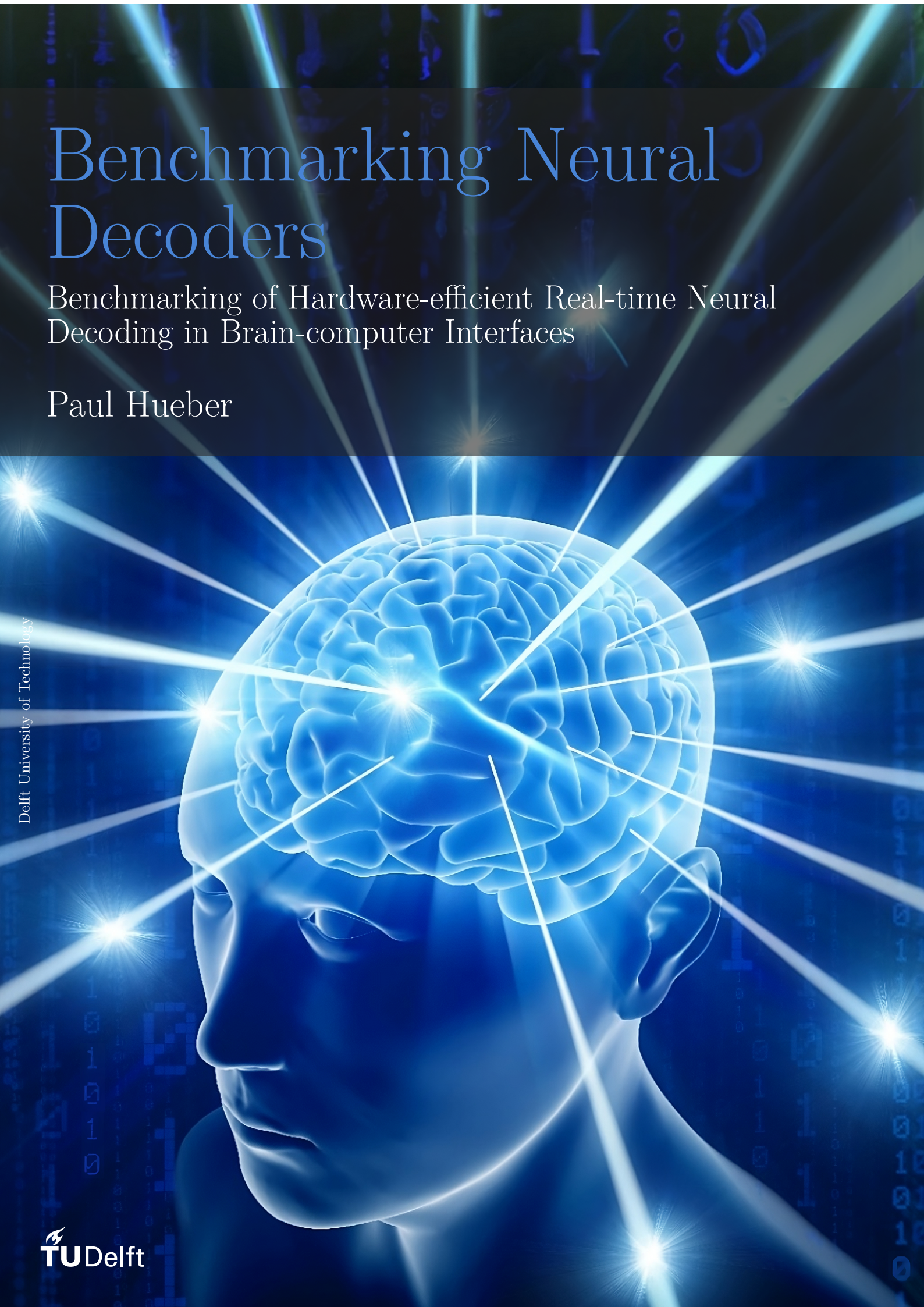
Benchmarking Neural Decoders

Benchmarking of Hardware-efficient Real-time Neural Decoding in Brain-computer Interfaces

Paul Hueber



Delft University of Technology

TU Delft

Benchmarking Neural Decoders

Benchmarking of Hardware-efficient Real-time Neural Decoding in Brain-computer Interfaces

by

Paul Hueber

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday November 24, 2023.

Student number:	5151961	
Project duration:	November 1, 2022 – November 1, 2023	
Thesis committee:	Dr. Nergis Tömen	TU Delft, supervisor
	Dr. Jan van Gemert	TU Delft
	Dr. Ricardo Marroquim	TU Delft

This thesis is confidential and cannot be made public until November 24, 2025.

Style: TU Delft Report Style, with modifications by Daan Zwaneveld and Paul Hueber

TU Delft

Preface

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Paul Hueber
Delft, November 2023

Summary

Brain-computer interfaces (BCIs) are technological systems that facilitate communication between the brain and external devices, allowing patients with disabilities to restore cognitive or physical functions. One promising application of BCIs is closed-loop neuromodulation (CLN), which is the process of applying electrical stimulation to neurons to alter neural activity for therapeutic treatment or research intra-cortical interactions. CLN requires neural decoding algorithms that interpret neural activity such that the BCI can provide real-time feedback or control external devices. The primary challenges in designing neural decoders for implantable closed-loop BCI systems are the requirement of low latency to enable real-time feedback and high energy efficency to prevent neural cell damage caused by heat diffusion. Previous benchmarks have provided limited insights into the effectiveness of neural decoders in this constrained environment. This paper introduces comprehensive metrics that consider a range of factors, such as model fidelity, latency, power consumption, and memory size, crucial for the practical application of neural decoders for implantable BCIs in CLN. This study benchmarks traditional neural decoders, artificial neural networks (ANNs), and spiking neural networks (SNNs) against these metrics on their ability to accurately predict the kinematics of a primate's finger movements from neural activity in the motor cortex. This thesis explores the advantages and disadvantages of each type of decoder, examining their individual and comparative performances against the established metrics and evaluating their suitability for integration into closed-loop BCI systems.

The research has yielded significant findings, presenting the SNNs as a potential candidate as the efficient, low-latency, and low-energy neural decoder required by constrained iBCI applications. Unlike traditional decoders and ANNs, SNNs exhibit competitive decoding accuracy while maintaining lower latency and power consumption. This study highlights inherent trade-offs in neural decoder optimization, especially in balancing fidelity with latency, power consumption, and memory size requirements. While more complex neural network architectures can potentially increase decoding accuracy, they may also lead to increased latency and power consumption, diminishing their suitability for real-time closed-loop applications. The comprehensive analysis and results from this benchmarking exercise provide significant insights into developing neural decoders for closed-loop implantable BCI.

In conclusion, this thesis proposes metrics to evaluate the suitability of neural decoders for closed-loop BCIs. SNNs have been shown to be capable of effectively extracting neural dynamics from extracellular spiking data while maintaining a delicate balance between fidelity, power consumption, and latency. This balance is crucial for their application in real-time, energy-sensitive environments of implantable BCIs. The research findings advocate for a promising direction in advancing neurotechnology and BCIs, emphasizing the necessity for a harmonized approach in decoder design, which must fulfill the critical requirements of implantable systems while ensuring competitive performance. This thesis contributes to the growing body of knowledge in neuromorphic computing for neuroscience and offers the neuromorphic SNN as a candidate for implantable closed-loop BCI, paving the way for advancements that could significantly enhance the efficacy and practicality of BCI technologies.

The remainder of this thesis report comprises two parts: the main scientific report and a technical background section, providing details on neuromodulation, the evaluated neural decoders, and the benchmark. The scientific report restates the motivation in Section 1. Section 2 details the low latency and high efficacy requirements of neural decoders for iBCIs for CLN, with Appendix B providing more background on neuromodulation. Section 3 introduces metrics crucial for a comprehensive evaluation of the practical application of neural decoders in this constrained environment, which are fully explained in Appendix C. Section 4 provides an overview of the benchmark and the evaluated neural decoders. The Appendices D, E, and F offer additional information on the evaluated decoders. Appendix G and Appendix H present background knowledge on the training and evaluation of neural networks. Optimization and regularization techniques are discussed in Appendix I . The results of the benchmark are presented in Section 5 and discussed in Section 6.

Contents

List of Figures

List of Tables

Abstract

Designing processors for implantable closed-loop neuromodulation systems presents a formidable challenge owing to the constrained operational environment, which requires low latency and high energy efficacy. Previous benchmarks have provided limited insights into power consumption and latency. However, this study introduces algorithmic metrics that capture the potential and limitations of neural decoders for closed-loop intra-cortical brain-computer interfaces in the context of energy and hardware constraints. This study benchmarks common decoding methods for predicting a primate's finger kinematics from the motor cortex and explores their suitability for low latency and high energy efficient neural decoding. The study found that ANN-based decoders provide superior decoding accuracy, requiring high latency and many operations to effectively decode neural signals. Spiking neural networks have emerged as a solution, bridging this gap by achieving competitive decoding performance within sub-10ms while utilizing a fraction of computational resources. These distinctive advantages of neuromorphic spiking neural networks make them highly suitable for the challenging closed-loop neural modulation environment. Their capacity to balance decoding accuracy and operational efficiency offers immense potential in reshaping the landscape of neural decoders, fostering greater understanding, and opening new frontiers in closed-loop intra-cortical human-machine interaction.

1. Introduction

Brain-computer interfaces (BCIs) have revolutionized the fields of neuroscience and medicine by enabling individuals with disabilities to interact with external devices and restore lost sensory [1], motor [2], or cognitive [3] functions. Intra-cortical BCIs (iBCIs), a type of invasive BCI that involves placing electrodes directly into the cortex of the brain, have great potential for closed-loop neuromodulation (CLN). CLN alters neural activity using personalized and responsive therapeutic electrical neural stimulation based on the subject's neural activity. CLN has higher efficacy [4] and lower risk of side effects [5] than fixed stimulation, that is, open-loop neuromodulation, as shown in Figure 1. CLN requires neural decoders that interpret neural activity such that the BCI can provide real-time feedback stimulation or control external devices based on the subject's neural activity.

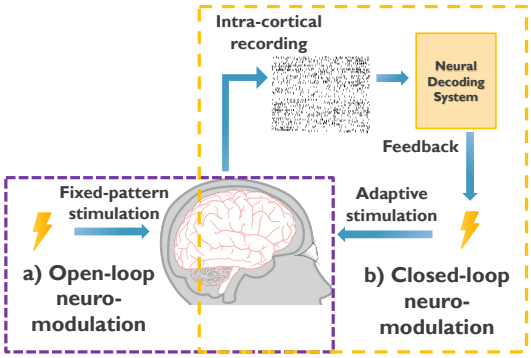


Figure 1: Paradigm of a closed-loop and open-loop neuromodulations. a) During open-loop neuromodulation, the subject receives fixed stimulation, in contrast to b) closed-loop neuromodulation (CLN), during which the subject receives adaptive stimulation based on recorded and decoded neural activities. CLN enables individualized, responsive therapeutic treatment based on the neural activity of the subject, improving the effectiveness of the therapeutic treatment and reducing side effects.

Designing iBCIs for CLN is challenging because of the highly resource-constrained environment of the implants. Even a slight temperature increase of one degree can cause damage to neural cells [6]. Moreover, a decoding time of a few milliseconds is required for CLN aimed at inter-areal interactions [7, 8]. This involves the development of highly energy-efficient and low-latency neural decoders that can overcome the constraints of low latency and energy consumption.

Benchmarking neural decoders is crucial in helping researchers assess the clinical viability of iBCI systems. However, traditional benchmarks have mainly focused on accuracy and operational costs, while neglecting other essential metrics necessary for practical clinical applications [9, 10].

In contrast, this study introduces a method to extrapolate algorithmic-to-hardware metrics, addressing the gap encompassing all essential constraints required to evaluate and compare the suitability of neural decoders for iBCIs in the context of closed-loop neuromodulation. Any benchmark designed to compare neural decoders for iBCI within the context of CLN must consider the co-optimization between hardware and software. Only then can we benchmark effectively and accurately evaluate the power consumption, latency, and fidelity of neural decoders, providing a holistic assessment of decoder suitability for real-time low-energy applications.

Section 2 underscores the necessity of local processing for *in vivo* iBCI for CLN and reiterates the critical energy efficiency and low-latency requirements inherent to the constrained operational environment. Section 3 presents the metrics designed

to suit the energy- and hardware-constrained environment of iBCIs suitable for CLN. Section 4 introduces six neural decoders and evaluates their ability to predict a primate's finger movements. Section 6 addresses the implications of the benchmarking results for the closed-loop applicability of neural decoders for iBCI. Finally, Section 7 discusses the future directions, limitations, and importance of this research.

This paper introduces metrics that researchers can use to avoid the complex co-optimization process by evaluating neural decoders algorithmically while incorporating hardware considerations. Six neural decoders were benchmarked on neural data suitable for closed-loop iBCI, and the presented neuromorphic spiking neural network (SNN) emerged as a potential candidate for the constrained iBCI environment for CLN.

2. Related Work

Intra-cortical neuronal recordings from the motor cortex using single recording units date back to the late 1960s, when Evarts conducted groundbreaking work capturing extracellular neural activity in conscious primates engaged in diverse motor tasks [11]. Today, almost 60 years later, neural recording has undergone revolutionary evolution owing to innovative technologies such as high-density probes [12], high-density microelectrode arrays (HD-MEAs) [13], and carbon nanotube yarn (CNTY) biosensors [14]. These technologies have made it possible to record the activities of more neurons with a higher spatial resolution and coverage and have paved the way for more clinically viable and high-performance BCIs.

BCIs help subjects with disabilities to interact with external devices, such as neuroprosthetics [15, 16], or restore lost sensory [1], motor [2], or cognitive [3] functions by translating neural activity from the brain into control commands through neural decoding. In addition to therapeutic applications, iBCIs advance our understanding of the complex neural processes that underlie behavior [17–20], cognition [21], and perception [22].

Two types of BCIs can be distinguished: non-invasive and invasive [23]. Invasive BCIs involve implanting electrodes into or on the cortex. Intra-cortical BCIs are a specific subtype of invasive BCIs in which electrodes are inserted into the cortex, which is the outermost layer of the brain. They provide the finest spatial and temporal resolution and excellent signal quality [24, 25]. Although iBCIs carry a higher risk owing to surgical implantation, their superior spatiotemporal resolution is crucial for high-precision neural decoding of complex tasks [15, 16].

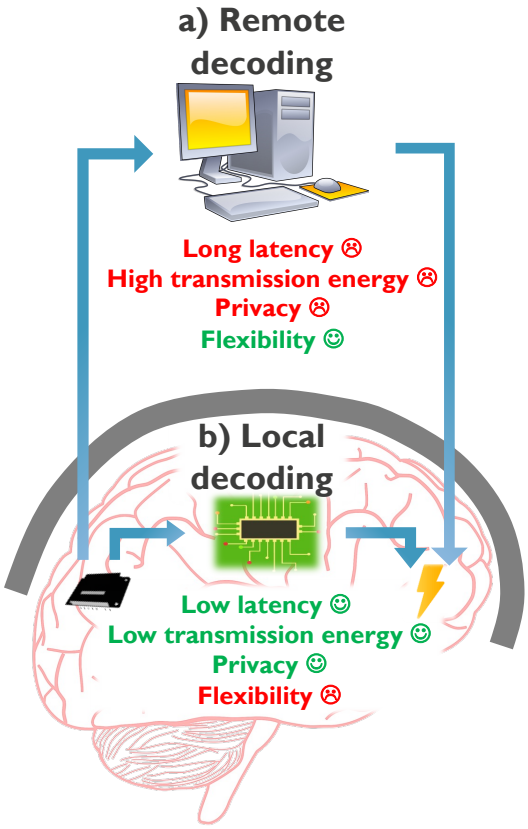


Figure 2: Advantages and disadvantages of external and local processing. a) External processing requires transferring neural data or extracted features to an external computing device. Wireless data transfer suffers from long latency, high transmission energy, and privacy concerns. b) Local neural processing can be significantly faster, with low transmission energy, at the cost of little flexibility.

One promising field of iBCI is neuromodulation. Traditionally, neuromodulation described the physiological processes by which neurons use neurotransmitters to regulate neural activity. More recently, neuromodulation has been adapted to refer to the process of altering neural activity via electric stimulation to restore normal neurological functions or study intra-cortical interaction [5].

Neuromodulation can be classified into open and closed-loop systems, as shown in Figure 1. Open-loop neuromodulation involves delivering neural stimulation without real-time feedback from the targeted neural system with predefined and fixed stimulation parameters, such as strength or timing. In contrast, CLN with iBCI uses bi-directional communication between the brain and computing devices, providing adaptive feedback to adapt and adjust the parameters of interventions, enabling personalized and responsive therapeutic neural modulation based on the subject's neural activity [5, 26]. The adaptive and interactive nature of CLN enhances efficacy [4] and reduces the side ef-

fects of neural stimulations [5]. For the remainder of this paper, CLN refers exclusively to neuromodulation via intra-cortical BCI.

CLN typically requires a powerful external computer to decode neural activities [27]. More sophisticated neural tasks, such as decoding sensory and motor cortex interactions, require high channel counts of neural recordings with fine spatiotemporal resolutions [15, 16], generating vast amounts of recording data, which impose significant limitations on the real-time applicability of neural decoders, see Figure 2. Transferring data from intracortical neural sensors to an external system requires energy intensive wireless transmission [28, 29], and a limited wireless transmission bandwidth can increase the latency [6, 30–32]. Moreover, the transfer of neural data for processing to an external computer raises privacy concerns.

To address these concerns, data transmission can be avoided by eliminating the need for external computing and decoding neural signals locally on the implant. This eliminates the necessity for intensive data communication collectively, except for programming the implant [27] or diagnostics [30]. Valencia and Alimohammad [27] highlighted the need for local processing in a fully implantable iBCI for *in vivo* closed-loop neural decoding, eliminating data transmission during inference and significantly reducing energy consumption, latency [30], and privacy concerns.

