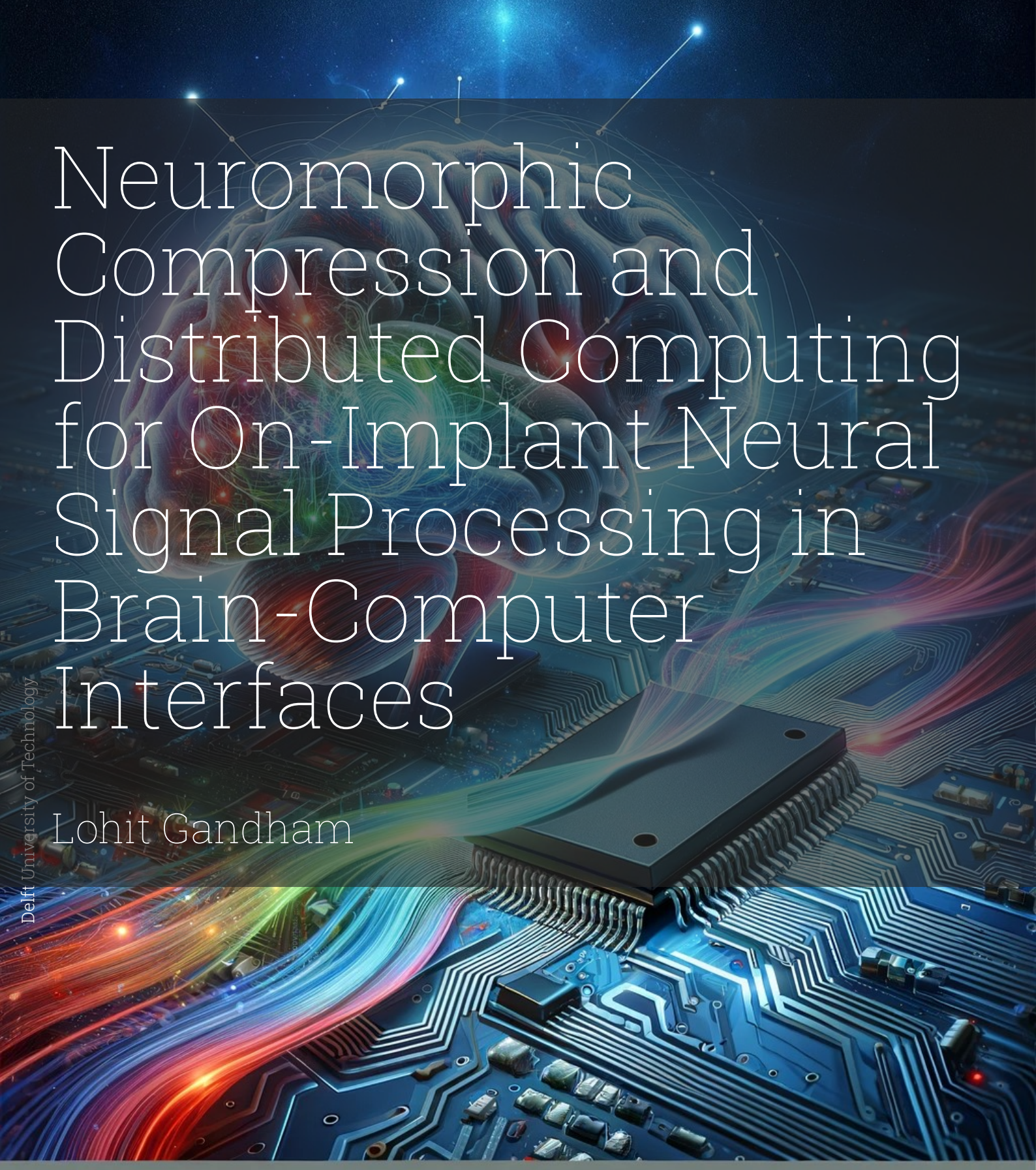


# Neuromorphic Compression and Distributed Computing for On-Implant Neural Signal Processing in Brain-Computer Interfaces

Lohit Gandham

Delft University of Technology

TU Delft

# Neuromorphic Compression and Distributed Computing for On-Implant Neural Signal Processing in Brain-Computer Interfaces

by

Lohit Gandham

to obtain the Master of Science in Computer Engineering and Embedded Systems Engineering degree at the Delft University of Technology, which will be defended behind closed doors on Wednesday, August 29th, 2024, at 12:30 PM.

| | |
|---|---|
| Student Number: | 5733766 |
| Project Duration: | 1st September, 2023 - 29th August, 2024 |
| Faculty: | Quantum and Computer Engineering, Delft |
| Thesis Committee: | Dr. Rajendra Bishnoi, TU Delft, Supervisor |
| | Dr. Said Hamudoui, TU Delft, Thesis Chair |
| | Dr. Nergis Tömen, TU Delft, External |
| | Dr. Hua- Peng Liaw, Imec, Supervisor |
| | Dr Yao-Hong Liu, Imec, Supervisor |

An electronic version of this thesis is available at http://repository.tudelft.nl/

TU Delft · imec

*”The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point”.*
- Claude Shannon

# Acknowledgements

I would like to express my deepest gratitude to my parents and brother for their unwavering support. They always believed in me, motivated me, and encouraged me throughout this unforgettable journey of learning. Their constant backing made all the difference in helping me reach this milestone.

I would also like to extend my sincere thanks to my supervisor, Dr. Rajendra Bishnoi, for his invaluable guidance, support, and encouragement throughout this project. His insights and expertise were instrumental in shaping this thesis into what it is today.

