| Wilcoxon test for RMSE | 0.6528 |
| t-test for RMSE | 0.9736 |

**Table 5.7:** Statistical tests for layer 1 and layer 2 over CC and RMSE metrics.

Retraining the third layer resulted in lower performance and a less pronounced positive trend.

The results also proved the efficacy of computing minimal fine-tuning on TEL, compared to extensively retraining the whole model from scratch at each session. From mean computed over the 25 ISs in Table 5.8 we can notice a little decrement in CC of only 2.3%, and an RMSE increase of 7%, which is slightly higher, but still a quite good result comparing the different efforts.

These findings suggest that adapting the earlier layer, particularly TEL (layer 1), effectively compensated for session-to-session variability in input signals. Finally, the positive trend in CC indicated that the model could benefit from prior knowledge while adjusting to new data.

## 5.6 Comparative Summary Analysis

The results across all experiments reveals important insights into the effectiveness of different retraining strategies. The best unsupervised approach has demonstrated to be simple Online Normalisation (ON) when performed using Spiking band Power (SBP) in synergy with the first hidden layer used as trainable encoding layer (TEL). To further enhance decoding ability of our setup, we then evaluated a supervised approach on the top of this findings. Leveraging prior knowledge showed to be critical, in particular resumed model resulted in the best performance overall, highlighting the importance of past learning in achieving robust long-term stability.

Following this path, short-term fine tuning proved to be an efficient approach, significantly reducing computational demands while maintaining decent prediction quality. Focusing on retraining individual layers, the first hidden layer (TEL) yielded performance comparable to retraining all layers, effectively compensating for session-to-session variability in input signals when dealing with SBP. TEL demonstrated to be crucial to address domain shifts and perform feature extraction (spike encoding). The final positive trends in CC over time indicated that the model continued to learn and adapt effectively even with minimal additional training of a single layer (TEL).

Our final method included SBP, ON, and short-term fine tuning on TEL, offering a practical solution for real-world applications where rapid adaptation and resource constraints are critical.

In Figure 5.19 We can see a direct and detailed comparison between our 3 main results, resumed model, achieving state-of-the-art performances, only ON, showing stable trend over time and overall decent prediction quality and, in the middle, the synergy with ON and fine-tuning TEL, showing a critical increment in metric scores while conserving theoretically light implementation.

The corresponding mean and standard deviation for each method are detailed in Table 5.8.

| **Method** | **CC** | **RMSE** |
|---|---|---|
| **Resumed** | **0.8797 ± 0.0242** | **37.1033 ± 5.4681** |
| Retrained | 0.8364 ± 0.0183 | 42.5367 ± 4.5032 |
| All Layers | 0.8185 ± 0.0188 | 44.8876 ± 3.9202 |
| **Layer 1 (TEL)** | **0.8125 ± 0.0239** | **45.8735 ± 4.9987** |
| Layer 2 | 0.8123 ± 0.0266 | 45.8624 ± 5.1468 |
| Layer 3 | 0.7940 ± 0.0320 | 48.0328 ± 6.0469 |
| online norm (ON) | 0.7462 ± 0.0430 | 55.2512 ± 6.56589 |
| fixed | 0.3168 ± 0.1071 | 79.5725 ± 9.1177 |