Designing iBCIs for CLN is challenging because of the highly resource-constrained environment of implants. The implant volume should be minimized to reduce the invasiveness and risks of the surgery, and low power is required to prevent tissue damage due to even a one degree temperature increase [33]. However, iBCIs are required to process an exponentially growing amount of data [34], which adds to the complexity of the decoding task.

Minimal heat diffusion is required to ensure safe local processing, without causing tissue damage. While Wolf [35] initially cautiously recommended limiting the temperature of implant electronics to below $40\,\mathrm{mW\,cm^{-2}}$ heat flux and $2\,^{\circ}\mathrm{C}$, a much lower limit of $1\,^{\circ}\mathrm{C}$ is vital for maintaining long-term neural cell health [6]. This means that $1\,\mathrm{cm}^2$ of implant electronics cannot exceed a power dissipation of $400\,\mu\mathrm{W}$ [36], which is $10{,}000\times$ lower than that of smartphone processors. Given the conservative nature of these recommendations [35], minimizing power consumption beyond the stated limits is paramount. However, traditional processors cannot satisfy this energy requirement [6], and low-power implants performing neural decoding is a major obstacle to the successful clinical adoption of real-time closed-loop iBCIs [34].

	Metric	Evaluates
Fidelity	R2	Proportion of explainable data variance
	r	Temporal alignment of label and prediction
Latency	Binning Latency	Time window of input data needed per inference
	Processing Latency	Operational delay optimized by the algorithm designer
Power Consumption	Total eff. Operations	Number of effective operations needed per inference
	Memory Access	Number of effective memory access needed per inference
Size	Memory Footprint	Number of bits required to store the decoder

Table 1: Overview of the proposed closed-loop iBCI metrics.

Real-time processing is another crucial requirement for closed-loop iBCIs, in which continuous neural recording, decoding, and feedback occur in real time. This necessitates fast and efficient neural decoding algorithms to ensure timely and seamless interaction between the nervous system and the iBCI. Controlling a robotic prosthesis requires a latency of less than 150 ms between the neural activity and arm movement for smooth and natural control [37], which is close to the biological delay for signal propagation between the brain and arm [18]. However, immediate feedback is crucial for the neural decoder to adjust the control system in real-time, allowing subjects to adapt and refine their actions in real-time. In this closed-loop iBCI, a much lower latency of less than 10 ms is necessary to enable inter-areal interactions for decoding sensory and motor tasks [7, 8].

Benchmarking neural decoders for online *in vivo* iBCIs is critical to ensure their optimal performance within the resource-constrained en-

vironment of implantable systems. By evaluating various decoders based on their fidelity, latency, and power consumption, researchers can identify the most suitable options that satisfy clinical safety requirements and ensure effective real-time operation, ultimately improving the efficacy of closed-loop neuromodulation. However, traditional benchmarks [10, 38, 39] predominantly emphasize the accuracy and fidelity of neural decoding methods. A recent addition, NeuroBench [9], expanded this focus to assess algorithm-hardware co-optimization, incorporating fidelity, efficiency, and performance metrics. While NeuroBench is well-suited for evaluating the operating cost of neural decoders, its algorithmic benchmark provides limited insights into power and latency, primarily relying on the effective operational cost as a proxy for hardware metrics. NeuroBench's system benchmark allows for the co-optimization of hardware and software; however, this track has not been fully deployed. This paper presents methods to extrapolate software-to-hardware metrics and allows to benchmark neural decoders for iBCIs for CLN.

3. Metrics

Traditionally, accuracy has been the primary metric used to assess the performance of neural decoders. However, in hardware-constrained iBCIs, other factors, such as latency and power consumption, are crucial to ensure the tractability of neural decoders for applications such as CLN. Evaluating decoders solely based on task performance often fails to capture their true potential and limitations. This section introduces the need for comprehensive metrics for algorithmic evaluation that encompass fidelity (*i.e.*, accuracy), latency, power consumption, and memory size for neural decoders. Table 1 presents an overview of the proposed metrics.

3.1. Model Fidelity

The fidelity of a neural decoder in an iBCI is its ability to correctly classify or predict. In closed-loop iBCI, making accurate predictions is crucial for effective and reliable control of external devices or neural stimulations, which should closely reflect the subject's intention or state.

Classification metrics, such as accuracy, have traditionally been used to assess neural decoding performance by determining the percentage of correct classifications of stimuli, such as odors [1], faces [40], or speech [2, 41]. However, these metrics do not consider the temporal continuity of neural data, which is critical for many neural decoding applications. Neural data require metrics that incorporate the correction of temporal regression to evaluate the decoding accuracy. In such cases, the coefficient of determination (R^2) [3, 10, 17, 27, 42–44] and the coefficient of correlation (Pearson's r) [42, 44–47] are widely used to assess the performance of neural decoding algorithms. The coefficient of determination measures the proportion of variance in the dependent variable explained by the model's prediction, while the coefficient of correlation evaluates the temporal alignment of the predictions and labels via a linear relationship. Both metrics should be reported to comprehensively assess the temporal regression performance of the model.

3.2. Latency

The latency of the neural decoder, defined as the time delay between the first input stimulus and the output response [48], comprises two architecture-specific subcomponents: the *binning latency* and the *processing latency*, as shown in Figure 2c. Assessing the latency using the *binning latency* and *processing latency* allows for a platform-agnostic evaluation of neural decoders.

The *binning latency* corresponds to the time span of the input data required for each prediction. This equates to the binning time window or the history of binning windows in the case of multiple windows. Minimizing the binning window size is crucial for limiting the total latency; however, longer windows allow decoders to better estimate the firing rate of neurons [17].

The *processing latency* is caused by the processing time of the neural decoder to produce a prediction. This combines the operational delays of preprocessing, network inference, and postprocessing. The processing time is a function of the system's required *effective* operations per inference, which can be assessed by computing the *effective* multiply-and-accumulate (MAC) operations of neural decoder algorithms. MAC operations are the core operations in the matrix-vector multiplication of the Perceptron, which comprises standard artificial neural networks (ANNs). One MAC operation corresponds to one inner product in matrix-vector multiplications and is the core unit optimized by a vector accelerator. Reporting the number of MAC operations considers that the inner products are optimized in hardware, and only their quantity needs to be minimized by the algorithm designer. Owing to their binary input data, binary neural networks (BNNs) and SNNs can forego MACs with a more computationally efficient accumulation (ACC).

Latency can be reported in wall time, such as absolute SI units [48] or relative system time, as the total number of clock cycles per inference. This paper reports latency in milliseconds, providing a more intuitive and user-centric perspective and allowing the reader to assess the system's ability to

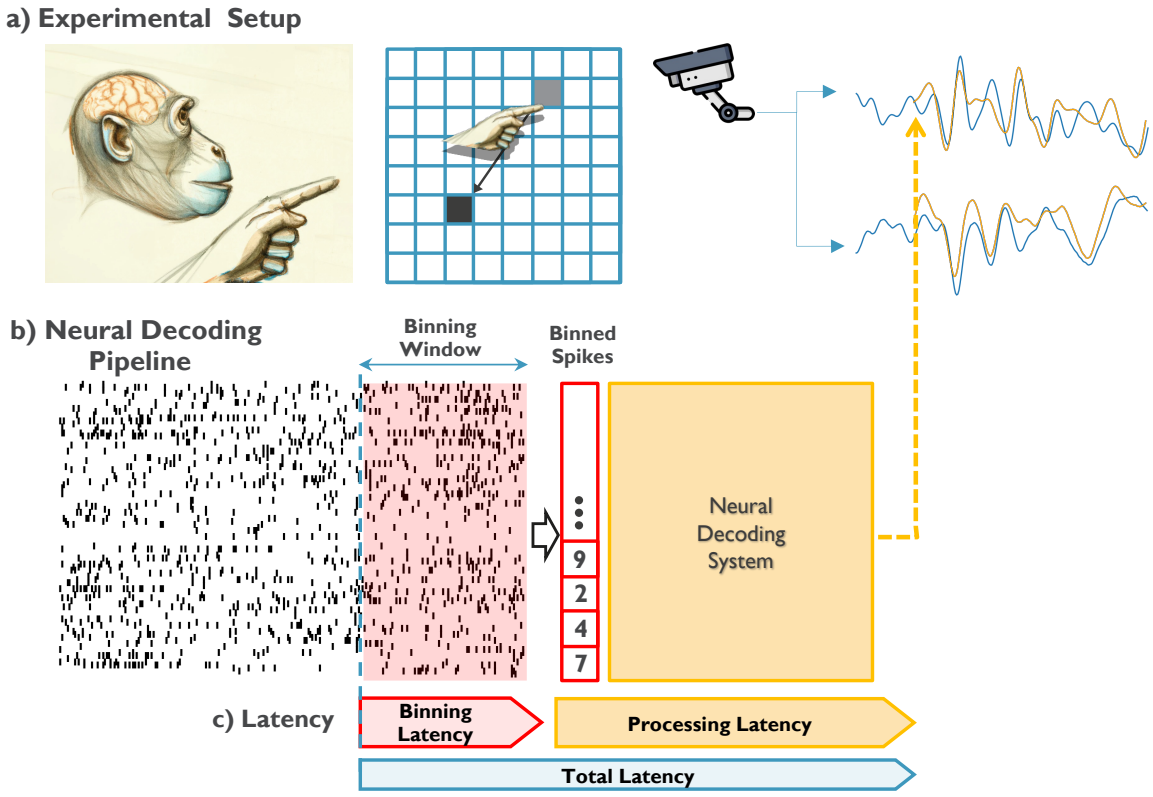


Figure 3: Experimental pipeline for evaluating closed-loop capability of neural decoders. a) A primate consecutively reaches for black boxes on a grid. Neural activity is recorded using one or two UTAH arrays placed at the primary motor and sensorimotor cortices, while finger position is tracked through an electromagnetic position sensor. Finger velocity is computed and used as a decoding target. b) Neural activity is processed as a spike train, binned, and processed by the neural decoding system to predict finger velocity. c) The visualization illustrates the latency of the entire decoding system, composed of the *binning latency* and the *processing latency*.

deliver timely and accurate responses. Converting the *processing latency* into seconds requires platform-specific assumptions regarding the necessary clock cycles per operation, the clock frequency, and the system's capability for parallel processing. For the remainder of this paper, a clock frequency of 1 MHz and 3 MAC operations per clock cycle were assumed [49]. For simplicity, a single addition is assumed to be equivalent to an MAC operation in terms of the required clock cycles.

3.3. Power Consumption

Power consumption is vital for evaluating neural decoders, particularly in resource-constrained iBCIs suitable for CLN. Local neural processing, which is implemented directly on an embedded device, is usually preferable to external processing because of latency, communication bandwidth, and privacy issues. However, local neural processing requires low energy consumption to minimize tissue heating for maintaining neural cell health.

To compare the energy efficiency of different neural decoders in a hardware- and architecture-independent manner, one can benchmark with two hardware-agnostic metrics: total *effective* operations and memory accesses. This considers only algorithmic optimizations, such as reducing the *effective* operational costs. Although algorithm-hardware co-optimizations can further improve latency, performance, and energy efficacy, they require binding the neural decoder to specific hardware and, thus, are not considered in this study.

The total *effective* computational cost is reported as the number of non-zero operations required per inference. This combines two relevant operations that dominate the neural network operations: multiplication and addition. In line with previous work on the relative energy cost of various operations, it is assumed that one multiplication is three times as costly as one addition [50]. To estimate the energy consumption of a neural decoder, total operations should be reported instead of MAC and ACC because MACs are assumed to be optimized by vector accelerators, which is the bottleneck for latency but does not map to the energy cost of a neural decoder. To minimize the en-

ergy consumption, neural decoders can exploit the sparsity of spiking data by distinguishing between *effective* and *ineffective* operations. This accounts for only non-zeros contributing to the products and, consequently, the accumulation, which can be leveraged by specialized hardware. Reporting the *effective* computational cost as the number of non-zero operations allows for a hardware-agnostic comparison of different networks, considering computational primitives in neuromorphic neural networks. In the remainder of this paper, MAC denotes *effective* MAC operations.

Reporting memory access is crucial for comprehensively estimating the energy consumption of neural decoders. Because a read-and-write operation to the memory requires one to two orders of magnitude more energy than the operations of arithmetic linear units (ALUs) [48, 50], it is insufficient to report only *effective* operations without including the memory access. Liao *et al.* [46] reported 10× more reads than *effective* operations during inference, highlighting that most of the energy consumption of an architecture comes from the memory read-and-write operations. Following their approach, the number of memory accesses in a network is conservatively estimated by assuming that an MAC operation consists of three loads and one store. By contrast, an ACC consists of two loads and one store [46].

3.4. Memory Size

Owing to the space volume constraints of iBCIs, the memory size, which in comparison to other function blocks consumes far more application-specific integrated circuit (ASIC) area, should be reported. If the memory requirements of the neural decoder are too large, external memory is required, which can consume more than 100× more energy than on-chip memory [51]. Estimating the memory footprint of a neural decoder involves calculating the sum of the network's parameters and variables, as well as the memory requirements of the input data from the binning windows. The memory size should be reported in bits to provide credit to architectures that limit the precision of weights and other variables.

4. Method

The methodology is structured into five subsections, addressing how this study evaluates and benchmarks various neural decoders suitable for iBCI in the context of CLN. Subsection 4.1 provides an overview of the neural data utilized in this benchmark, which comprises spiking neural data recorded from non-human primates (NHP), representing a relevant scenario for closed-loop iBCI applications. Subsections 4.2, 4.3, and 4.4 introduce the reviewed neural decoders. Decoders are categorized into three main groups: traditional neural decoders, artificial neural network (ANN)-based decoders, and neuromorphic spiking neural network (SNN)-based decoders. The selection of traditional and ANN decoder was based on the existing literature, whereas the long short-term memory (LSTM) and SNN-based decoders were optimized as part of this study. Subsection 4.5 outlines the experimental setup and details the conditions and parameters under which the benchmark was conducted. Together, these methodological components provide a detailed pipeline for evaluating neural decoding methods for closed-loop iBCI applications.

4.1. Neural Recording Dataset

The ANN and SNN neural decoders were trained and benchmarked on the "primate reaching task" of the neuromorphic benchmark NeuroBench [9], as shown in Figures 3a and 3b. This benchmark employed the same data as Makin *et al.* [17], which were previously used to evaluate their traditional decoders. The R2 value published by Makin *et al.* [17] is reported here, and the metrics for power consumption and latency are calculated conservatively based on the most time-consuming and energy-intensive matrix operations, as specified in Subsection 4.2 and Subsection 4.3.

The dataset is a subset of six out of 33 recording sessions of two Rhesus primates during subsequent reaching tasks, as shown in Figure 3a, which was released by Dyer *et al.* [52]. These sessions encompassed two NHPs and the entire recording period. Three NHP *I* sessions were recorded with a 96-channel Utah array from Blackrock Neurotech, implanted in the primary motor cortex. The three NHP *L* sessions had an additional Utah array in the sensorimotor cortex, enabling the simultaneous recording of 192 channels. Each recording session comprised 354 to 819 individual reaches, and those longer than 8 s were discarded as the longer timespan indicated that the primate was inattentive and the neural data flawed.

Spikes were detected from the raw neural data using a threshold of 3.5 to 4 times the root-mean-square (RMS) noise. The finger position was recorded using an electromagnetic position sensor at 250Hz, and the velocity was computed as the discrete gradient of the position. Predicting the translation-invariant finger velocity better aligns with the natural dynamics of limb movements and corresponds to how neural activity encodes kinematics [53]. Makin *et al.* [17] and Dyer *et al.* [52] reported the detailed experimental setup and data acquisition procedures.

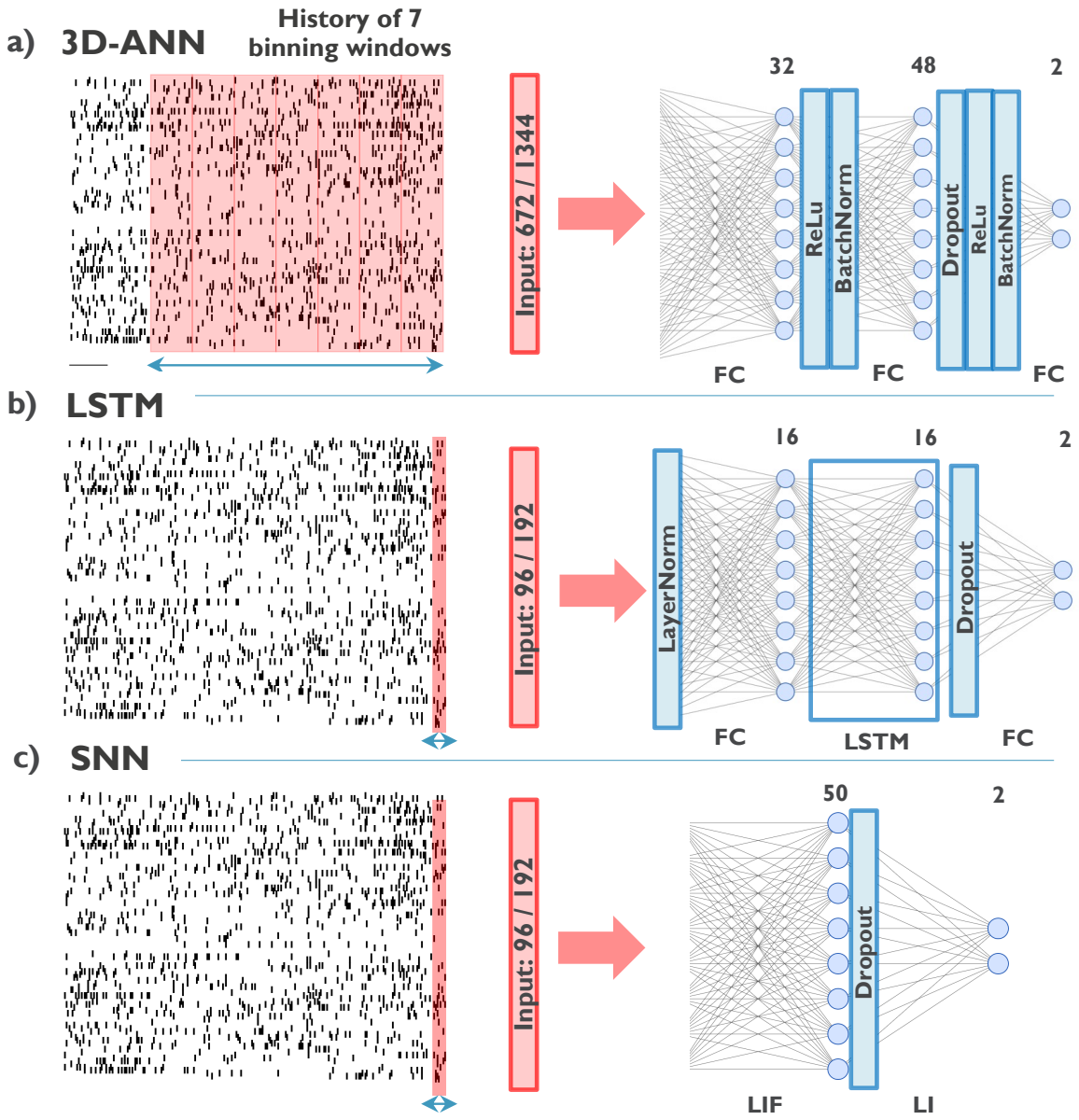


Figure 4: Architecture of three neural network-based decoders. The data extraction and binning are visualized in red, and the network architecture in blue. Each layer's dimensions and type are stated above and below, respectively. a) The ANN uses seven binning windows of 28 milliseconds as input. These extracted windows are flattened and processed by two hidden layers with 32 and 48 Neurons, respectively to produce a two-dimensional output. b) The LSTM extracts spikes with a temporal resolution of 4 milliseconds, effectively representing the neural data as a spike train. Following this step, the data undergoes dimensionality reduction via a fully connected (FC) layer with 16 neurons, after which a Long Short-Term Memory (LSTM) cell is employed. A final FC layer produces the two-dimensional output c) Similarly, the SNN decoder is designed to extract the spike train, processing the data through a network featuring 50 hidden Leaky Integrate-and-Fire (LIF) neurons. The final output of this network comprises the membrane potential of two LIF neurons.

4.2. Traditional Neural Decoders

Classic neural decoding methods have been extensively used in brain-computer interfaces. Of the six decoders Makin *et al.* [17] presented, the three best-performing models were selected to represent traditional neural decoders. Those decoders are the 'unscented' Kalman Filter (UKF), the static, and the dynamic 'recurrent exponential-family harmonium' (rEFH), and they are evaluated due to their established performance and versatility in handling various neural data modalities [6, 10, 17]. Makin *et al.* [17] also considered Linear Regression, conventional Kalman Filter (KF), and Wiener Filter (WF). However, because these decoders had worse R2 performance and scaled poorly with de-

creasing binning windows, they were not considered suitable for this neural decoding benchmark for closed-loop iBCIs suitable for CLN.

The UKF approximates the state distribution by applying a full nonlinearity to a minimal set of carefully selected representative points. It was designed as an extension of the KF to address the problem of its exploding residuals on the true posterior mean and covariance. The UKF solves this problem by replacing the normal sampled state distribution with deterministic sampling of this distribution [54]. The UKF, as described by Makin *et al.* [17] and Wan and Van Der Merwe [54], has a state space of deterministically sampled 40 variables, which requires $O(n^3/6)$ operations owing to the Cholesky decomposition. As a conservative estimate of the *processing latency*, total effective operations, and memory accesses, only the matrix multiplications required during inference and computationally intensive matrix inversion were considered. This large-scale matrix inversion was assumed to require at least 200 ms [55].

The rEFH, instead of assuming Gaussian state variables, models the state variables as a variant of a restricted Boltzmann Machine (RBM) [56] and explicitly samples the spike count from a Poisson distribution. The static variant converts the latent space of the RBM into kinematics via static mapping, that is, a matrix multiplication, whereas the dynamic version uses a KF. For the static and dynamic rEFH models, only the forward pass of the RBM uses higher-dimensional matrix multiplications, and is thus considered. There are four times as many hidden neurons in the RBM than there are input channels and 1800 output neurons mapped to the kinematic output via either matrix multiplication or a KF [17]. All traditional decoders were evaluated using binning windows of 16 ms, 32 ms, 64 ms, and 128 ms.

4.3. Artificial Neural Networks (ANN)

This study used two artificial neural network (ANN)-based decoders as baselines owing to their established history of high-accuracy predictions. Previous studies have demonstrated that ANN neural decoders consistently outperform traditional decoders [10, 46]. One fully connected ANN, published as a baseline for the NeuroBench benchmark [9], and one LSTM network were evaluated.

The ANN is a conventional 3-layer feedforward network with 32 and 48 hidden and two output neurons, respectively. This implementation uses a history of multiple non-overlapping binning windows that are flattened and processed as input data. The default setup, including the history of the seven binning windows, is shown in Figure 4a. To explore the latency versus fidelity trade-off, a history of 4, 7, and 14 binning windows of 28 ms was considered, and the number of neurons in the hidden layers was halved and doubled.

The recurrent LSTM, as shown in Figure 4b, uses a fully connected layer to reduce the input dimensionality to 16, followed by a single LSTM cell with 16 hidden neurons. The feedforward layer returns the predicted kinematics. Similar to the ANN, various binning windows of 4 ms, 8 ms, and 16 ms and wider networks with 64 and 128 hidden neurons were examined. Both Networks normalize the data using batch normalization (BN) and layer normalization (LN) and regularize with Dropout.

4.4. Spiking Neural Networks (SNN)

Spiking neural networks are variants of neural networks that attempt to mimic the properties, processes, and functions of biological neurons. This makes them inherently recurrent and allows them to exploit sparsity to achieve lower latency [19, 57–64] and power consumption [3, 46, 57–61].

SNNs at their core consist of stateful spiking neurons, a more bio-plausible variant of the perceptron, with leaky integrate-and-fire (LIF) being the most widely used neuron model. As the Perceptron, the LIF weighs and accumulates the input, but instead of returning this weighted accumulation, it is added to the neuron's membrane potential. If the membrane potential exceeds a threshold, the neuron produces a binary output, that is, a spike, and the membrane potential is reset; otherwise, the membrane potential decays per time step, as conceptually illustrated in Figure 5. This enables the neuron to combine information over multiple time steps, making it inherently recurrent. In addition to their binary output, LIF neurons exploit binary input data, enabling sparsity in networks built around these neurons. The binary input separates the operations into effective and ineffective because multiplications by zero do not contribute to the accumulation. Therefore, the multiplication of weights times input can be forgone by adding only weights with non-zero input.

The implemented neuromorphic SNN decoder is a simplified version of the model proposed by Liao *et al.* [46] with fewer learnable parameters. In a preliminary exploration, reducing the complexity of the network only marginally impacted the accuracy while significantly improving the operational cost. Each hidden layer was decreased to 50 LIF-neurons without a bias term and fixed decay ($\tau = 0.96$). The BN layer is removed because combining BN and Dropout leads to models with different feature variances inside the network during training and testing [65]. The threshold of LIF

neurons was set to one. To further push for lower latency and power consumption and explore the various tradeoffs against accuracy, the number of hidden layers was decreased from three to two and one (*SNN3*, *SNN2*, and *SNN1*, respectively). *SNN1* is a baseline model for the "primate reaching task" of NeuroBench [9] and is depicted in Figure 4c.

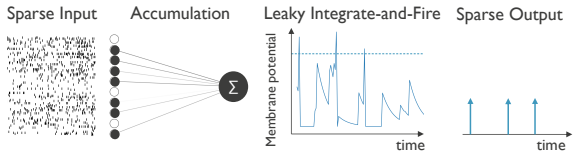


Figure 5: A scheme of the LIF neuron. In this neuron model, incoming neural signals are integrated into the membrane potential, and if the accumulated value surpasses a specified threshold, the neuron generates a binary output signal. The "leaky" property allows for the gradual decay of the accumulated charge, further contributing to the model's simplicity. The intrinsic recurrent nature of the LIF neuron originates in the updated membrane potential.

4.5. Experimental Setup

The spike train was binned according to the window requirements of the respective network, and a sliding window with a stride of 4 ms extracted the spiking neural data. The finger velocity was selected as the target label as shown in Figure 3b.

Predicting a single data point from a time series can amplify the vanishing gradient and dead-neuron problems. Therefore, the SNN was trained to predict the time window of the primate's finger kinematics and, for consistency, was tested to predict individual finger velocities as the ANNs. The Loss for this setup was a linearly weighted mean squared error (MSE) from zero to one to account for the warm-up steps. The ANN and LSTM were trained using the conventional MSE Loss. All neural networks were optimized using the PyTorch automatic backpropagation engine.

Each session was divided into the first 75 percent of the reaches shuffled and, in contrast to the Neurobench-proposed evaluation pipeline, was used for model selection using 10-fold group cross-validation (CV) with early stopping and the last 25 percent for testing. This allows finding the optimal hyperparameters and number of training epochs while reporting the average performance instead of potential outliers.

5. Results

We considered six decoders trained on reconstructing a primate's finger velocity given binned neural activity and evaluated them with metrics that allow evaluating of a decoder's latency and power consumption. Table 2 provides a complete overview of the performance of all decoders.

5.1. Latency vs. Fidelity

A higher R^2 performance can typically be achieved using deeper and more complex architectures or by better estimating the neural firing rate with a longer binning window. Both approaches negatively impact the latency and, thus, the capability of the neural decoder to facilitate real-time closed-loop feedback. This performance tradeoff is visualized for the six decoders in Figure 6.

The UKF requires computationally intense matrix operations and a matrix inverse, making it significantly slower than most other decoders and achieving lower fidelity. In contrast to the rEFH filters, the R^2 score improved with decreasing binning windows. The best-performing rEFH filter had an R^2 of 63.19% (std=0.17) with a latency of 129 ms, significantly outperforming the UKF with an R^2 of 45.10% (std=0.12) and a latency of 270 ms both in terms of fidelity and latency. The rEFH filters can achieve comparable R^2 scores to NN-based decoders. However, they require long binning windows to approximate the state distribution and experience a stark drop in the R^2 scores with smaller binning windows. Nevertheless, the trends of the rEFH and ANN models indicate that the rEFH can achieve a better latency versus fidelity tradeoff than the ANN. This stems from the ANN requiring a considerable history of binning windows to extract temporal information, which results in a high latency. For the ANN, reducing the number of binning windows led to a significant decrease in fidelity. The shallow LSTM attains a much lower latency versus fidelity tradeoff than the traditional decoders and ANNs achieving peak R^2 scores above 60% while having a latency between 4.05 ms and 16.05 ms. Notably, the fidelity drops significantly when using a longer binning window of 16 ms, indicating that the LSTM relies on the temporal dimension of the neural data to extract information about neural dynamics. The SNN achieved the best trade-off with competitive R^2 scores and lower latency for all three models examined. The latency scales only marginally with an increase in the number of layers.

The overall trend of the six decoders shows that recurrent neural networks, such as LSTM and SNN, can achieve competitive fidelity to non-recurrent

Decoder type		Fidelity				Latency			Power Consumption		Size
		R2	$\pm\sigma$	r	$\pm\sigma$	Bin (ms)	MAC	Total (ms)	Eff. Ops.	Memory Access	Memory Foot print
Traditional Decoders	UKF	0.4510*	0.15	-	-	16	28,799	269.6	23M	116k	736Mb
		0.4485*	0.16	-	-	32					
		0.4290*	0.17	-	-	64					
		0.4122*	0.32	-	-	128					
	rEFH static	0.2103*	0.17	-	-	16	3,131	17	7M	12k	224Mb
		0.3879*	0.20	-	-	32		33			
		0.5042*	0.17	-	-	64		65			
		0.5961*	0.14	-	-	128		129			
	rEFH dynamic	0.4248*	0.18	-	-	16	3,167	17.1	7M	13k	224Mb
		0.4996*	0.17	-	-	32		33.1			
		0.6038*	0.14	-	-	64		65.1			
		0.6319*	0.14	-	-	128		129.1			
ANN neural decoders	ANN (H7)	0.6369	0.06	0.8023	0.03	196	162	196.1	34k	136k	1Mb
	ANN (H4)	0.5768	0.05	0.7615	0.03	112		112.1	20k	80k	647kb
	ANN (H14)	**0.6515**	0.09	**0.8159**	0.04	392		392.1	66k	265k	2Mb
	ANN (2x)	0.6403	0.06	0.8051	0.03	196	322	196.1	71k	285k	2Mb
	ANN (0.5x)	0.6300	0.05	0.7989	0.03	196	82	196.1	17k	67k	532kb
	LSTM (B4)	0.6014	0.04	0.7781	0.03	4	146	**4.05**	14k	69k	611kb
	LSTM (B8)	0.6391	0.05	0.8027	0.04	8	146	8.05			
	LSTM (B16)	0.5026	0.15	0.7128	0.10	16	146	16.05			
	LSTM (2x)	0.6036	0.05	0.7784	0.03	4	578	4.19	53k	224k	2Mb
	LSTM (4x)	0.6109	0.06	0.7834	0.04	4	1154	4.38	162k	659k	5Mb
ANN neural decoders	SNN1	0.5948	0.06	0.7723	0.04	4	167	**4.05**	**167**	**1503**	**158kb**
	SNN2	0.6292	0.06	0.7968	0.04	4	202	4.07	202	1815	240kb
	SNN3	0.6071	0.08	0.7861	0.05	4	305	4.08	229	2068	322kb

Table 2: Experimental results of 6 neural decoders for decoding finger velocities. Six recording sessions of primates executing reaching tasks were selected and split into the first 75 percent of the reaches for 10-fold cross-validation and the final 25 percent for testing. The traditional decoders were evaluated with different binning windows. Five variants of the ANN were investigated: The default ANN implementation (*H7*), also used in NeuroBench [9], had seven binning windows of 28ms each. *H4* and *H14* had 4 and 14 binning windows, significantly affecting the decoder's *binning latency*. The default decoder's width was doubled to produce a more expressive neural network (*2x*) and halved to minimize the operational cost (*0.5x*). The LSTM was evaluated using binning windows of 4, 8, and 16 milliseconds (*B4*, *B8*, *B16*). To examine the scalability of the low-latency decoder, its LSTM cell was increased by *2x* and *4x*. Three different SNNs are reported with a varying number of hidden layers (*SNN1*, *SNN2*, *SNN3*). Notably, the latency of the UKF is consistently high, given the delay caused by the computationally intensive matrix inverse.
*The R2 results for the UKF and the rEFH were reported by Makin *et al.* [17].

decoders while extracting neural dynamics from intra-cortical spike data and having significantly lower latency, with SNNs maintaining lower latencies for higher fidelity than LSTMs.

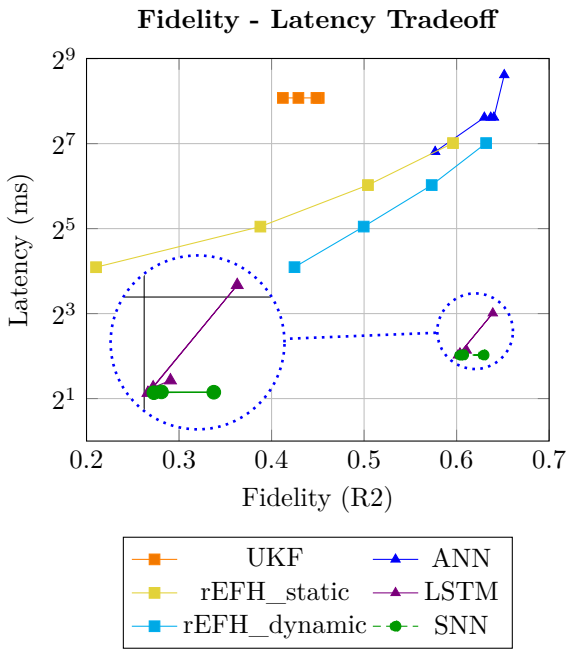


Figure 6: The fidelity versus latency trade-off of the six evaluated decoders. The R2 fidelity is plotted on the horizontal axis, and the vertical axis is the millisecond latency. In the plot, traditional decoders are represented as squares, artificial neural network (ANN)-based decoders as triangles, and spiking neural networks (SNNs) as circles. The decoders with the best trade-off are in the bottom right corner. Recurrent decoders such as LSTM and SNN achieve substantially lower latency while maintaining competitive R2 scores.

5.2. Power Consumption vs. Fidelity

MACs, reported as the number of inner products of matrix-vector multiplications, are indifferent to the length of the vectors in the inner product. This makes them inadequate for assessing power consumption. Instead, the total number of required operations during inference provides a better estimate of the actual energy cost. Figure 7 shows the tradeoff between power consumption and fidelity of the examined decoders.