**Table 5.8:** Mean and standard deviation of CC and RMSE for different adaptations methods with SBP.

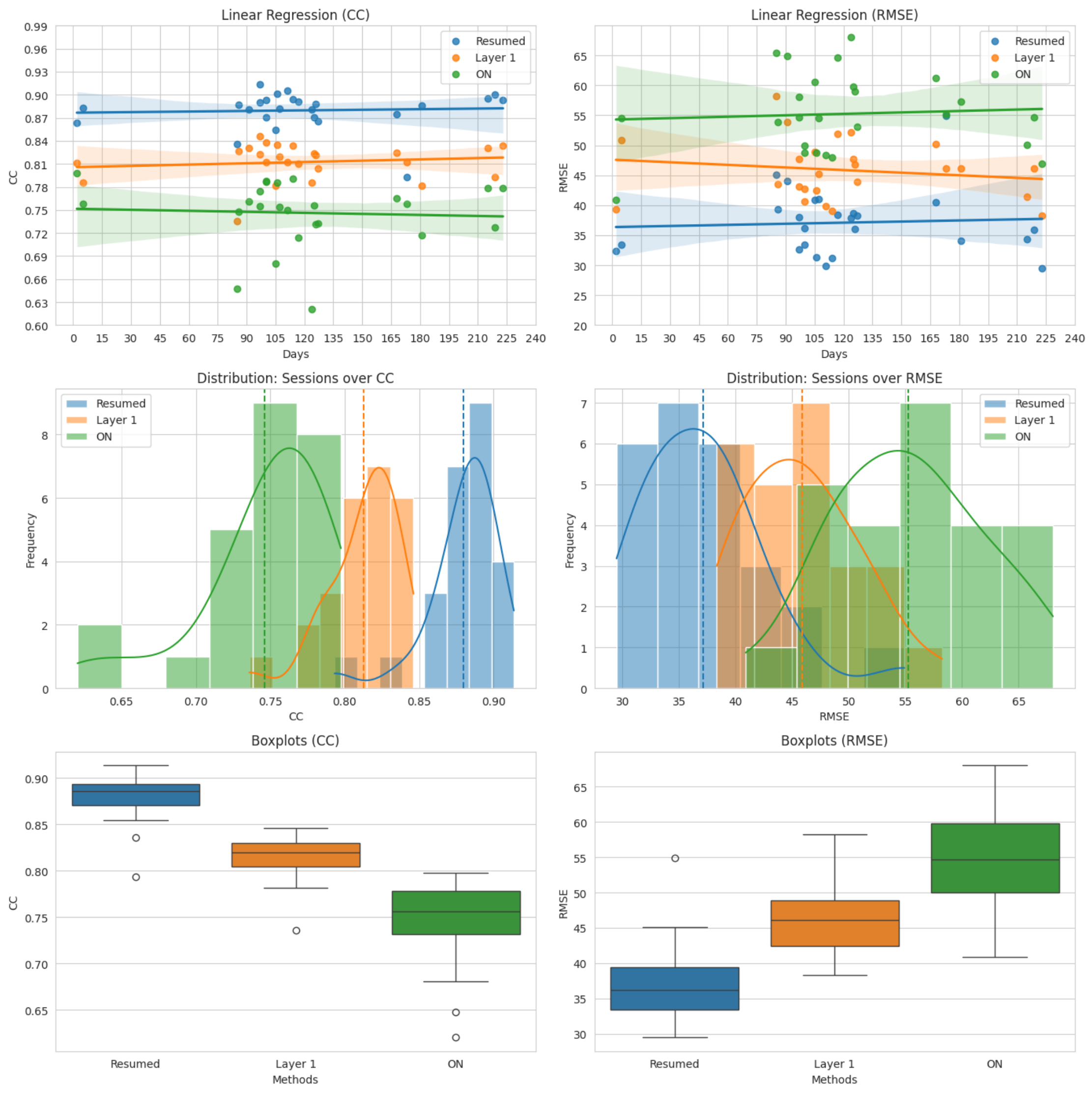


**Figure 5.19:** Comparison between our 3 main results on both CC AND RMSE, with SBP.

# Chapter 6

# Conclusions and Future Works

This thesis investigated preprocessing methods, models and adaptation strategies to effectively decode 2D hand velocity from M1 cortex of NHP Indy, while achieving long-term stability, power efficiency and state-of-the-art accuracy.

First, we accurately selected the model, which is a regression-based spiking neural network designed to predict 2D velocity from 96 input channels. The output layer consists of two Leaky Integrate (LI) neurons, while the preceding layers are composed of Leaky Integrate-and-Fire (LIF) neurons. These fully connected layers are trained using Spatio-Temporal Back Propagation (STBP), allowing the model to capture the temporal dynamics essential for accurate kinematic predictions. To further enhance performance, the model incorporates Dropout regularization to prevent overfitting and the AdamW optimizer to improve training efficiency.

Later, we conducted a series of experiments and researches to find the best preprocessing method to address not only prediction quality when used in synergy with our model, but also computational simplicity and theoretical power efficiency. We found all these characteristics in spiking band power (SBP).

At last, our work explored a variety of adaptation strategies aimed at enhancing the performance and long-term stability of the model. One of the most effective unsupervised approaches combined Spiking Band Power (SBP) with Simple Online Normalisation (ON), alongside using the first hidden layer as a trainable encoding layer (TEL). This configuration theoretically offers robustness to session-to-session variability, maintaining stable performance, while ensuring power efficiency and keeping the net completely fixed.