All traditional decoders require matrix operations with large inner products, leading to, by far, the most operations to effectively decode neural signals. The average number of operations for the traditional decoders across the experiments was between 700,000 and 2,300,000. The three decoders have the same high number of total operations across the various binning windows because unlike the other models, the length of the binning window does not affect the operational cost. Traditional decoders had the most extensive range of R^2 scores, ranging from 23% to 66%. NN-based decoders represent a shift towards more computationally efficient decoding algorithms. Our experiments demonstrated significantly reduced power consumption compared to traditional decoders. The LSTM with 16 hidden neurons required 4700 operations, whereas the ANN with seven binned windows required approximately 34,000 operations. Compared to traditional decoders, this significant improvement comes with consistently high fidelity, with R^2 values of ANN-based decoders ranging from 57% to 65%. The SNN decoders exhibited the lowest power consumption among the decoder types considered in this study. On average, the SNN decoders required 200 operations while achieving competitive fidelity levels, with R^2 values ranging from 60% to 63%.

Comparing memory access instead of total effective operations reveals the same tradeoff and is not reiterated. A comparison of the power consumption and fidelity metrics reveals an intriguing tradeoff among the three decoder types. While traditional decoders require substantial operations, they offer a range of fidelity levels. In contrast, ANN-based and SNN-based decoders provide the advantage of reduced power consumption, with SNN decoders exhibiting the lowest computational load while maintaining competitive fidelity levels.

5.3. Bayesian Information Criterion (BIC)

The Bayesian Information Criterion (BIC) serves as a valuable instrument for comparing various neural networks, effectively addressing the concern of increased model complexity and its potential impact on performance enhancement. The BIC introduces a penalization term for the number of model parameters, thereby discouraging the adoption of overly complex models with excessive weights and biases. This penalty term effectively balances fidelity and model complexity, enabling us to discern whether a model's performance improvements stem from an increased number of learnable parameters or from architectural design.

Figure 8 shows the BIC values for the various configurations of the three neural network (NN)-based decoders. Notably, the single-layer SNN, characterized by its minimal complexity, attained the lowest BIC score. In contrast, SNN2 and SNN3 had significantly higher BIC scores despite exhibiting performance improvements. This discrepancy suggests that the performance improvements are disproportional to the increased number of learnable parameters in these models. All SNNs achieved much lower BIC scores than the traditional and ANN-based neural decoders.

6. Discussion

In conclusion, optimizing neural decoders for iBCI systems capable of CLN presents a delicate balance, requiring careful consideration of the trade-offs between fidelity, latency, power consumption, and memory size. Our findings emphasize that although more complex and deeper neural architectures with more trainable parameters hold the potential for improved decoding accuracy, optimizing only for fidelity by increasing the complexity of the network can result in reduced usability for closed-loop iBCIs.

The decoding accuracy reaffirms the findings of Glaser *et al.* [10] that conventional neural network-based decoders can achieve the highest R^2 scores. However, we observed that this comes at the expense of increased latency and power consumption. Even when only considering fidelity, evaluating the BIC across the three NN-based decoders showed that SNNs consistently outperformed the ANN and the LSTM, achieving significantly lower BIC scores. This indicates that the performance improvement of the ANN is due to disproportionally more learnable parameters. Remarkably, the single-layer SNN emerged as the top performer out of the models we benchmarked in this paper, signifying its suitability for effectively learning data variance, particularly when considering the number of learnable parameters. This highlights that the shallow SNN is preferable for robust and energy-efficient neural decoding, given its complexity among the three neural network-based decoder types we evaluated.

The ability of a neural decoder to effectively harness sparsity is a crucial design consideration in closed-loop iBCI systems. Conventional neural data are characterized by their inherent sparsity and temporal encoding [18, 32, 66, 67], with rate-based encoding accounting for a mere fraction of the neural activity in regions such as the visual cortex [68]. The inherent sparsity of neural spikes provides SNNs with a distinct advantage, enabling them to capitalize on the spatiotemporal structure of the input data, which is less pronounced in non-neuromorphic ANNs.

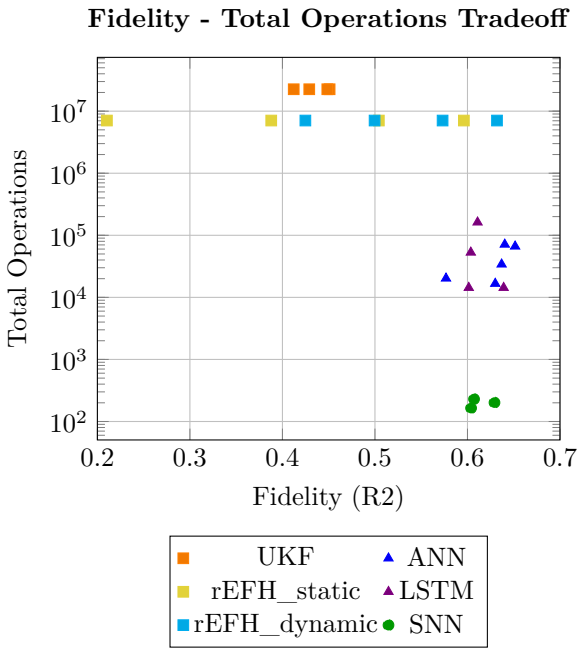


Figure 7: Visualized trade-off between R2 fidelity and total operations. The specific closed-loop setting requires neural decoders in the bottom right corner. ANN-based decoders, such as the ANN and the LSTM depicted as triangles, are an improvement compared to traditional decoders, represented as squares. Yet, the SNN, denoted by circles, achieves the lowest operational cost while maintaining competitive R2 scores. Comparing memory access instead of total operations shows the same trend and is redundant.

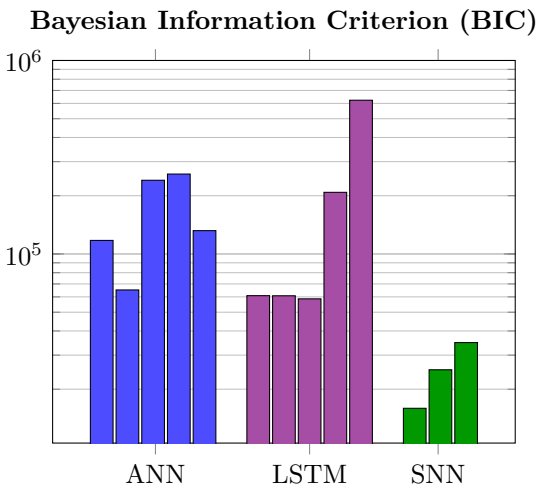


Figure 8: The Bayesian information criterion (BIC) of the evaluated NN-based decoder shows that the SNN achieves far lower BIC scores than other decoders, suggesting that SNNs are more suitable for the constrained closed-loop setting. A low BIC indicates that a decoder can learn better, given its complexity. The lowest BIC score is achieved by *SNN1*, which is one baseline in NeuroBench [9]. The various models appear in the same order as presented in table 2.

SNNs have previously demonstrated their potential for reduced power consumption and low latency in various applications. In our study, we reaffirm and extend these prior findings, indicating that SNNs can extract neural dynamics from extracellular spiking data while maintaining competitive fidelity and showing superior performance in terms of power consumption and latency.

The latency metric introduced in our analysis operates under the assumption that the computation of an inner product is equivalent to a single addition in terms of clock cycles. Although

this abstraction aligns with standard practices for MAC operations, it is essential to acknowledge that this assumption may only partially represent potential hardware optimizations for vector accumulations. While the process of counting additions might initially appear as a potential disadvantage when comparing SNNs to ANNs in the context of latency, we observed that despite this methodological abstraction, SNNs consistently achieved substantially lower latency when compared to traditional decoders and ANN. The longer latency of traditional decoders and ANNs is primarily attributed to their reliance on long binning windows to extract temporal information and their high operational costs. Notably, the sole exception to this trend is the LSTM, which demonstrates a latency level comparable to that of SNNs. This observation reaffirms the findings of Zenke *et al.* [66], who demonstrated the efficiency of recurrent neural networks (RNNs), such as LSTMs and SNNs, in exploiting the temporal structure of neural data, emphasizing their capacity to achieve competitive fidelity with low latency in a closed-loop neural decoding system. These insights underscore that even without accounting for potential hardware optimizations, SNNs can exhibit a marked advantage in terms of latency over traditional decoder models and non-recurrent ANNs, which require more extensive computational resources owing to their temporal information extraction procedures.

The advantages of employing neuromorphic SNNs for closed-loop iBCIs are more apparent when the energy cost is evaluated using the total number of operations. In this context, SNNs significantly outperform traditional and ANN-based decoders. Our results show that traditional and ANN-based decoders require several orders of magnitude more operations to attain performance levels comparable to those of SNNs. The remarkable reduction in the number of operations required by SNNs can be attributed to their constrained design, which minimizes the number of learnable parameters while still delivering competitive performance. However, the primary driving forces behind the substantially lower energy cost associated with SNNs are their ability to exploit sparsity [46] and their more energy-efficient operations. This capacity allows for approximately 5% of the operations to be executed, emphasizing the exceptional efficiency achieved with SNNs while maintaining high decoding accuracy. We observed an estimated energy consumption of approximately 2 µW for the SNNs [50, 69], which was 50 times lower than that of the LSTM, 100 times lower for the ANN, and 10,000 times lower than that of the UKF.

The field of neurotechnology and BCIs are expanding rapidly. New advancements and technologies enable the development of more effective, user-friendly, and versatile BCIs. This paper discusses two main challenges in designing processors for implantable closed-loop neural decoders: low energy consumption to minimize heat diffusion, and low latency to enable real-time closed-loop neuromodulation. We defined metrics for neural decoders and benchmarked common decoding methods to predict a primate's finger kinematics from the motor cortex. This study explores the suitability of low latency and low computing neural decoders and highlights the potential advantages of neuromorphic SNNs for closed-loop neuromodulation (CLN). Our results show that SNNs can balance decoding accuracy and operational efficiency, offering immense potential for reshaping the landscape of neural decoders and opening new frontiers in closed-loop intra-cortical human-brain interaction.

Using neuromorphic SNNs for CLN is an area of research with great promise, as indicated by successfully predicting fine motor kinematics of NHPs in this study. However, the list of evaluated decoders is non-exhaustive, and only a single neural decoder, the SNN, was optimized for latency and power consumption. This allows for the comparison of commonly used decoders, yet it is biased towards efficient and fast neuromorphic decoders. Additionally, only one exemplary dataset was evaluated as representative of closed-loop iBCI tasks requiring low latency and power consumption. Therefore, this study only highlights the potential of SNN as an neural decoder for iBCI for CLN, however, further studies are required to explore the suitability of these networks for other types of neural decoding tasks and to optimize their performance to meet the requirements of CLN systems. Developing fully implantable iBCIs with local processing capabilities is crucial for reducing energy consumption and latency and improving the real-time applicability of CLN systems.

Overall, the outlook for neural engineering and BCIs is bright. New developments will improve neural recording and decoding technologies, ultimately enhancing our understanding of the brain and its complex neural processes.

7. Conclusion

Our study introduces methods to extrapolate algorithmic-to-hardware metrics that allow evaluating the low latency and high energy efficacy requirements of iBCI suitable for CLN. We present six commonly used neural decoders and compare them in predicting NHP fine motor kinematics from binned neural activity. Our results high-

light the potential advantages of the neuromorphic SNNs as a neural decoder for iBCIs capable of CLN. In our benchmark, we observed that SNNs outperform other commonly used decoders, with evident performance differences when compared to benchmarked traditional decoders. Notably, the exceptionally low latency of SNNs and LSTMs, surpassing that of traditional decoders and non-recurrent neural networks, arises from their innate ability to extract temporal information from spiking neural data. The power efficiency can be attributed the adeptness of SNNs in exploiting sparsity and their deliberately constrained architectural designs. Our results show that SNNs can achieve competitive decoding performance in less than 5 ms, using less than 1% of computational resources and more than 50 times less energy than other neural decoding methods in this benchmark. This makes them highly suitable candidates for closed-loop iBCI challenges and positions them as a game-changing technology for reshaping the landscape of neural decoders. Significant advancements in CLN can be achieved by adopting SNNs as the preferred neural decoder. Their capacity for efficient and accurate neural signal processing has the potential to revolutionize BCI applications, enhancing our ability to interact with and understand the intricacies of the human brain.

References

[1] Ming Li et al. "Odor Recognition with a Spiking Neural Network for Bioelectronic Nose". en. In: *Sensors* 19.5 (Feb. 2019), p. 993. ISSN: 1424-8220. DOI: 10.3390/s19050993. URL: http://www.mdpi.com/1424-8220/19/5/993 (visited on 07/12/2023).

[2] David A. Moses et al. "Neuroprosthesis for Decoding Speech in a Paralyzed Person with Anarthria". en. In: *New England Journal of Medicine* 385.3 (July 2021), pp. 217–227. ISSN: 0028-4793, 1533-4406. DOI: 10.1056/NEJMoa2027540. URL: http://www.nejm.org/doi/10.1056/NEJMoa2027540 (visited on 07/12/2023).

[3] Jerry Tang et al. "Semantic reconstruction of continuous language from non-invasive brain recordings". en. In: *Nature Neuroscience* 26.5 (May 2023), pp. 858–866. ISSN: 1097-6256, 1546-1726. DOI: 10.1038/s41593-023-01304-9. URL: https://www.nature.com/articles/s41593-023-01304-9 (visited on 08/29/2023).

[4] Andy Zhou, Benjamin C Johnson, and Rikky Muller. "Toward true closed-loop neuromodulation: artifact-free recording during stimulation". en. In: *Current Opinion in Neurobiology* 50 (June 2018), pp. 119–127. ISSN: 09594388. DOI: 10.1016/j.conb.2018.01.012. URL: https://linkinghub.elsevier.com/retrieve/pii/S095943881730243X (visited on 10/24/2023).

[5] Stavros Zanos. "Closed-Loop Neuromodulation in Physiological and Translational Research". en. In: *Cold Spring Harbor Perspectives in Medicine* 9.11 (Nov. 2019), a034314. ISSN: 2157-1422. DOI: 10.1101/cshperspect.a034314. URL: http://perspectivesinmedicine.cshlp.org/lookup/doi/10.1101/cshperspect.a034314 (visited on 10/24/2023).

[6] Julie Dethier et al. "Design and validation of a real-time spiking-neural-network decoder for brain–machine interfaces". In: *Journal of Neural Engineering* 10.3 (June 2013), p. 036008. ISSN: 1741-2560, 1741-2552. DOI: 10.1088/1741-2560/10/3/036008. URL: https://iopscience.iop.org/article/10.1088/1741-2560/10/3/036008 (visited on 10/22/2023).

[7] Davide Ciliberti, Frédéric Michon, and Fabian Kloosterman. "Real-time classification of experience-related ensemble spiking patterns for closed-loop applications". en. In: *eLife* 7 (Oct. 2018), e36275. ISSN: 2050-084X. DOI: 10.7554/eLife.36275. URL: https://elifesciences.org/articles/36275 (visited on 10/23/2023).

[8] Iraklis Petrof, Angela N. Viaene, and S. Murray Sherman. "Properties of the primary somatosensory cortex projection to the primary motor cortex in the mouse". en. In: *Journal of Neurophysiology* 113.7 (Apr. 2015), pp. 2400–2407. ISSN: 0022-3077, 1522-1598. DOI: 10.1152/jn.00949.2014. URL: https://www.physiology.org/doi/10.1152/jn.00949.2014 (visited on 09/01/2023).

[9] Jason Yik et al. "NeuroBench: Advancing Neuromorphic Computing through Collaborative, Fair and Representative Benchmarking". In: (2023). Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2304.04640. URL: https://arxiv.org/abs/2304.04640 (visited on 09/12/2023).

[10] Joshua I. Glaser et al. "Machine Learning for Neural Decoding". en. In: *eneuro* 7.4 (July 2020), ENEURO.0506–19.2020. ISSN: 2373-2822. DOI: 10.1523/ENEURO.0506-19.2020. URL: https://www.eneuro.org/lookup/doi/10.1523/ENEURO.0506-19.2020 (visited on 08/28/2023).

[11] E V Evarts. "Pyramidal tract activity associated with a conditioned hand movement in the monkey." en. In: *Journal of Neurophysiology* 29.6 (Nov. 1966), pp. 1011–1027. ISSN: 0022-3077, 1522-1598. DOI: 10.1152/jn.1966.29.6.1011. URL: https://www.physiology.org/doi/10.1152/jn.1966.29.6.1011 (visited on 10/22/2023).

[12] Nicholas A. Steinmetz et al. "Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings". en. In: *Science* 372.6539 (Apr. 2021), eabf4588. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.abf4588. URL: https://www.science.org/doi/10.1126/science.abf4588 (visited on 10/19/2023).

[13] David A Schwarz et al. "Chronic, wireless recordings of large-scale brain activity in freely moving rhesus monkeys". en. In: *Nature Methods* 11.6 (June 2014), pp. 670–676. ISSN: 1548-7091, 1548-7105. DOI: 10.1038/nmeth.2936. URL: https://www.nature.com/articles/nmeth.2936 (visited on 10/19/2023).

[14] Joseph T. Marmerstein, Grant A. McCallum, and Dominique M. Durand. "Decoding Vagus-Nerve Activity with Carbon Nanotube Sensors in Freely Moving Rodents". en. In: *Biosensors* 12.2 (Feb. 2022), p. 114. ISSN: 2079-6374. DOI: 10.3390/bios12020114. URL: https://www.mdpi.com/2079-6374/12/2/114 (visited on 10/19/2023).

[15] Mikhail A Lebedev et al. "Future developments in brain-machine interface research". en. In: *Clinics* 66 (Jan. 2011), pp. 25–32. ISSN: 18075932. DOI: 10.1590/S1807-59322011001300004. URL: https://linkinghub.elsevier.com/retrieve/pii/S1807593222015812 (visited on 10/22/2023).

[16] Mikhail A. Lebedev and Miguel A.L. Nicolelis. "Toward a whole-body neuroprosthetic". en. In: *Progress in Brain Research*. Vol. 194. Elsevier, 2011, pp. 47–60. ISBN: 978-0-444-53815-4. DOI: 10.1016/B978-0-444-53815-4.00018-2. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780444538154000182 (visited on 10/22/2023).