In addition to unsupervised methods, supervised adaptation strategies were also evaluated. The resumed model—leveraging prior learning—achieved the highest overall performance, highlighting the importance of preserving past knowledge for robust long-term stability. Short-term fine-tuning of the first hidden layer (TEL) emerged as an efficient approach, reducing computational overhead while maintaining prediction quality. This targeted fine-tuning produced results comparable to model performances when retrained from scratch at each session, theoretically making it an efficient solution for real-world applications where computational resources are limited.

The final proposed method offers a practical and lightweight solution. it integrates the SNN-based decoder with SBP, ON, and fine-tuning of TEL, allowing us to reach a stable CC of 0.81 ± 0.02 and an RMSE of 45.87 ± 5. This result is not distant from scores obtained by complete retraining the model on each session, better by only 2.7% in CC and 7% in RMSE, with a much superior computational workload. Notably, the reduced need for extensive retraining implies that the system can adapt quickly to session-to-session variability with minimal involvement from both the patient and clinical staff. This significantly lowers the burden on the user, as the decoder requires less recalibration and effort to maintain accuracy over time, making it an attractive option for real-world brain-computer interface (BCI) applications where ease of use is critical.

Moreover, for BCI applications, it is essential that the decoder consumes as little energy as possible, remains computationally simple, and generates minimal heat—especially when considering potential implantation in the human body. Ensuring low power consumption and thermal efficiency is crucial to prevent overheating, extend device longevity, and enhance patient safety. However, some aspects of the results remain theoretical, as the model has not been implemented on hardware to evaluate power consumption, nor has the decoder been tested in a real closed-loop scenario with real patients. These gaps suggest important directions for future work, where hardware implementation will allow for the investigation of power efficiency, and closed-loop testing will provide insight into the system's real-time performance, adaptability, and interaction with the human brain.

Another promising area for future exploration is optimizing the methodology used to fine-tune TEL. While the current approach is in principle effective, it may be more computationally intensive than necessary. Spiking neural networks (SNNs) offer alternative algorithms, such as Spike-Timing-Dependent Plasticity (STDP), that enable local and unsupervised learning. Leveraging such techniques for TEL could enable fully unsupervised continual adaptations, resulting in highly efficient long-term stability and greatly enhancing the usability of the decoder in practical applications for the final user.

In conclusion, this thesis lays a strong theoretical foundation for enhancing model adaptability and efficiency through a combination of supervised and unsupervised adaptation strategies. Future work should focus on hardware implementation, real-world closed-loop testing, and exploring more efficient fine-tuning methods like STDP to further improve long-term stability, computational efficiency, and overall usability.

# Bibliography

[1] M. D. Murphy, D. J. Guggenmos, D. T. Bundy, and R. J. Nudo. «Current challenges facing the translation of brain computer interfaces from preclinical trials to use in human patients». In: *Frontiers in Cellular Neuroscience* 9 (2016), p. 497. DOI: 10.3389/fncel.2015.00497 (cit. on pp. 3, 18, 54).

[2] Nur Ahmadi, Timothy G. Constandinou, and Christos-Savvas Bouganis. «Decoding Hand Kinematics from Local Field Potentials Using Long Short-Term Memory (LSTM) Network». In: *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*. 2019, pp. 415–419. DOI: 10.1109/NER.2019.8717045 (cit. on pp. 3, 18–20, 22, 49, 51, 53, 54).

[3] Nur Ahmadi, Timothy Constandinou, and Christos Bouganis. «Robust and accurate decoding of hand kinematics from entire spiking activity using deep learning». In: *Journal of Neural Engineering* 18 (Jan. 2021). DOI: 10.1088/1741-2552/abde8a (cit. on pp. 3, 19, 20, 22, 49–51).

[4] Mikhail A. Lebedev and Miguel A. L. Nicolelis. «Brain-Machine Interfaces: From Basic Science to Neuroprostheses and Neurorehabilitation». In: *Physiological Reviews* 97.2 (2017). PMID: 28275048, pp. 767–837. DOI: 10.1152/physrev.00027.2016 (cit. on pp. 3, 4, 16–18, 54).