[17] Joseph G Makin et al. "Superior arm-movement decoding from cortex with a new, unsupervised-learning algorithm". In: *Journal of Neural Engineering* 15.2 (Apr. 2018), p. 026010. ISSN: 1741-2560, 1741-2552. DOI: 10.1088/1741-2552/aa9e95. URL: https://iopscience.iop.org/article/10.1088/1741-2552/aa9e95 (visited on 08/28/2023).

[18] Daniel W. Moran and Andrew B. Schwartz. "Motor Cortical Representation of Speed and Direction During Reaching". en. In: *Journal of Neurophysiology* 82.5 (Nov. 1999), pp. 2676–2692. ISSN: 0022-3077, 1522-1598. DOI: 10.1152/jn.1999.82.5.2676. URL: https://www.physiology.org/doi/10.1152/jn.1999.82.5.2676 (visited on 09/01/2023).

[19] Kaushalya Kumarasinghe, Nikola Kasabov, and Denise Taylor. "Brain-inspired spiking neural networks for decoding and understanding muscle activity and kinematics from electroencephalography signals during hand movements". en. In: *Scientific Reports* 11.1 (Jan. 2021), p. 2486. ISSN: 2045-2322. DOI: 10.1038/s41598-021-81805-4. URL: https://www.nature.com/articles/s41598-021-81805-4 (visited on 09/01/2023).

[20] Guy Hotson et al. "High Precision Neural Decoding of Complex Movement Trajectories Using Recursive Bayesian Estimation With Dynamic Movement Primitives". In: *IEEE Robotics and Automation Letters* 1.2 (July 2016), pp. 676–683. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2016.2516590. URL: https://ieeexplore.ieee.org/document/7378310/ (visited on 10/20/2023).

[21] Yunzhe Liu et al. "Decoding cognition from spontaneous neural activity". en. In: *Nature Reviews Neuroscience* 23.4 (Apr. 2022), pp. 204–214. ISSN: 1471-003X, 1471-0048. DOI: 10.1038/s41583-022-00570-z. URL: https://www.nature.com/articles/s41583-022-00570-z (visited on 10/20/2023).

[22] Qianli Yang et al. "Revealing nonlinear neural decoding by analyzing choices". en. In: *Nature Communications* 12.1 (Nov. 2021), p. 6557. ISSN: 2041-1723. DOI: 10.1038/s41467-021-26793-9. URL: https://www.nature.com/articles/s41467-021-26793-9 (visited on 10/20/2023).

[23] Yuanrui Dong et al. "Neural Decoding for Intracortical Brain–Computer Interfaces". en. In: *Cyborg and Bionic Systems* 4 (Jan. 2023), p. 0044. ISSN: 2692-7632. DOI: 10.34133/cbsystems.0044. URL: https://spj.science.org/doi/10.34133/cbsystems.0044 (visited on 10/22/2023).

[24] M. F. Mridha et al. “Brain-Computer Interface: Advancement and Challenges”. en. In: *Sensors* 21.17 (Aug. 2021), p. 5746. ISSN: 1424-8220. DOI: 10.3390/s21175746. URL: https://www.mdpi.com/1424-8220/21/17/5746 (visited on 10/22/2023).

[25] Christian Klaes. “Invasive Brain-Computer Interfaces and Neural Recordings From Humans”. en. In: *Handbook of Behavioral Neuroscience*. Vol. 28. Elsevier, 2018, pp. 527–539. ISBN: 978-0-12-812028-6. DOI: 10.1016/B978-0-12-812028-6.00028-8. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780128120286000288 (visited on 10/22/2023).

[26] Katherine W. Scangos et al. “Closed-loop neuromodulation in an individual with treatment-resistant depression”. en. In: *Nature Medicine* 27.10 (Oct. 2021), pp. 1696–1700. ISSN: 1078-8956, 1546-170X. DOI: 10.1038/s41591-021-01480-w. URL: https://www.nature.com/articles/s41591-021-01480-w (visited on 10/24/2023).

[27] Daniel Valencia and Amir Alimohammad. “Towards in vivo neural decoding”. en. In: *Biomedical Engineering Letters* 12.2 (May 2022), pp. 185–195. ISSN: 2093-9868, 2093-985X. DOI: 10.1007/s13534-022-00217-z. URL: https://link.springer.com/10.1007/s13534-022-00217-z (visited on 09/01/2023).

[28] Yuxuan Luo et al. “A 74- W 11-Mb/s Wireless Vital Signs Monitoring SoC for Three-Lead ECG, Respiration Rate, and Body Temperature”. In: *IEEE Transactions on Biomedical Circuits and Systems* 13.5 (Oct. 2019), pp. 907–917. ISSN: 1932-4545, 1940-9990. DOI: 10.1109/TBCAS.2019.2922295. URL: https://ieeexplore.ieee.org/document/8735805/ (visited on 10/20/2023).

[29] Hyejung Kim et al. “A Configurable and Low-Power Mixed Signal SoC for Portable ECG Monitoring Applications”. In: *IEEE Transactions on Biomedical Circuits and Systems* 8.2 (Apr. 2014), pp. 257–267. ISSN: 1932-4545, 1940-9990. DOI: 10.1109/TBCAS.2013.2260159. URL: https://ieeexplore.ieee.org/document/6544283/ (visited on 10/20/2023).

[30] Yuming He et al. “An Implantable Neuromorphic Sensing System Featuring Near-Sensor Computation and Send-on-Delta Transmission for Wireless Neural Sensing of Peripheral Nerves”. In: *IEEE Journal of Solid-State Circuits* 57.10 (Oct. 2022), pp. 3058–3070. ISSN: 0018-9200, 1558-173X. DOI: 10.1109/JSSC.2022.3193846. URL: https://ieeexplore.ieee.org/document/9858104/ (visited on 10/20/2023).

[31] John D. Simeral et al. “Home Use of a Percutaneous Wireless Intracortical Brain-Computer Interface by Individuals With Tetraplegia”. In: *IEEE Transactions on Biomedical Engineering* 68.7 (July 2021), pp. 2313–2325. ISSN: 0018-9294, 1558-2531. DOI: 10.1109/TBME.2021.3069119. URL: https://ieeexplore.ieee.org/document/9390339/ (visited on 10/20/2023).

[32] Nir Even-Chen et al. “Power-saving design opportunities for wireless intracortical brain–computer interfaces”. en. In: *Nature Biomedical Engineering* 4.10 (Aug. 2020), pp. 984–996. ISSN: 2157-846X. DOI: 10.1038/s41551-020-0595-9. URL: https://www.nature.com/articles/s41551-020-0595-9 (visited on 08/28/2023).

[33] Karthik Sriram et al. “SCALO: An Accelerator-Rich Distributed System for Scalable Brain-Computer Interfacing”. en. In: *Proceedings of the 50th Annual International Symposium on Computer Architecture*. Orlando FL USA: ACM, June 2023, pp. 1–20. ISBN: 9798400700958. DOI: 10.1145/3579371.3589107. URL: https://dl.acm.org/doi/10.1145/3579371.3589107 (visited on 10/22/2023).

[34] Ian H Stevenson and Konrad P Kording. “How advances in neural recording affect data analysis”. en. In: *Nature Neuroscience* 14.2 (Feb. 2011), pp. 139–142. ISSN: 1097-6256, 1546-1726. DOI: 10.1038/nn.2731. URL: https://www.nature.com/articles/nn.2731 (visited on 10/22/2023).

[35] Patrick D. Wolf. “Thermal Considerations for the Design of an Implanted Cortical Brain–Machine Interface (BMI)”. eng. In: *Indwelling Neural Implants: Strategies for Contending with the In Vivo Environment*. Ed. by William M. Reichert. Frontiers in Neuroengineering. Boca Raton (FL): CRC Press/Taylor & Francis, 2008. ISBN: 978-0-8493-9362-4. URL: http://www.ncbi.nlm.nih.gov/books/NBK3932/ (visited on 10/22/2023).

[36] Sohee Kim et al. “Thermal Impact of an Active 3-D Microelectrode Array Implanted in the Brain”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 15.4 (Dec. 2007), pp. 493–501. ISSN: 1534-4320, 1558-0210. DOI: 10.1109/TNSRE.

2007.908429. URL: https://ieeexplore.ieee.org/document/4359231/ (visited on 10/22/2023).

[37] Meel Velliste et al. "Cortical control of a prosthetic arm for self-feeding". en. In: *Nature* 453.7198 (June 2008), pp. 1098–1101. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature06996. URL: https://www.nature.com/articles/nature06996 (visited on 09/01/2023).

[38] Felix Pei et al. "Neural Latents Benchmark '21: Evaluating latent variable models of neural population activity". In: (2021). Publisher: arXiv Version Number: 4. DOI: 10.48550/ARXIV.2109.04463. URL: https://arxiv.org/abs/2109.04463 (visited on 10/30/2023).

[39] Peter Mattson et al. "MLPerf: An Industry Standard Benchmark Suite for Machine Learning Performance". In: *IEEE Micro* 40.2 (Mar. 2020), pp. 8–16. ISSN: 0272-1732, 1937-4143. DOI: 10.1109/MM.2020.2974843. URL: https://ieeexplore.ieee.org/document/9001257/ (visited on 10/30/2023).

[40] Rufin VanRullen and Leila Reddy. "Reconstructing faces from fMRI patterns using deep generative neural networks". en. In: *Communications Biology* 2.1 (May 2019), p. 193. ISSN: 2399-3642. DOI: 10.1038/s42003-019-0438-y. URL: https://www.nature.com/articles/s42003-019-0438-y (visited on 10/20/2023).

[41] Debadatta Dash, Paul Ferrari, and Jun Wang. "Decoding Imagined and Spoken Phrases From Non-invasive Neural (MEG) Signals". In: *Frontiers in Neuroscience* 14 (Apr. 2020), p. 290. ISSN: 1662-453X. DOI: 10.3389/fnins.2020.00290. URL: https://www.frontiersin.org/article/10.3389/fnins.2020.00290/full (visited on 10/20/2023).

[42] Chethan Pandarinath et al. "Inferring single-trial neural population dynamics using sequential auto-encoders". en. In: *Nature Methods* 15.10 (Oct. 2018), pp. 805–815. ISSN: 1548-7091, 1548-7105. DOI: 10.1038/s41592-018-0109-9. URL: https://www.nature.com/articles/s41592-018-0109-9 (visited on 08/28/2023).

[43] Sonia Todorova et al. "To sort or not to sort: the impact of spike-sorting on neural decoding performance". In: *Journal of Neural Engineering* 11.5 (Oct. 2014), p. 056005. ISSN: 1741-2560, 1741-2552. DOI: 10.1088/1741-2560/11/5/056005. URL: https://iopscience.iop.org/article/10.1088/1741-2560/11/5/056005 (visited on 10/15/2023).

[44] 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". en. In: *Nature Biomedical Engineering* 4.10 (July 2020), pp. 973–983. ISSN: 2157-846X. DOI: 10.1038/s41551-020-0591-0. URL: https://www.nature.com/articles/s41551-020-0591-0 (visited on 08/28/2023).

[45] Elijah Taeckens, Ryan Dong, and Sahil Shah. *A Biologically Plausible Spiking Neural Network for Decoding Kinematics in the Hippocampus and Premotor Cortex*. en. preprint. Bioengineering, Nov. 2022. DOI: 10.1101/2022.11.09.515838. URL: http://biorxiv.org/lookup/doi/10.1101/2022.11.09.515838 (visited on 08/28/2023).

[46] Jiawei Liao et al. "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)*. Incheon, Korea, Republic of: IEEE, June 2022, pp. 134–137. ISBN: 978-1-66540-996-4. DOI: 10.1109/AICAS54282.2022.9869846. URL: https://ieeexplore.ieee.org/document/9869846/ (visited on 07/20/2023).

[47] Simin Li, Jie Li, and Zheng Li. "An Improved Unscented Kalman Filter Based Decoder for Cortical Brain-Machine Interfaces". In: *Frontiers in Neuroscience* 10 (Dec. 2016). ISSN: 1662-453X. DOI: 10.3389/fnins.2016.00587. URL: http://journal.frontiersin.org/article/10.3389/fnins.2016.00587/full (visited on 10/15/2023).

[48] Vivienne Sze et al. *Efficient Processing of Deep Neural Networks*. en. Synthesis Lectures on Computer Architecture. Cham: Springer International Publishing, 2020. ISBN: 978-3-031-00638-8 978-3-031-01766-7. DOI: 10.1007/978-3-031-01766-7. URL: https://link.springer.com/10.1007/978-3-031-01766-7 (visited on 07/20/2023).

[49] Zhong Liu et al. "Optimizing convolutional neural networks on multi-core vector accelerator". en. In: *Parallel Computing* 112 (Sept. 2022), p. 102945. ISSN: 01678191. DOI: 10.1016/j.parco.2022.102945. URL:

https://linkinghub.elsevier.com/retrieve/pii/S0167819122000424 (visited on 09/29/2023).

[50] Mark Horowitz. "1.1 Computing's energy problem (and what we can do about it)". In: *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2014, pp. 10–14. ISBN: 978-1-4799-0920-9 978-1-4799-0918-6. DOI: 10.1109/ISSCC.2014.6757323. URL: http://ieeexplore.ieee.org/document/6757323/ (visited on 07/20/2023).

[51] Yu-Pei Liang et al. "Brief Industry Paper: An Energy-Reduction On-Chip Memory Management for Intermittent Systems". In: *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. Nashville, TN, USA: IEEE, May 2021, pp. 429–432. ISBN: 978-1-66540-386-3. DOI: 10.1109/RTAS52030.2021.00044. URL: https://ieeexplore.ieee.org/document/9470471/ (visited on 10/27/2023).

[52] Eva L. Dyer et al. "A cryptography-based approach for movement decoding". en. In: *Nature Biomedical Engineering* 1.12 (Dec. 2017), pp. 967–976. ISSN: 2157-846X. DOI: 10.1038/s41551-017-0169-7. URL: https://www.nature.com/articles/s41551-017-0169-7 (visited on 10/26/2023).

[53] Wei Wang et al. "Motor Cortical Representation of Position and Velocity During Reaching". en. In: *Journal of Neurophysiology* 97.6 (June 2007), pp. 4258–4270. ISSN: 0022-3077, 1522-1598. DOI: 10.1152/jn.01180.2006. URL: https://www.physiology.org/doi/10.1152/jn.01180.2006 (visited on 10/23/2023).

[54] E.A. Wan and R. Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. Lake Louise, Alta., Canada: IEEE, 2000, pp. 153–158. ISBN: 978-0-7803-5800-3. DOI: 10.1109/ASSPCC.2000.882463. URL: http://ieeexplore.ieee.org/document/882463/ (visited on 09/17/2023).

[55] V.Y. Pan, F. Soleymani, and L. Zhao. "An efficient computation of generalized inverse of a matrix". en. In: *Applied Mathematics and Computation* 316 (Jan. 2018), pp. 89–101. ISSN: 00963003. DOI: 10.1016/j.amc.2017.08.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S0096300317305611 (visited on 09/27/2023).

[56] James L McClelland, David E Rumelhart, PDP Research Group, et al. *Parallel distributed processing, volume 2: Explorations in the microstructure of cognition: Psychological and biological models*. Vol. 2. MIT press, 1987.

[57] Jason K. Eshraghian et al. "Training Spiking Neural Networks Using Lessons From Deep Learning". In: (2021). Publisher: arXiv Version Number: 6. DOI: 10.48550/ARXIV.2109.12894. URL: https://arxiv.org/abs/2109.12894 (visited on 09/01/2023).

[58] Amirhossein Tavanaei et al. "Deep learning in spiking neural networks". en. In: *Neural Networks* 111 (Mar. 2019), pp. 47–63. ISSN: 08936080. DOI: 10.1016/j.neunet.2018.12.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S0893608018303332 (visited on 09/01/2023).

[59] Julian Göltz et al. "Fast and energy-efficient neuromorphic deep learning with first-spike times". In: *Nature Machine Intelligence* 3.9 (Sept. 2021). arXiv:1912.11443 [cs, q-bio, stat], pp. 823–835. ISSN: 2522-5839. DOI: 10.1038/s42256-021-00388-x. URL: http://arxiv.org/abs/1912.11443 (visited on 09/01/2023).

[60] Peter U. Diehl et al. "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing". In: *2015 International Joint Conference on Neural Networks (IJCNN)*. Killarney, Ireland: IEEE, July 2015, pp. 1–8. ISBN: 978-1-4799-1960-4. DOI: 10.1109/IJCNN.2015.7280696. URL: http://ieeexplore.ieee.org/document/7280696/ (visited on 09/01/2023).

[61] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. "Learning to be efficient: algorithms for training low-latency, low-compute deep spiking neural networks". en. In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. Pisa Italy: ACM, Apr. 2016, pp. 293–298. ISBN: 978-1-4503-3739-7. DOI: 10.1145/2851613.2851724. URL: https://dl.acm.org/doi/10.1145/2851613.2851724 (visited on 09/01/2023).

[62] Nitin Rathi and Kaushik Roy. "DIET-SNN: A Low-Latency Spiking Neural Network With D irect I nput E ncoding and Leakage and T hreshold Optimization". In: *IEEE Transactions on Neural Networks and Learn-*

ing Systems 34.6 (June 2023), pp. 3174–3182. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2021.3111897. URL: https://ieeexplore.ieee.org/document/9556508/ (visited on 09/01/2023).

[63] Matthew S. Willsey et al. "Real-time brain-machine interface in non-human primates achieves high-velocity prosthetic finger movements using a shallow feedforward neural network decoder". en. In: *Nature Communications* 13.1 (Nov. 2022), p. 6899. ISSN: 2041-1723. DOI: 10.1038/s41467-022-34452-w. URL: https://www.nature.com/articles/s41467-022-34452-w (visited on 06/26/2023).