[5] Shiv Mudgal, Suresh Sharma, Jitender Chaturvedi, and Anil Sharma. «Brain computer interface advancement in neurosciences: Applications and issues». In: *Interdisciplinary Neurosurgery: Advanced Techniques and Case Management* 20 (Feb. 2020), p. 100694. DOI: 10.1016/j.inat.2020.10069 (cit. on pp. 4, 16, 17).

[6] Xiaolong Wu, benjamin metcalfe benjamin, Shenghong He, Huiling Tan, and Dingguo Zhang. «A Review of Motor Brain-Computer Interfaces using Intracranial Electroencephalography based on Surface Electrodes and Depth Electrodes». In: *TechRxiv* (Apr. 2023). DOI: 10.36227/techrxiv.22340677.v1. URL: http://dx.doi.org/10.36227/techrxiv.22340677.v1 (cit. on pp. 4, 16).

[7] Shoeb Shaikh, Rosa So, Tafadzwa Sibindi, Camilo Libedinsky, and Arindam Basu. «Towards Intelligent Intracortical BMI (i$^2$BMI): Low-Power Neuromorphic Decoders That Outperform Kalman Filters». In: *IEEE Transactions on Biomedical Circuits and Systems* 13.6 (2019), pp. 1615–1624. DOI: 10.1109/TBCAS.2019.2944486 (cit. on pp. 4, 52, 54).

[8] LINKS Foundation. *LINKS Foundation: Leading Innovation and Research*. Accessed: 2024-09-03. 2024. URL: https://linksfoundation.com/en/ (cit. on p. 4).

[9] SMILIES Lab, Politecnico di Torino. *SMILIES Lab: Smart Micro/nanosystems and Internet of Things Laboratory*. Accessed: 2024-09-03. 2024. URL: https://www.smilies.polito.it (cit. on p. 4).

[10] B-Cratos Project. *B-Cratos: Wireless Brain-Connect inteRfAce TO machineS*. Accessed: 2024-09-03. 2024. URL: https://www.b-cratos.eu (cit. on pp. 4, 6).

[11] Casey Henley. *Foundations of Neuroscience*. Illustrator: Casey Henley. Licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. Michigan State University Libraries, 2021. ISBN: 978-1-62610-109-8 (cit. on pp. 9, 13–15).

[12] Dana Foundation. *Anatomy and Function of Brain Areas*. Accessed: August 30, 2024. 2023. URL: https://dana.org/app/uploads/2023/09/anatomy-function-brain-areas-basics-aug-2019-2024.jpeg (cit. on p. 11).

[13] Alan J. McComas. «The Neuromuscular System». In: *Exercise Physiology: People and Ideas*. Ed. by Charles M. Tipton. New York, NY: Springer New York, 2003, pp. 39–97. ISBN: 978-1-4614-7543-9. DOI: 10.1007/978-1-4614-7543-9_2. URL: https://doi.org/10.1007/978-1-4614-7543-9_2 (cit. on p. 12).

[14] Cleveland Clinic. *Somatic Nervous System*. Accessed: 2024-09-01. URL: https://my.clevelandclinic.org/health/body/23291-somatic-nervous-system (cit. on p. 13).

[15] Encyclopædia Britannica. *Homunculus*. Image. Accessed: September 1, 2024. 2024. URL: https://www.britannica.com/science/homunculus-biology#/media/1/270724/277135 (cit. on p. 14).

[16] Hans Scherberger. «Neural Prostheses for Reaching». In: *Encyclopedia of Neuroscience* (Dec. 2009). DOI: 10.5167/uzh-32047 (cit. on p. 16).

[17] Álvaro Costa, Enrique Hortal, Eduardo Iáñez, and José M. Azorín. «A Supplementary System for a Brain-Machine Interface Based on Jaw Artifacts for the Bidimensional Control of a Robotic Arm». In: *PLOS ONE* 9.11 (Nov. 2014), pp. 1–13. DOI: 10.1371/journal.pone.0112352. URL: https://doi.org/10.1371/journal.pone.0112352 (cit. on p. 17).

[18] E. J. Hwang and R. A. Andersen. «The utility of multichannel local field potentials for brain–machine interfaces». In: *Journal of Neural Engineering* 10.4 (2013), p. 046005. DOI: 10.1088/1741-2560/10/4/046005 (cit. on p. 18).