[64] Shoeb Shaikh et al. *Towards Intelligent Intracortical BMI (i ^{2} BMI): Low-power Neuromorphic Decoders that outperform Kalman Filters*. en. preprint. Bioengineering, Sept. 2019. DOI: 10.1101/772988. URL: http://biorxiv.org/lookup/doi/10.1101/772988 (visited on 08/29/2023).

[65] Xiang Li et al. "Understanding the Disharmony Between Dropout and Batch Normalization by Variance Shift". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 2677–2685. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00279. URL: https://ieeexplore.ieee.org/document/8953671/ (visited on 09/19/2023).

[66] Friedemann Zenke et al. "Visualizing a joint future of neuroscience and neuromorphic engineering". en. In: *Neuron* 109.4 (Feb. 2021), pp. 571–575. ISSN: 08966273. DOI: 10.1016/j.neuron.2021.01.009. URL: https://linkinghub.elsevier.com/retrieve/pii/S089662732100009X (visited on 09/01/2023).

[67] Wolfgang Maass. "Networks of spiking neurons: The third generation of neural network models". en. In: *Neural Networks* 10.9 (Dec. 1997), pp. 1659–1671. ISSN: 08936080. DOI: 10.1016/S0893-6080(97)00011-7. URL: https://linkinghub.elsevier.com/retrieve/pii/S0893608097000117 (visited on 09/01/2023).

[68] Bruno A. Olshausen and David J. Field. "What Is the Other 85 Percent of V1 Doing?" en. In: *23 Problems in Systems Neuroscience*. Ed. by J. Leo Van Hemmen and Terrence J. Sejnowski. 1st ed. Oxford University Press-New York, Jan. 2006, pp. 182–212. ISBN: 978-0-19-514822-0 978-0-19-986467-6. DOI: 10.1093/acprof:oso/9780195148220.003.0010. URL: https://academic.oup.com/book/4769/chapter/147027016 (visited on 09/01/2023).

[69] Guangzhi Tang et al. "Open the box of digital neuromorphic processor: Towards effective algorithm-hardware co-design". In: (2023). Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2303.15224. URL: https://arxiv.org/abs/2303.15224 (visited on 09/01/2023).

A. Defintions

Mathematical Notations

The mathematical notation follows the convention of representing

a	scalars as lowercase letters
$\boldsymbol{v}$	vectors as bold lowercase letters
W	matrices as capitalized letters

Neural Network Notations

The following notation describes the labeled data used in this study

X	dataset
y	labels
o	predictions of a model

B. Neuromodulation

Neuromodulation is a therapeutic approach that involves the targeted delivery of electrical, chemical, or other agents to specific neurological sites in the body to modulate the activity of neural circuits. This method allows for regulating or atlering neural activity to restore normal function or alleviate symptoms associated with various neurological disorders.

Open-Loop versus Closed-Loop

Distinguishing between open-loop neuromodulation (OLN) and closed-loop neuromodulation (CLN), as shown in Figure 1, involves understanding the control mechanisms of the intervention. In OLN, the stimulation is delivered according to a pre-defined, fixed pattern without direct feedback from the subject's neural activity. This approach is practical for some conditions but lacks adaptability to real-time changes in the subject's physiological state. In contrast, CLN incorporates real-time feedback from the subject's neural signals to dynamically adjust the parameters of the stimulation. This responsive and adaptive nature allows for personalized and more precise interventions, making

CLN particularly promising for optimizing therapeutic outcomes in conditions where neural dynamics vary over time [1].

Neuromodulation Applications

Two successful applications of neuromodulation techniques are deep brain stimulation (DBS) and vagus nerve stimulation (VNS).

DBS involves implants that emit electrical impulses to modulate neural activity and has been shown to mitigate the effects of symptoms of Parkinson's Disease [2, 3]. Traditionally, DBS delivers fixed electrical stimulation, i.e., open-loop, irrespective of the subject's brain activity. However, DBS can benefit from being implemented in a closed-loop circuit [1].

VNS is a medical procedure that involves the implantation of a device, typically a small pulse generator, into the body to stimulate the vagus nerve. The vagus nerve is a major component of the autonomic nervous system, and it plays a crucial role in regulating various bodily functions, including heart rate, digestion, and respiratory processes. VNS has been approved by regulatory agencies for specific medical conditions, such as drug-resistant epilepsy (DRE) [4, 5], which are epileptic seizure symptoms antiepileptic drugs cannot mitigate. The procedure is reversible [6], and it has been shown that closed-loop VNS more effectively reduces the frequency and severity of seizures compared to open-loop VNS [1, 7, 8].

C. Metrics

This section defines the detailed formulas for computing the reported metrics.

Fidelity

Fidelity, which in this study describes the accuracy of predictions made by neural decoders, is crucial in many applications. It evaluates how well a model captures and replicates the underlying patterns in the data it is trained on. Two common metrics used to assess model fidelity are the coefficient of determination (R^2) and the coefficient of correlation (Pearson's r).

The R^2 is a statistical measure that assesses the proportion of the variance in the dependent variable $\boldsymbol{y}$ that is predictable from the independent variable(s) X by a regression model. In essence, R^2 quantifies the goodness of fit of a model by indicating the percentage of variability in y that can be explained by X. It ranges from $-\inf$ to 1, where $R^2 = 1$ indicates that the model perfectly predicts the dependent variable, $R^2 = 0$, signifies that the model fails to explain any variability. A model which performs worse than a simple mean-based model has a negative $R^2 < 0$.

The coefficient of determination (R^2) is defined as

$$SS_{res} = \sum_i (y_i - o_i)^2 \tag{1}$$

$$SS_{tot} = \sum_i (y_i - \bar{y}_i)^2 \tag{2}$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{3}$$

with the output of the model $\boldsymbol{o}$, the dependent variable $\boldsymbol{y}$, and its average $\bar{y} = 1/n \sum_i (y_i)$.

The Pearson's r, is a statistical measure that quantifies the strength and direction of a linear relationship between two variables, here the model's output o and the label y. Ranging from -1 to 1, a positive Pearson's r indicates a positive linear correlation, meaning that as one variable increases, the other also tends to increase. Conversely, a negative Pearson's r signifies a negative linear correlation, suggesting that as one variable increases, the other tends to decrease. A value of 0 indicates no linear correlation. The magnitude of Pearson's r reflects the degree of correlation, with 1 or -1 signifying a perfect linear relationship, while values closer to 0 represent weaker associations.

The coefficient of correlation (r) is defined as

$$r(o, y) = \frac{\sum_i (o_i - \bar{o})^2 (y_i - \bar{y})^2}{\sqrt{\sum_i (o_i - \bar{o})^2} \sqrt{\sum_i (y_i - \bar{y})^2}} \tag{4}$$

with the dependent variable $\boldsymbol{y}$ and it's mean $\bar{y} = 1/n \sum_i (y_i)$ and the output of the model $\boldsymbol{o}$ with the average prediction $\bar{o} = 1/n \sum_i (o_i)$.

Latency

The latency *lat* reported in this study comprises two parts: the *binning latency* lat_{bin} and the *processing latency* lat_{proc}.

The total latency is reported as

$$lat = lat_{bin} + \frac{lat_{proc}}{c_{freq}} \tag{5}$$

where the *processing latency* is $lat_{proc} = \mathrm{MAC}/r$ and the multiply-and-accumulates (MACs) are the number of effective MAC operations, assuming one addition is equivalent to one MAC. The clocking frequency c_{freq} is assumed to be 1 MHz, and it is assumed that $r = 3$ MAC operations can be processed per clock cycle.

The number of MAC operations counts the inner products required by the equations specified in

Appendix D, E, and F. This means that the MACs of the following operations were defined as

$$\boldsymbol{a}^T\boldsymbol{b} = 1 \tag{6}$$
$$\boldsymbol{a}B = m \tag{7}$$
$$AB = mo \tag{8}$$

where $\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times o}$

The sparsity was only included for the spiking neural network (SNN) since the other neural decoders could not to exploit sparse data. To convert the number of additions in an SNN into the effective number of MACs, operations were multiplied by the sparsity. The sparsity, defined as the ratio of how frequently the neuron in a layer produced binary outputs, was multiplied layer-wise with the respective number of additions.

Power Consumption

The number of total *effective* operations *Ops* distinguishes two core operations, addition and multiplication (*mul*, *add*), and was computed as

$$Ops = mul + 3add \tag{9}$$

The total operations for the equations specified in Appendix D, E, and F were computed using

$$\boldsymbol{a}^T\boldsymbol{b} = (n, n-1) \tag{10}$$
$$\boldsymbol{a}B = (no, (n-1)o) \tag{11}$$
$$AB = (nmo, (n-1)mo) \tag{12}$$

where $\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times o}$. This was combined into one value using the assumption that multiplications are three times as intensive in the context of energy consumption [9].

$$\boldsymbol{a}^T\boldsymbol{b} = n + 3(n-1) \tag{13}$$
$$\boldsymbol{a}B = no + 3(n-1)o \tag{14}$$
$$AB = nmo + 3(n-1)mo) \tag{15}$$

This was computed precisely for the neural networks (NNs) and. The operational complexity of traditional decoders was conservatively bounded by their most intensive operations due to a lack of insight into the precise implementation.

To convert the number of total operations into the effective number of total operations, the were multiplied by the sparsity. The sparsity, the ratio of how frequently the neuron in a layer produced binary outputs, was multiplied layer-wise with the respective number of operations.

The memory accesses M_{access} are computed as

$$M_{access} = 4mul + 3add \tag{16}$$

Model Size

The model size was computed as the sum of the size of every learnable parameter. The size of a vector $\boldsymbol{a} \in \mathbb{R}^n$ was size$(a) = n$ and size$(A) = nm$ for matrix $A \in \mathbb{R}^{n\times m}$.

D. Traditional Decoders

The landscape of neural decoding has been dominated by traditional methods for decades, with the Kalman Filter (KF) serving until now as a cornerstone in decoding neural signals [10–12]. However, it convinces with its simplicity and comparingly reasonable performance, but has been outperformed by improved methods such as the 'unscented' Kalman Filter (UKF) and the 'recurrent exponential-family harmonium' (rEFH), surpassing the conventional KF in performance. This section provides an overview of the UKF and the rEFH emphasizing the most computationally intensive operations required for inference.

Unscented Kalman Filter

The UKF is a variant of the traditional KF, designed to estimate the state of a nonlinear dynamic system. Unlike the conventional KF, which estimates the state distribution parameters using the expectation-maximation (EM) [13] algorithm, the UKF employs a more sophisticated technique known as the unscented transformation [14]. This transformation involves selecting a minimal set of carefully chosen *sigma* points, capturing the true mean and covariance of the state-space, passing them through a nonlinear transformation, and using these updated points to compute the mean and covariance of the predicted state. This methodology provides a more accurate representation of the system's non-linearities, enhancing the filter's performance in scenarios where the underlying dynamics are nonlinear.

The UKF used in this study was implemented by [15] specifically for tracking the kinematics of non-human primates (NHP). The state space consists of 40 variables comprises hand position, hand velocity, hand acceleration, target position, and a multiplicative interaction of hand position and hand velocity, of 5 bins of 50 ms each.

The most computationally intensive operations are the update of the state space s and the covariance matrix P as defined in Equations (8)-(9) in the Supplementary Material section of [15]. Computing the sigma points requires a matrix square root of a square matrix $m \in \mathbb{R}^{40\times 40}$ via the Cholesky decomposition, as shown in Equations (10)-(12), which requires $40^3/6$ multiplications and additions. The matrix inverse in Equation (17) of

the covariance matrix P is the square of the number of recording channels, *i.e.*, 9,216 and 36,864 for the NHP I and L, respectively. This computation is conservatively estimated to require 200 ms [16]. The final major computations are the update of the KF in Equation (17)-(19).

Recurrent Expontential-Family Harmonium

The rEFH is a type of probabilistic generative model that is a variant of the restricted Boltzmann Machine (RBM). RBMs are unsupervised learning models for feature learning, dimensionality reduction, and density estimation. The rEFH, in particular, extends the traditional exponential-family harmonium (EFH) by including recurrent connections, enabling the learning of temporal dependencies in sequential data.

The EFH is an energy-based model designed for encoding and decoding probability distributions over a set of visible and hidden variables. By introducing recurrent connections in the rEFH, the model gains the ability to capture temporal dependencies in sequential data, making it suitable for applications involving time-series data or sequences. The recurrent connections allow the model to learn and represent patterns that evolve over time, enhancing its capability to generate realistic samples and capture the underlying dynamics of temporal sequences.

The rEFH as implemented by [11] only requires large matrix multiplications in Equation (5) of the forward pass of its RBM. The RBM requires four times as many hidden neurons as input channels and 1800 output neurons, which are mapped to the prediction of the kinematics via a simple matrix multiplication or a dynamic KF.

E. Artificial Neural Networks

Artificial neural networks (ANNs) have experienced significant advancements in recent years, becoming a driving force across various domains such as computer vision, natural language processing, and speech recognition. These breakthroughs in ANNs have prompted researchers in many fields to explore their applicability, such as in neural decoding for brain-computer interfaces (BCIs) [10, 17–19]. This section is structured to introduce ANNs, beginning with the foundational building block, the perceptron, followed by the more complex multi-layer perceptron (MLP) and recurrent neural networks (RNNs). The training paradigm of ANNs will be introduced in Appendix G.

Perceptron

The perceptron, as proposed by Mcculloch and Pitts in 1943 [20], is a binary classifier designed to mimic the decision-making process of a single neuron. It was defined as the linear learnable function

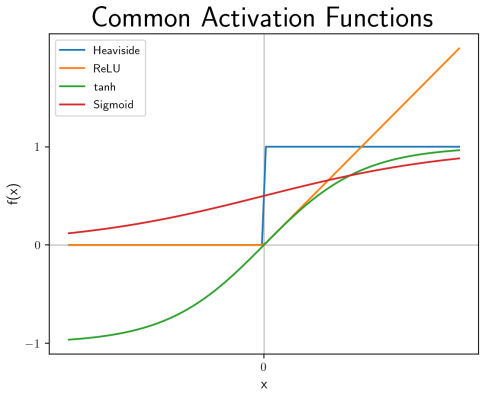


Figure 9: Visualization of four common activation functions used in neural networks: Heaviside step function, Sigmoid, Rectified Linear Unit (ReLU), and Tanh.

$$f(x) = \theta\left(\sum_i w_i x_i + b\right) \tag{17}$$

that maps a range of inputs ($\boldsymbol{x} \in \mathbb{R}^{n\times 1}$) to a single binary output $f(x) \in \mathbb{R}^1$, where the weights $\boldsymbol{w} \in \mathbb{R}^{n\times 1}$ and the bias $b \in \mathbb{R}^1$ are learnable real-valued parameters and θ is the *Heaviside* activation function, as shown in Figure 9. Equation (17) can be simplified as the inner product $\langle w, x\rangle$ or one MAC operation. In matrix notation, this can be re-formulated as

$$f(x) = \theta\left(\boldsymbol{w}^T\boldsymbol{x}\right) \tag{18}$$

with $\boldsymbol{w} \in \mathbb{R}^{(n+1)\times 1}$ and $\boldsymbol{x} \in \mathbb{R}^{(n+1)\times 1}$ extended by $w_{n+1} = b$ and $x_{n+1} = 1$. The remainder of this appendix includes the bias term by default.

The *Heaviside* activation function assumes that the input x is linearly separable, which makes the conventional perceptron insufficient to classify non-linearly separable data correctly. For this reason, the *Heaviside* activation function is often replaced by non-linear activations, with the rectified linear unit (ReLU), defined as $\theta_{ReLU}(x) = max(0, x)$, being the most prominent as shown in Figure 9. Using a non-linear activation function or removing the activation extends the perceptron to regression tasks, where the output is a real continuous scalar.

Fully Connected Neural Network

The perceptron can be extended to m outputs $\boldsymbol{f}(x) \in \mathbb{R}^m$. The non-linear perceptron with multiple outputs is commonly denoted as a fully-connected (FC) layer in NNs and can be defined mathematically as

$$\boldsymbol{f}(x) = \theta\left(W^T \boldsymbol{x}\right) \tag{19}$$

with $W \in \mathbb{R}^{(n+1)\times(m+1)}$ weights. This FC layer is a core building block of conventional ANNs. Multiple FC layers can be stacked to form MLPs or what are widely known as fully connected networks in ANNs. Mathematically stacking 2 FC layers can be defined as

$$\boldsymbol{f}^1(x) = \theta\left(W_1^T \boldsymbol{x}\right) \tag{20}$$
$$\boldsymbol{f}^2(x) = \theta\left(W_2^T \boldsymbol{x}\right) \tag{21}$$
$$\boldsymbol{f}(x) = f^2(f^1(\boldsymbol{x})) \tag{22}$$

with an input $\boldsymbol{x} \in \mathbb{R}^{(n+1)\times 1}$, the weight matrix of the first layer $W_1 \in \mathbb{R}^{(n+1)\times(m+1)}$ and of the second layer$W_2 \in \mathbb{R}^{(m+1)\times(o+1)}$. This NN has n inputs, m hidden perceptrons or *neurons* and o outputs.

Recurrent Neural Networks

Fully-connected networks are ineffective when handling time series data, where data arrives continuously in a stream, as FC layers cannot capture the temporal dependencies of the data. This temporal dimension can contain crucial information and context required to decode the input data successfully. RNNs define NNs that can extract temporal information of the input data by retaining information of previously processed input.

Mathematically, this is formulated as

$$\boldsymbol{h}_t = \theta\left(W_{ih}^T \boldsymbol{x}_t + W_{hh}^T \boldsymbol{h}_{t-1}\right) \tag{23}$$

with $\boldsymbol{x}_t$ the input and $\boldsymbol{h}_t$ the network's state at the current time step t. The state of the network at the previous time step $t-1$ is $\boldsymbol{h}_{t-1}$ and θ is the activation function. The matrix W_{ih} contains the learned weights for the input and W_{hh} defines how previously seen information is incorporate.