[19] Nur Ahmadi, Trio Adiono, Ayu Purwarianti, Timothy G. Constandinou, and Christos-Savvas Bouganis. «Improved Spike-Based Brain-Machine Interface Using Bayesian Adaptive Kernel Smoother and Deep Learning». In: *IEEE Access* 10 (2022), pp. 29341–29356. DOI: 10.1109/ACCESS.2022.3159225 (cit. on pp. 18, 20, 21, 49, 51).

[20] Elaine Astrand, Claire Wardak, and Suliann Ben Hamed. «Selective visual attention to drive cognitive brain–machine interfaces: from concepts to neurofeedback and rehabilitation applications». In: *Frontiers in Systems Neuroscience* 8 (2014). ISSN: 1662-5137. DOI: 10.3389/fnsys.2014.00144. URL: https://www.frontiersin.org/journals/systems-neuroscience/articles/10.3389/fnsys.2014.00144 (cit. on p. 18).

[21] P. Ahmadipour, Y. Yang, E. F. Chang, and M. M. Shanechi. «Adaptive tracking of human ECoG network dynamics». In: *Journal of Neural Engineering* 18.1 (Feb. 2021), p. 016011. DOI: 10.1088/1741-2552/abae42 (cit. on pp. 19, 20).

[22] Nur Ahmadi, Timothy Constandinou, and Christos Bouganis. «Impact of referencing scheme on decoding performance of LFP-based brain-machine interface». In: *Journal of Neural Engineering* 18 (Nov. 2020). DOI: 10.1088/1741-2552/abce3c (cit. on pp. 19, 21).

[23] John W. Kelly, Daniel P. Siewiorek, Asim Smailagic, and Wei Wang. «Automated Filtering of Common-Mode Artifacts in Multichannel Physiological Recordings». In: *IEEE Transactions on Biomedical Engineering* 60.10 (2013), pp. 2760–2770. DOI: 10.1109/TBME.2013.2264722 (cit. on p. 20).

[24] Nur Ahmadi, Timothy Constandinou, and Christos Bouganis. «Inferring entire spiking activity from local field potentials». In: *Scientific Reports* 11 (Sept. 2021). DOI: 10.1038/s41598-021-98021-9 (cit. on pp. 20, 22).

[25] Joseph G Makin, Joseph E O'Doherty, Mariana M B Cardoso, and Philip N Sabes. «Superior arm-movement decoding from cortex with a new, unsupervised-learning algorithm». In: *Journal of Neural Engineering* 15.2 (Jan. 2018), p. 026010. DOI: 10.1088/1741-2552/aa9e95. URL: https://dx.doi.org/10.1088/1741-2552/aa9e95 (cit. on pp. 20, 52, 56).

[26] Samuel R. Nason et al. «A low-power band of neuronal spiking activity dominated by local single units improves the performance of brain–machine interfaces». In: *Nature Biomedical Engineering* 4 (2020), pp. 973–983 (cit. on pp. 22–25, 51).

[27] Jiawei Liao, Lars Widmer, Xiaying Wang, Alfio Di Mauro, Samuel R. Nason-Tomaszewski, Cynthia A. Chestek, Luca Benini, and Taekwang Jang. «An Energy-Efficient Spiking Neural Network for Finger Velocity Decoding for Implantable Brain-Machine Interface». In: *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. 2022, pp. 134–137. DOI: 10.1109/AICAS54282.2022.9869846 (cit. on pp. 24, 25, 35, 38, 50–52, 65, 76).

[28] Shirin Dora and Nikola Kasabov. «Spiking Neural Networks for Computational Intelligence: An Overview». In: *Big Data and Cognitive Computing* 5.4 (2021). ISSN: 2504-2289. DOI: 10.3390/bdcc5040067. URL: https://www.mdpi.com/2504-2289/5/4/67 (cit. on pp. 25, 28, 41).

[29] Sanaullah Sanaullah, Shamini Koravuna, and Thorsten Jungeblut. «Exploring spiking neural networks: a comprehensive analysis of mathematical models and applications». In: *Frontiers in Computational Neuroscience* 17 (Aug. 2023). DOI: 10.3389/fncom.2023.1215824 (cit. on pp. 25, 28, 40, 41).