Traditional RNNs suffer from vanishing or exploding gradients, which the long short-term memory (LSTM) as introduced by [21] addresses by adding *gates* which allow for a more precise distinction of which temporal information is relevant and needs to be *remembered* and which can be neglected or *forgotten*. Because of its versatility the LSTM is widely used for temporal data. Mathematically it is defined as

$$\boldsymbol{i}_t = \sigma\left(W_{ii}^T \boldsymbol{x}_t + W_{hi}^T \boldsymbol{h}_{t-1}\right) \tag{24}$$
$$\boldsymbol{f}_t = \sigma\left(W_{if}^T \boldsymbol{x}_t + W_{hf}^T \boldsymbol{h}_{t-1}\right) \tag{25}$$
$$\boldsymbol{g}_t = \tanh\left(W_{ig}^T \boldsymbol{x}_t + W_{hg}^T \boldsymbol{h}_{t-1}\right) \tag{26}$$
$$\boldsymbol{o}_t = \sigma\left(W_{io}^T \boldsymbol{x}_t + W_{ho}^T \boldsymbol{h}_{t-1}\right) \tag{27}$$
$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t \tag{28}$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \tag{29}$$

The LSTM comprises four gates: the input gate i, the forget gate f, the cell gate g and the output gate o. Each gate has two weight matrices, which are multiplied by the input x_t and the previous activation h_{t-1}. The cells state c_t of the LSTM at time step t is updated in Equation (28). The forget gate f specifies how much of the previous cell state c_{t-1} should be included (or "remembered") in the updated cell state $\boldsymbol{i}_t \odot \boldsymbol{g}_t$, where $\boldsymbol{i}_t$ is the input and $\boldsymbol{g}_t$ is a candidate cell state. The output gate $\boldsymbol{o}_t$ is multiplied element-wise by cell state $\boldsymbol{c}_t$. The activation σ is the sigmoid function, and $\odot$ is the element-wise product.

F. Spiking Neural Networks

While ANNs have achieved significant milestones in various domains, their performance is far from the capabilities of the biological neural networks of our brains. ANNs excel in tasks such as computer vision [22], speech recognition [23], natural language processing [24], or game playing [25], often surpassing human performance [26]. Yet, they often fall short in areas where the brain demonstrates unparalleled efficiency and adaptability [27]. The brain's ability to operate in real-time and process vast amounts of sensory information simultaneously remains a formidable benchmark that current ANNs struggle to match.

While it is not yet known how precisely the brain encodes information, two key properties of how our brain processes information can give insights into the incredible efficiency of the brain: the binary nature of neurons spiking and sparse activity. In contrast to the continuous signaling in standard ANNs, as shown in Equation (19), biological neurons communicate through discrete spiking events, providing a more energy-efficient and computationally simple mechanism.

The second characteristic of biological neural networks is the phenomenon of sparse activation. Unlike traditional FC ANNs, biological neurons exhibit selective activation, where only a fraction of neurons are active at any given time. This sparsity in activation promotes efficient and selective information representation in the brain.

SNNs aim to replicate and leverage these advantageous characteristics of biological neurons, incorporating binary interactions and sparse activation into their design. By doing so, SNNs strive to enhance the overall efficacy and biological realism of neural network, pushing artificial systems closer to the efficiency and complexity observed in the intricate neural architectures of living organisms.

Leaky Integrate and Fire

The leaky integrate-and-fire (LIF) model is the most prevalent neuron model for gradient-based learning SNNs due to its computational efficiency and ease of training [27]. This model represents a simplified but biologically inspired model of how real neurons integrate and transmit information that includes two properties of biological neurons: binary and sparse spikes. The LIF neuron integrates the binary input X and adds it to the membrane potential u, as shown in Figure 5. This accumulation can be mathematically expressed as WX, where W are the learned weights of the LIF neuron. However, since X is binary, the multiplication can be forgone, reducing the operation to an accumulation of weights of non-zero inputs X. When the membrane potential u surpasses a predefined threshold V_{th}, the neuron "spikes", that is produces a binary output o. One key feature of the LIF neuron is the introduction of a leak τ, which models the gradual dissipation of the membrane potential in the absence of input, mimicking the natural behavior of neuronal membranes in the brain.

Mathematically the LIF can be defined as

$$u_t = f(o_{t-1})u_{t-1} + W^T \boldsymbol{x_t} + b \tag{30}$$
$$o_t = g(u_t) \tag{31}$$

where u_t is the membrane potential and o_t the output at time step t. The input x_t is multiplied by the weights W and the bias b, similar to Equation (19) of an FC layer. The reset function $f(x)$ defines how the memory potential is reset after the neuron spikes and how it decays otherwise. Wu *et al.* [28] define

$$f(o) = \begin{cases} \tau & o = 0 \\ 0 & o = 1 \end{cases} \tag{32}$$

The gate function $g(u)$ defines the spiking behavior of the SNN with regards to the membrane threshold V_{th}

$$g(u) = \begin{cases} 1 & u \geq V_{th} \\ 0 & u < V_{th} \end{cases} \tag{33}$$

Advantages of the LIF Neuron

The LIF neuron provides a suitable compromise between biological plausibility and computational feasibility, making it an effective building block for SNNs. Using accumulation (ACC) instead of MAC reduces the computational load of SNNs, as the MAC operation in the matrix-vector operation is replaced with the more computationally efficient ACC. Furthermore, the binary nature of both input and output in SNNs enables the replication of the second characteristic of biological neurons: sparsity. Effectively, only non-zero operations contribute to the ACC, leading to a division of operations into *effective* ones, contributing to aggregation, and *ineffective* ones. This characteristic can be exploited by specialized hardware capable of executing only non-zero operations, resulting in higher energy efficiency.

Surrogate Gradients

Whereas the binary output of the LIF enables leveraging sparsity, it also makes gradient-based learning more challenging. The spiking output hinders the flow of gradients during training. Unlike continuous activation functions in traditional NNs, the spiking behavior of Equation (33) is non-differentiable at the spike times, disrupting the smooth gradient flow.

The non-differentiable nature of Equation (33) arises from the discrete nature of neural spikes. Surrogate gradient functions approximate the gradient of the spiking function $\partial g(h)/\partial u$ with continuous, differentiable functions $h(u) \approx \partial g(h)/\partial u$, as shown in Figure 10 allowing for the calculation of gradients. This facilitates the application of gradient-based optimization algorithms like back-propagation through time (BPTT) an algorithm introduced in Appendix G, which provides details of training SNNs.

G. Training

Training NNs refers to the overall process of fine-tuning the learnable parameters of a network to optimize its performance on a specific task. There are three main training paradigms for NN, comprising supervised learning, where models are trained on labeled data, unsupervised learning, which explores patterns in unlabeled data, and semi-supervised learning, a hybrid approach combining labeled and unlabeled data to enhance model performance. This paper presents a supervised task with labeled data (X, y), where the NN is tasked with learning the labels y from the input data X.

Supervised learning consists of three main stages: presenting the network with the input data X, eval-

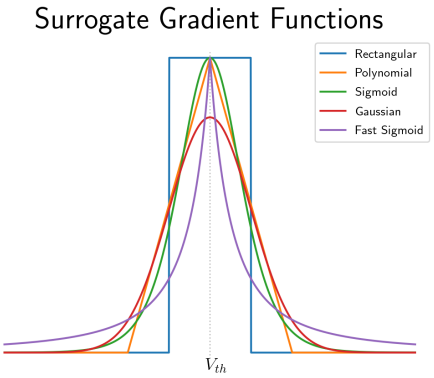


Figure 10: Visualization of surrogate gradient functions employed in spiking neural networks: Rectangular, Polynomial, Sigmoid, Gaussian, and Fast Sigmoid. These functions $h(x)$ play a crucial role in approximating the true gradient $\partial g/\partial u$, facilitating efficient training of spiking neural networks.

uating the error of prediction o of the NN compared against the labels y via a *Loss* function, and adjusting the trainable parameters. This process is repeated iteratively through a series of forward and backward passes. The primary goal of the training paradigm is to optimize the learnable parameters of the network to make accurate predictions or classifications on unseen data. The following three sections will present these stages.

Forward Pass

The forward pass involves the propagation of the input data X through all the layers of an NN, with each layer applying a set of learned weights and biases to transform the input and produce a final output o. This can be visualized in a *computational graph*, as shown in Figure 11. These forward passes have been defined in Equation (19) for the FC, Equation (29) for the LSTM and Equation (29) for the SNN, and Equation (22) presents how information is processed across multiple layers. Typically, the training data X is grouped into smaller sets of *batches* which balance the computational efficiency of batch processing and the stochastic nature of updating model parameters using individual data samples.

Loss Function

The Loss function measures the error between the predicted output o of the NN and the actual targeted labels y in the training dataset. The Loss serves as a measure of the model's error, and the goal of the training process is to minimize it, effectively improving its performance.

This work used the mean squared error (MSE)

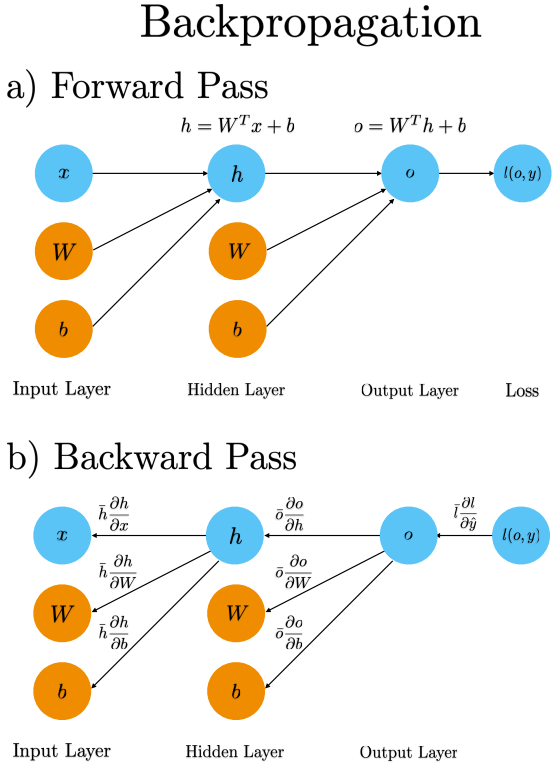


Figure 11: Exemplary computational graph visualizing the forward and backward pass. a) The forward pass propagates training data through the NN from input x to final prediction o by computing the output of each layer and propagating it to the following layer. b) The backward pass computes the gradient of every learnable parameter in the network with respect to the loss $l(y, o)$. The backpropagation algorithm updates the gradients iteratively via the chain rule and minimizes the computational load.

Loss l_{ann} for the ANN, defined as

$$l_{ann}(y, o) = \frac{1}{bn} \sum_{i}^{b} \sum_{k}^{n} (y_{ik} - o_{ik})^2 \tag{34}$$

with the labels $y \in \mathbb{R}^{b \times n}$ and the predictions $o \in \mathbb{R}^{b \times n}$, with a batch size b and n output features.

The LSTM and the SNN returned the output $o \in \mathbb{R}^{b \times t \times n}$ along the time dimension of size t. To counter vanishing gradients, the same dimensions of the labels were extracted and the MSE l_{snn} was scaled linearly from 0 to 1 along the time domain.

$$l_{lstm}(y, o) = \frac{1}{btn} \sum_{i}^{b} \sum_{j}^{t} \sum_{k}^{n} (y_{ijk} - o_{ijk})^2 \tag{35}$$

$$l_{snn}(y, o) = \frac{1}{btn} \sum_{i}^{b} \sum_{j}^{t} \sum_{k}^{n} \frac{t}{j} (y_{ijk} - o_{ijk})^2 \tag{36}$$

Backward Pass

The backward pass involves computing the gradient of the loss function with respect to every learnable parameter of the network. These gradients indicate how much each parameter contributed to the error and are used to update the weights and biases. This process uses efficient algorithms for computing the gradients, such as backpropagation and backpropagation through time (BPTT), that exploit the chain rule and re-use computations to reduce the computational load. The gradients are calculated layer by layer, starting from the output and moving backward through the network, following the *computational graph* shown in Figure 11.

Backpropagation

The backpropagation algorithm is shown in Figure 11. The depicted network consists of two conventional FC layers which take labeled (X, y) data as inputs and produce a prediction o. The gradient of the error, as defined by the loss function $l(o, y)$, is propagated to the previous node in the computational graph o. The gradients can be used to update the learnable parameters in a network respective to their contribution to the error.

The gradient of the output $\boldsymbol{o}$ is computed as

$$\bar{o} = \bar{l}\frac{\partial l}{\partial o} \tag{37}$$

where $\bar{a} := \frac{\partial l}{\partial a}$ and $\bar{l} := 1$. Repeating this for the learnable parameters W and b of the hidden layer, shown in orange, yields

$$\bar{W} = \bar{o}\frac{\partial o}{\partial W} \tag{38}$$

$$\bar{b} = \bar{o}\frac{\partial o}{\partial b} \tag{39}$$

$$\tag{40}$$

This process is continued until the gradients of all learnable parameters are computed. The efficacy of backpropagation stems from exploiting previously calculated values, such as the gradient of o, $\bar{o}$, which are repeatably required in computations. This process is automated using the PyTorch automatic backpropagation engine.

Backpropagation Through Time

Since SNNs or LSTMs process data over time steps and retain information from previous data points, gradients need to be computed along the time domain. BPTT, as developed independently by [29–31], *unravels* the computational graph of RNNs along the time domain, as shown in Figure 12. This idea has also been applied to SNNs named spatio-temporal backpropagation (STBP) [28]. *Unraveling* the computational graph is computationally intensive, which tends to slow the training of LSTMs and SNNs.

Gradient Descent

Gradient descent is an optimization algorithm used to minimize the loss function l during the training of NNs. The process involves iteratively adjusting every NN learnable parameters x in the opposite direction of the gradient $-\bar{x} := -\frac{\partial l}{\partial x}$. Ideally, the learnable parameters are updated using the average of the gradients computed from the entire training dataset. However, this is computationally intensive and can become infeasible, especially for larger datasets. This is called batch gradient descent. Stochastic gradient descent, in contrast, updates the gradients for every data sample. While this is computationally more efficient, it introduces variability in gradients and converges slower. The most commonly used method is "mini-batch" gradient descent, where the NN processes batched data and gradient updates are computed per "mini-batch". This combines the benefits of batch and stochastic gradient descent and is employed in this study with a batch size of 128.

H. Evaluation

Developing a neural network usually comprises two parts: training and testing. This section describes key concepts for training NN and assessing their performance, encompassing critical steps such as splitting the data into mutually exclu-

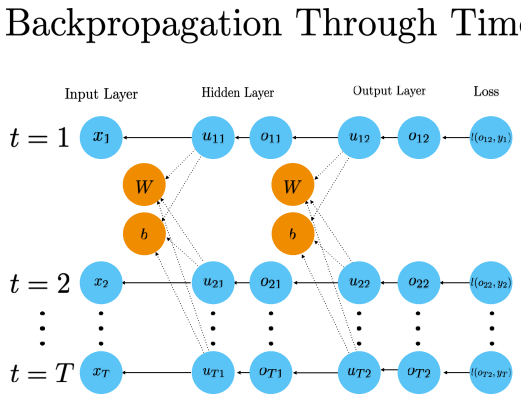


Figure 12: Exemplary computational graph visualizing backpropagation through time (BPTT) for a spiking neural network (SNN). The computational graph is "unraveled" along the time domain, as shown on the vertical axis and across the layers on the horizontal axis. The learnable parameters, shown in orange, require gradient information from all nodes indicated by dotted arrows. This can become computationally intensive for deeper neural networks or longer time windows.

sive sets for training, testing and validation, cross-validation, and visually analyzing learning curves.

Splitting the Data

To ensure the development of robust NN and reliable performance evaluation, it is common practice to split the entire data set X into mutually exclusive training, testing, and validation sets. The training data is typically randomized and processed for multiple iterations or epochs by the NN to learn the underlying data distributions of the data. One epoch corresponds to one training cycle given the entire training data, which is followed by adjusting the learnable parameters of the NN through iterative optimization processes. The core goal of training NN is to develop a model that can generalize well to unseen data. This is where the testing set becomes critical; by evaluating the NN on data it has never encountered during training, researchers can estimate how well the NN performs in real-world scenarios, helping identify potential issues such as overfitting. The validation data plays a crucial role in fine-tuning the NN's hyperparameters. The hyperparameters, such as the learning rate, the momentum p or the Dropout ratio p, are distinct from the learnable parameters adjusted during training and significantly impact the NN's architecture and behavior. Assessing the NN's performance on the validation set allows to adjust hyperparameters to optimize the NN for better generalization.

This separation of data into distinct sets ensures a fair evaluation and guards against the risk of data leakage, where the NN might inadvertently incorporate information from the testing or validation sets during training, leading to an overestimated performance. In essence, carefully allocating the data into training, testing, and validation sets is fundamental when evaluating NNs.

In this study, the neural data was $X \in \mathbb{R}^{b \times t \times f}$, with batch size b, a time window t, and f output features. The ANNs used a time window $t = 1$, and the RNNs processed windows containing 50 time steps. In the inference, that is, testing and validation, the batch size was set to $b = 1$ to account for data arriving continuously.

Cross-Validation

Cross-validation is a crucial technique in machine learning for assessing the performance and generalization ability of a model, particularly when faced with limited data. The standard approach involves partitioning the dataset into two sets, one for training and validation and the second set for testing. The first set is dividing the dataset into multiple subsets or "folds." This ensures a robust evaluation across different data configurations. The model is trained on a fraction of the "folds" and then evaluated on the remaining unseen "folds". This process is repeated several times, each fold taking a turn as the validation set. The results are then averaged or otherwise aggregated to provide a more robust estimate of the model's performance. This is shown in Figure 13.