[30] Alessio Carpegna, Alessandro Savino, and Stefano Di Carlo. «Spiker: an FPGA-optimized Hardware accelerator for Spiking Neural Networks». In: *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. ISSN: 2159-3477. July 2022, pp. 14–19. DOI: 10.1109/ISVLSI54635.2022.00016 (cit. on p. 25).

[31] Alessio Carpegna, Alessandro Savino, and Stefano Di Carlo. *Spiker+: a framework for the generation of efficient Spiking Neural Networks FPGA accelerators for inference at the edge.* arXiv:2401.01141 [cs]. Jan. 2024. DOI: 10.48550/arXiv.2401.01141. URL: http://arxiv.org/abs/2401.01141 (visited on 01/26/2024) (cit. on p. 25).

[32] Dario Padovano, Alessio Carpegna, Alessandro Savino, and Stefano Di Carlo. «SpikeExplorer: Hardware-Oriented Design Space Exploration for Spiking Neural Networks on FPGA». en. In: *Electronics* 13.9 (Jan. 2024). Number: 9 Publisher: Multidisciplinary Digital Publishing Institute, p. 1744. ISSN:

2079-9292. DOI: 10.3390/electronics13091744. URL: https://www.mdpi.com/2079-9292/13/9/1744 (visited on 09/06/2024) (cit. on p. 25).

[33] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. «Deep learning in spiking neural networks». In: *Neural Networks* 111 (Mar. 2019), pp. 47–63. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2018.12.002. URL: http://dx.doi.org/10.1016/j.neunet.2018.12.002 (cit. on pp. 25, 76).

[34] Jason K. Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. *Training Spiking Neural Networks Using Lessons From Deep Learning*. 2023. arXiv: 2109.12894 [cs.NE]. URL: https://arxiv.org/abs/2109.12894 (cit. on pp. 25, 28, 40, 73, 76).

[35] Laith Alzubaidi et al. «Review of deep learning: concepts, CNN architectures, challenges, applications, future directions». In: *Journal of Big Data* 8.1 (2021), pp. 1–74. DOI: 10.1186/S40537-021-00444-8 (cit. on pp. 26, 28).

[36] Diab Abueidda, Seid Koric, Rashid Abu Al-Rub, Corey Parrott, Kai James, and Nahil Sobh. *A deep learning energy method for hyperelasticity and viscoelasticity*. Jan. 2022. DOI: 10.48550/arXiv.2201.08690 (cit. on p. 26).

[37] unknown. *Recurrent Neural Network (RNN)*. Accessed: 2024-09-23. URL: https://dengliangshi.github.io/images/bptt/rnn.png (cit. on p. 27).

[38] Mingfeng Li. «Comprehensive Review of Backpropagation Neural Networks». In: *Academic Journal of Science and Technology* 9 (Jan. 2024), pp. 150–154. DOI: 10.54097/51y16r47 (cit. on p. 32).

[39] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. «Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks». In: *Frontiers in Neuroscience* 12 (May 2018). ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00331. URL: http://dx.doi.org/10.3389/fnins.2018.00331 (cit. on p. 35).

[40] Paul Werbos. «Generalization of Backpropagation with Application to a Recurrent Gas Market Model». In: *Neural Networks* 1 (Dec. 1988), pp. 339–356. DOI: 10.1016/0893-6080(88)90007-X (cit. on p. 35).

[41] Chenlin Zhou et al. «Direct training high-performance deep spiking neural networks: a review of theories and methods». In: *Frontiers in Neuroscience* 18 (July 2024). DOI: 10.3389/fnins.2024.1383844 (cit. on p. 36).

[42] unknown. *Error Propagation in RNN*. Accessed: 2024-09-23. URL: https://dengliangshi.github.io/images/bptt/error.png (cit. on p. 36).

[43] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization.* 2019. arXiv: `1711.05101 [cs.LG]`. URL: `https://arxiv.org/abs/1711.05101` (cit. on p. 38).

[44] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. «Dropout: A Simple Way to Prevent Neural Networks from Overfitting». In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: `http://jmlr.org/papers/v15/srivastava14a.html` (cit. on pp. 38, 39).

[45] Juan Gallego, Matthew Perich, Raeed Chowdhury, Sara Solla, and Lee Miller. «Long-term stability of cortical population dynamics underlying consistent behavior». In: *Nature Neuroscience* 23 (Feb. 2020), pp. 1–11. DOI: `10.1038/s41593-019-0555-4` (cit. on pp. 43–45, 52).

[46] Marcello Zanghieri, Mattia Orlandi, Elisa Donati, Emanuele Gruppioni, Luca Benini, and Simone Benatti. «Online Unsupervised Arm Posture Adaptation for sEMG-based Gesture Recognition on a Parallel Ultra-Low-Power Microcontroller». In: *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS).* 2023, pp. 1–5. DOI: `10.1109/BioCAS58349.2023.10388902` (cit. on pp. 43–45, 52, 62).

[47] Joseph E. O'Doherty, Mariana M. B. Cardoso, Joseph G. Makin, and Philip N. Sabes. *Nonhuman Primate Reaching with Multichannel Sensorimotor Cortex electrophysiology.* May 2017. DOI: `10.5281/zenodo.788569`. URL: `https://doi.org/10.5281/zenodo.788569` (cit. on pp. 48, 49, 55, 56, 67).

[48] Marta Bono. «Time Robustness of Deep Learning Models for Real-Time Neural Decoding of Arm Movement». Rel. Gabriella Olmo, Paolo Viviani. Corso di Laurea Magistrale in Ingegneria Biomedica. Politecnico di Torino, 2023 (cit. on pp. 49, 55, 57, 59, 67).

[49] Paolo Viviani et al. «Deep Learning for Real-Time Neural Decoding of Grasp». In: *Machine Learning and Knowledge Discovery in Databases: Applied Data Science and Demo Track.* Ed. by Gianmarco De Francisci Morales, Claudia Perlich, Natali Ruchansky, Nicolas Kourtellis, Elena Baralis, and Francesco Bonchi. Cham: Springer Nature Switzerland, 2023, pp. 379–393. ISBN: 978-3-031-43427-3 (cit. on p. 49).

[50] Nur Ahmadi, Timothy Constandinou, and Christos Bouganis. «End-to-End Hand Kinematic Decoding from LFPs Using Temporal Convolutional Network». In: Oct. 2019, pp. 1–4. DOI: `10.1109/BIOCAS.2019.8919131` (cit. on p. 49).

[51] K. Yamazaki, V. K. Vo-Ho, D. Bulsara, and N. Le. «Spiking Neural Networks and Their Applications: A Review». In: *Brain Sciences* 12.7 (June 2022), p. 863. DOI: 10.3390/brainsci12070863. URL: https://doi.org/10.3390/brainsci12070863 (cit. on p. 50).

[52] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Prasad Joshi, Andrew Lines, Andreas Wild, Hong Wang, and Deepak Mathaikutty. «Loihi: A Neuromorphic Manycore Processor with On-Chip Learning». In: *IEEE Micro* PP (Jan. 2018), pp. 1–1. DOI: 10.1109/MM.2018.112130359 (cit. on p. 50).

[53] Alberto Dequino, Alessio Carpegna, Davide Nadalini, Alessandro Savino, Luca Benini, Stefano Di Carlo, and Francesco Conti. *Compressed Latent Replays for Lightweight Continual Learning on Spiking Neural Networks*. 2024. arXiv: 2407.03111 [cs.NE]. URL: https://arxiv.org/abs/2407.03111 (cit. on p. 52).

[54] Elijah A. Taeckens and Sahil Shah. «A Spiking Neural Network with Continuous Local Learning for Robust Online Brain Machine Interface». In: *bioRxiv* (2023). DOI: 10.1101/2023.08.16.553602. eprint: https://www.biorxiv.org/content/early/2023/12/13/2023.08.16.553602.full.pdf. URL: https://www.biorxiv.org/content/early/2023/12/13/2023.08.16.553602 (cit. on p. 52).

[55] Kristopher Jensen, Ta-Chu Kao, Jasmine Stone, and Guillaume Hennequin. «Scalable Bayesian GPFA with automatic relevance determination and discrete noise models». In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 10613–10626. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/58238e9ae2dd305d79c2ebc8c1883422-Paper.pdf (cit. on p. 54).

[56] Minki Kim, Jeong-woo Sohn, and Sung-Phil Kim. «Decoding Kinematic Information From Primary Motor Cortex Ensemble Activities Using a Deep Canonical Correlation Analysis». In: *Frontiers in Neuroscience* 14 (Oct. 2020). DOI: 10.3389/fnins.2020.509364 (cit. on p. 54).