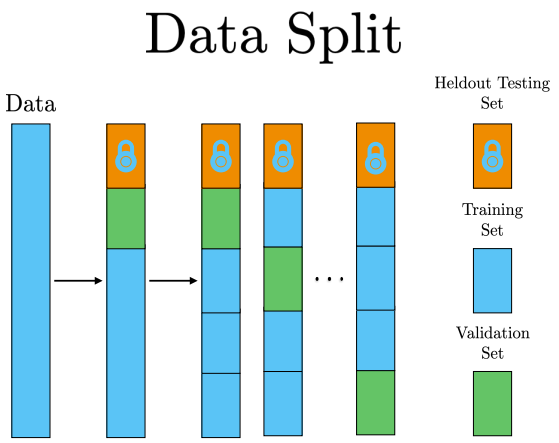


Figure 13: The data is split first into two parts, one for training and validation, and the second one is held-out for testing. The first part is partitioned into four subsets or "folds", and the model is trained and validated four times, each using a different fold as the test set and the remaining folds for training. This process ensures a comprehensive and robust evaluation of the performance of the model across different data subsets.

One common form of cross-validation is k-fold cross-validation, where the dataset is partitioned into k equal-sized folds. The model is trained on $k - 1$ folds and validated on the remaining one. This process is repeated k times, ensuring that each fold serves as a test set exactly once. Cross-validation helps mitigate the impact of dataset variability and provides a more reliable assessment of a model's performance by leveraging multiple train-test splits. It is especially beneficial in scenarios where the available data is limited, as it allows researchers to make the most effective use of the available samples for training and evaluation, producing a more robust and trustworthy assessment of a model's capabilities.

This study splits the data into 10-fold data based on. One reach spans from the onset of the visual stimuli until the moment the NHP reaches the target, as shown in Figure 3.

Learning Curves

Learning curves are graphical representations that visualize the performance of a model that evolves over time, often in relation to the amount

of training data it is exposed to. Typically, a learning curve depicts the training and validation performance metric, such as accuracy or loss, over successive epochs of the training process, as shown in the bottom row of Figure 14. These curves provide valuable insights into the model's behavior and training dynamics.

In the early stages of training, learning curves might display erratic behavior, reflecting the model's struggle to capture patterns in the data. As the model encounters more examples, the curve tends to stabilize, and performance metrics converge to a steady state. Learning curves are instrumental in diagnosing potential issues during training, such as underfitting or overfitting. A downward-sloping training curve with a concurrently increased validation curve may indicate overfitting, while a stagnant or decreasing validation curve suggests underfitting, as shown in Figure 14. By observing these patterns, researchers can make informed decisions about adjusting the complexity of a model, gathering additional data, or fine-tuning hyperparameters to enhance the overall performance of the model.

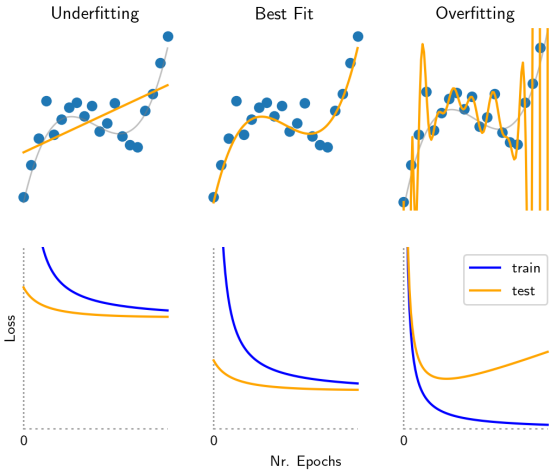


Figure 14: Visualization of Learning Curves. The top row shows the predictions of three different models, and the bottom row visualizes their respective learning curves. The first model, shown in the top left, is too simple to extract meaningful information from the training data, shown in blue, and ergo underfits. This is indicated by a high training and testing error in the bottom left. The second model is well able to capture the underlying data variance. The third model is overly complex, resulting in a high training error and a poor generalization ability to the unseen test data.

I. Optimization

This section introduces key methodologies, problems, and techniques employed in this paper to refine and enhance the performance of neural networks. Visually investigating learning curves allows for detecting overfitting and underfitting, which can be mitigated using regularization techniques such as Dropout and Early stopping. Similarly, Batch normalization (BN) and layer normalization (LN) are commonly used to improve learning. These techniques play a crucial role in shaping the capacity of the model to generalize well to new data, striking a balance between complexity and simplicity, and improving the robustness and accuracy of the network.

Overfitting and Underfitting

Overfitting and underfitting are two common challenges related to the NN's ability to generalize from training data to new, unseen data.

Overfitting occurs when a neural network learns the training data too well, capturing not only the underlying data variance but also noise and random fluctuations present in the data. As a result, the model's training performance is high, but it fails to generalize effectively to unseen testing data. Overfitting can be identified by observing a significantly increased error when evaluating the model on the validation or testing data compared to the training set, as shown in Figure 14.

On the other hand, underfitting occurs when a neural network is too simple to capture the underlying data variance of the training data. This results in poor performance on the training and the testing data, shown in Figure 14, as the model lacks the complexity needed to represent the true relationships within the data.

Underfitting can be mitigated by increasing the complexity of the NN by increasing its width or the number of layers. Overfitting tends to be a much more complex problem because the primary solution, to use more training data, is often not feasible. Therefore, regularization techniques are employed to reduce the complexity of the NN, effectively improving its ability to generalize to unseen data.

Normalization

One problem of training multi-layered neural networks is that the input of every layer changes continuously, owing to the updated weights in the previous layer. This requires careful optimization of the learnable parameters in a network which increases the complexity of the training and makes it slower. This problem, termed *internal covariate shift* [32], can be countered by normalizing each neuron with zero mean and unit variance. Ideally, normalization techniques normalize each neuron by the true mean and the true variance of the data distribution. This is not possible in practice, given the limited training data, batch-processing

or real-time inference. Two widely used normalization techniques that approximate the mean and std of neurons are BN and LN.

Batch Normalization

BN [32] computes the normalization statistics, the mean μ and the variance σ^2 per neuron j across the training batch of size b as

$$\mu_j = \frac{1}{b}\sum_i^b x_i \tag{41}$$

$$\sigma_j^2 = \frac{1}{b}\sum_i^b (x_i - \mu)^2 \tag{42}$$

$$BN(\boldsymbol{x}) = \frac{\boldsymbol{x} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \tag{43}$$

Calculating these normalization statistics require processing in batches, which makes them infeasible for inference when data points are evaluated individually, that is, a batch size of 1. Therefore, running normalization statistics, in the form of a running mean $\hat{\boldsymbol{\mu}}$ and running standard deviation $\hat{\boldsymbol{\sigma}}$, are computed during training and inserted into Equation (43). These running statistics are updated per neuron j with a momentum ρ after every batch as

$$\hat{\mu}_0 = 0 \tag{44}$$

$$\hat{\sigma}_0 = 1 \tag{45}$$

$$\hat{\mu}_j = (1-\rho)\hat{\mu}_{j-1}^2 + \rho\mu \tag{46}$$

$$\hat{\sigma}_j^2 = (1-\rho)\hat{\sigma}_{j-1}^2 + \rho\sigma^2 \tag{47}$$

Layer Normalization

Computing mini-batches is impractical for recurrent neural networks which receive a continuous stream of input data. LN [33] computes the normalization statistics, the mean μ and the variance σ^2, per time step across a layer of size l as

$$\mu = \frac{1}{l}\sum_i^l x_i \tag{48}$$

$$\sigma^2 = \frac{1}{l}\sum_i^l (x_i - \mu)^2 \tag{49}$$

$$LN(\boldsymbol{x}) = \frac{\boldsymbol{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{50}$$

Since the computation of the mean μ and the variance σ^2 can be computed per time step, LN does not require running statistics for inference. BN and LN have learnable parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, which scale and shift the normalizations in Equation (43) and Equation (50) to form

$$BN(\boldsymbol{x}) = \frac{\boldsymbol{x} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \odot \gamma + \beta \tag{51}$$

$$LN(\boldsymbol{x}) = \frac{\boldsymbol{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} \odot \gamma + \beta \tag{52}$$

Dropout

Dropout [34] is a widely used and simple method to prevent overfitting. It involves multiplying the activation of a layer in an NN by random binary variables $\boldsymbol{a}$ sampled from a Bernoulli distribution with probability p, as shown in Equation (54). Dropout $\boldsymbol{a}$ is resampled per training sample, meaning that the operation disables a fraction of p percent of neurons in the layer. This forces the network to learn ensembles, *i.e.*, to train and combine different sub-sets of the neurons in the layer that robustly learn to extract the variance of the data.

$$\boldsymbol{a} \sim Bernoulli(p) \tag{53}$$

$$Dropout(\boldsymbol{x}) = \boldsymbol{a}\boldsymbol{x} \tag{54}$$

Equation (54) is only applied during the training of the network. During inference, this regularization technique is disabled, which requires scaling the input x by the Dropout probability p to account for the changed number of neurons. This is equivalent to changing Equation (54) to

$$Dropout(\boldsymbol{x}) = \frac{\boldsymbol{a}}{p}\boldsymbol{x} \tag{55}$$

Dropout and Batch Normalization

The combination of the two most common regularization techniques, Dropout followed by BN, often leads to worse performance [35]. This originates in the scaling by the inverse Dropout probability $1/p$ in Equation (55) which scales the variance and consequently the running variance in Equation (47) by $1/p^2$. Since Dropout changes functionality during training and inference, the running variance $\hat{\sigma}_i^2$ is inconsistent during these phases. This results in trained networks with inconsistent feature variances during training and inference, which have unstable performance [35].

Early Stopping

Early stopping is a regularization technique used in training neural networks to prevent overfitting. The process involves monitoring loss of the model on the validation data during the training. If the performance stops improving or degrades for a specified number of training epochs, the training

process is stopped early, preventing the model from continuing to learn and potentially overfitting on the training data. The threshold for consecutive training epochs without improvement in the validation loss is called *patience*.

References Appendix

[1] Stavros Zanos. "Closed-Loop Neuromodulation in Physiological and Translational Research". en. In: *Cold Spring Harbor Perspectives in Medicine* 9.11 (Nov. 2019), a034314. ISSN: 2157-1422. DOI: 10.1101/cshperspect.a034314. URL: http://perspectivesinmedicine.cshlp.org/lookup/doi/10.1101/cshperspect.a034314 (visited on 10/24/2023).

[2] Gerd Tinkhauser et al. "The modulatory effect of adaptive deep brain stimulation on beta bursts in Parkinson's disease". In: *Brain* 140.4 (2017), pp. 1053–1067.

[3] Simon Little and Peter Brown. "The functional role of beta oscillations in Parkinson's disease". In: *Parkinsonism & related disorders* 20 (2014), S44–S48.

[4] Patrick Kwan et al. *Definition of drug resistant epilepsy: consensus proposal by the ad hoc Task Force of the ILAE Commission on Therapeutic Strategies*. 2010.

[5] Jing-Jing Fan et al. "Research progress of vagus nerve stimulation in the treatment of epilepsy". In: *CNS neuroscience & therapeutics* 25.11 (2019), pp. 1222–1228.

[6] Rhaya L Johnson and Christopher G Wilson. "A review of vagus nerve stimulation as a therapeutic intervention". In: *Journal of inflammation research* (2018), pp. 203–213.

[7] Robert S Fisher et al. "Automatic vagus nerve stimulation triggered by ictal tachycardia: clinical outcomes and device performance—the US E-37 trial". In: *Neuromodulation: Technology at the Neural Interface* 19.2 (2016), pp. 188–195.

[8] Preci Hamilton et al. "Clinical outcomes of VNS therapy with AspireSR®(including cardiac-based seizure detection) at a large complex epilepsy and surgery centre". In: *Seizure* 58 (2018), pp. 120–126.

[9] Mark Horowitz. "1.1 Computing's energy problem (and what we can do about it)". In: *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2014, pp. 10–14. ISBN: 978-1-4799-0920-9 978-1-4799-0918-6. DOI: 10.1109/ISSCC.2014.6757323. URL: http://ieeexplore.ieee.org/document/6757323/ (visited on 07/20/2023).

[10] Joshua I. Glaser et al. "Machine Learning for Neural Decoding". en. In: *eneuro* 7.4 (July 2020), ENEURO.0506–19.2020. ISSN: 2373-2822. DOI: 10.1523/ENEURO.0506-19.2020. URL: https://www.eneuro.org/lookup/doi/10.1523/ENEURO.0506-19.2020 (visited on 08/28/2023).

[11] Joseph G Makin et al. "Superior arm-movement decoding from cortex with a new, unsupervised-learning algorithm". In: *Journal of Neural Engineering* 15.2 (Apr. 2018), p. 026010. ISSN: 1741-2560, 1741-2552. DOI: 10.1088/1741-2552/aa9e95. URL: https://iopscience.iop.org/article/10.1088/1741-2552/aa9e95 (visited on 08/28/2023).

[12] Jiawei Liao et al. "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)*. Incheon, Korea, Republic of: IEEE, June 2022, pp. 134–137. ISBN: 978-1-66540-996-4. DOI: 10.1109/AICAS54282.2022.9869846. URL: https://ieeexplore.ieee.org/document/9869846/ (visited on 07/20/2023).

[13] Zoubin Ghahramani, Daniel M Wolpert, and Michael I Jordan. "Generalization to local remappings of the visuomotor coordinate transformation". In: *Journal of Neuroscience* 16.21 (1996), pp. 7085–7096.

[14] E.A. Wan and R. Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. Lake Louise, Alta., Canada: IEEE, 2000, pp. 153–158. ISBN: 978-0-7803-5800-3. DOI: 10.1109/ASSPCC.2000.882463. URL: http://ieeexplore.ieee.org/document/882463/ (visited on 09/17/2023).

[15] Simin Li, Jie Li, and Zheng Li. "An Improved Unscented Kalman Filter Based Decoder for Cortical Brain-Machine Interfaces". In: *Frontiers in Neuroscience* 10 (Dec. 2016). ISSN: 1662-453X. DOI: 10.3389/fnins.2016.00587. URL: http://journal.frontiersin.org/article/10.3389/fnins.2016.00587/full (visited on 10/15/2023).

[16] V.Y. Pan, F. Soleymani, and L. Zhao. "An efficient computation of generalized inverse of a matrix". en. In: *Applied Mathematics and Computation* 316 (Jan. 2018), pp. 89–101. ISSN: 00963003. DOI: 10.1016/j.amc.2017.08.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S0096300317305611 (visited on 09/27/2023).

[17] Jerry Tang et al. "Semantic reconstruction of continuous language from non-invasive brain recordings". en. In: *Nature Neuroscience* 26.5 (May 2023), pp. 858–866. ISSN: 1097-6256, 1546-1726. DOI: 10.1038/s41593-023-01304-9. URL: https://www.nature.com/articles/s41593-023-01304-9 (visited on 08/29/2023).

[18] Chethan Pandarinath et al. "Inferring single-trial neural population dynamics using sequential auto-encoders". en. In: *Nature Methods* 15.10 (Oct. 2018), pp. 805–815. ISSN: 1548-7091, 1548-7105. DOI: 10.1038/s41592-018-0109-9. URL: https://www.nature.com/articles/s41592-018-0109-9 (visited on 08/28/2023).

[19] Matthew S. Willsey et al. "Real-time brain-machine interface in non-human primates achieves high-velocity prosthetic finger movements using a shallow feedforward neural network decoder". en. In: *Nature Communications* 13.1 (Nov. 2022), p. 6899. ISSN: 2041-1723. DOI: 10.1038/s41467-022-34452-w. URL: https://www.nature.com/articles/s41467-022-34452-w (visited on 06/26/2023).

[20] Warren Mcculloch and Walter Pitts. "A Logical Calculus of Ideas Immanent in Nervous Activity". In: *Bulletin of Mathematical Biophysics* 5 (1943), pp. 127–147.

[21] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[22] Jason K Eshraghian. "Human ownership of artificial creativity". In: *Nature Machine Intelligence* 2.3 (2020), pp. 157–160.

[23] Yu Zhang, William Chan, and Navdeep Jaitly. "Very deep convolutional networks for end-to-end speech recognition". In: *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2017, pp. 4845–4849.

[24] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[25] Oriol Vinyals et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (2019), pp. 350–354.

[26] Douwe Kiela et al. "Plotting Progress in AI". In: *Contextual AI Blog* (2023). https://contextual.ai/blog/plotting-progress.

[27] Jason K Eshraghian et al. "Training spiking neural networks using lessons from deep learning". In: *Proceedings of the IEEE* (2023).

[28] Yujie Wu et al. "Spatio-temporal backpropagation for training high-performance spiking neural networks". In: *Frontiers in neuroscience* 12 (2018), p. 331.

[29] MC Mozer. "A Focused Backpropagation Algorithm for Temporal Pattern Recognition. Backpropagation: Theory, architectures, and applications. ResearchGate". In: *ResearchGate. Hillsdale, NJ: Lawrence Erlbaum Associates* (1995), pp. 137–169.

[30] Paul J Werbos. "Generalization of backpropagation with application to a recurrent gas market model". In: *Neural networks* 1.4 (1988), pp. 339–356.

[31] Anthony J Robinson and Frank Fallside. *The utility driven dynamic error propagation network*. Vol. 1. University of Cambridge Department of Engineering Cambridge, 1987.

[32] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[33] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[34] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[35] Xiang Li et al. "Understanding the Disharmony Between Dropout and Batch Normalization by Variance Shift". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 2677–2685. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00279. URL: https://ieeexplore.ieee.org/document/8953671/ (visited on 09/19/2023).