My heartfelt appreciation goes to my colleagues at Imec for their technical support and for creating a stimulating research environment. I am especially grateful to Dr. Yao-Hong Liu, Dr. Hua-Peng Liaw, Dr. Guangzhi Tang, Mr. Yuming He, and Dr. Manolis Sifalakis, from whom I learned not only about cutting-edge technologies but also about the importance of an unwavering work ethic. Their constant support throughout my thesis journey helped me grow as an individual and made me more rounded as a researcher.

Finally, I would like to thank my thesis committee members, Dr. Said Hamudoui and Dr. Nergis Tömen, for taking the time to review, attend, and evaluate my work.

I apologize if I have missed anyone, but please know that you all have my deepest gratitude for contributing to the success of this thesis.

*Lohit Gandham*
*Delft, August 2024*

# Abstract

Neuromorphic systems offer a promising solution to the computational challenges of intra-cortical Brain-Computer Interfaces (iBCIs), leveraging the event-driven nature of biological neural networks for enhanced power efficiency and data scalability. The exponential growth in neural data rates due to advancements in high-density neural recording creates significant hurdles in data transmission and processing. This paper introduces a novel neuromorphic compression architecture that combines delta-modulated level crossing events with Spiking Neural Networks (SNNs) trained through a two-stage process with grouped inputs. The proposed system achieves a compression ratio exceeding 94× and a peak Signal-to-Noise and Distortion Ratio (SNDR ) of 22 dB compared to the signal without Local Field Potentials. This substantial improvement over existing methods drastically reduces data throughput making it ideal for implantable devices where resource constraints are critical. Validated on hippocampal and cerebellar datasets, our approach significantly outperforms state-of-the-art techniques in terms of computational and compression efficiency, SNDR, and memory usage—requiring just 384 bits per channel to store necessary data for processing and 18.07K Multiply and Accumulate operations (MACs) per spike. These findings underscore the potential of neuromorphic SNNs as a transformative technology for future iBCIs, advancing neural prosthetics and brain-machine interfaces.

*Keywords: Neuromorphic, Spiking Neural Networks, Brain-Computer Interface, Neural Signal Compression, Delta Modulation, Implantable Devices, Signal-to-Noise and Distortion Ratio, Compression, Autoencoders.*

# Contents

**Acknowledgements** ii

**Abstract** iii

**Nomenclature** ix

**1 Introduction** 1
- 1.1 Motivation and Current Challenge . . . 1
- 1.2 Problem Statement . . . 3
- 1.3 Contributions . . . 3
- 1.4 Organization . . . 4

**2 Background** 5
- 2.1 Neuromorphic Computing and Fundamentals of Neural Recording . . . 5
  - 2.1.1 Neural Recording Process . . . 6
- 2.2 Basics of Deep Neural Network Topologies . . . 7
  - 2.2.1 Fully Connected Network . . . 7
  - 2.2.2 Convolutional Neural Network (CNN) . . . 8
  - 2.2.3 Recurrent Neural Networks (RNNs) . . . 9
  - 2.2.4 Long-Short-Term-Memory Networks (LSTM) . . . 10
  - 2.2.5 Spiking Neural Networks (SNNs) . . . 11
- 2.3 Hardware Aware Optimizations . . . 13
  - 2.3.1 Batch Normalization Folding in Neural Networks . . . 13
  - 2.3.2 Quantization in Neural Networks and Symmetric Quantization . . . 13

**3 Related Work** 15
- 3.1 Transformations . . . 15
- 3.2 Delta Modulation for Neural Recording Systems . . . 15
- 3.3 Block-based compression . . . 16
- 3.4 Compressed Sensing . . . 16
- 3.5 Autoencoders . . . 17

**4 Proposed Neuromorphic Compression System** 19
- 4.1 Delta-modulated Analog-to-Digital Converter ($\Delta$-ADC) . . . 20
- 4.2 Neuromorphic Hybrid Autoencoder . . . 21
  - 4.2.1 Encoder – Decoder Network . . . 21
  - 4.2.2 Knowledge Distillation . . . 22
  - 4.2.3 Spiking Neural Network . . . 22
- 4.3 Training the Spiking Neural Network . . . 23
  - 4.3.1 2 Stage Student Encoder-Decoder Training . . . 23
  - 4.3.2 Hardware Aware Optimizations . . . 24

**5 Metrics and Performance Evaluation** 27
- 5.1 Compression Ratio (CR) . . . 27
- 5.2 Signal-to-Noise and Distortion Ratio (SNDR) . . . 28
- 5.3 Root Mean Squared Error (RMSE) . . . 28
- 5.4 Memory per Channel (bits/ch) . . . 29
- 5.5 Multiply and Accumulate Operations per Second (MAC) . . . 29
- 5.6 Memory Access . . . 30

**6 Results and Experiments** 31
- 6.1 Performance Summary and State-of-the-Art (SoTA) Comparison . . . 31

6.2 Influence of Teacher Network Architectures on Performance . . . 32
6.3 Impact of Teacher Network Depth . . . 34
6.4 Optimization of Latent Space Dimensionality . . . 35
6.5 Evaluation of Student Network Depth . . . 37
6.6 Pruned Student Model Configuration . . . 37
6.7 Quantization . . . 38

**7 Discussions and Conclusion** **41**
7.1 Discussions . . . 41
7.2 Conclusion . . . 42

**8 Future Scope** **43**

**References** **45**

**Appendix A:RTL Implementation of SNN** **49**
A.1 Hardware Implementation . . . 49
A.1.1 Neuron-Level RTL Implementation . . . 49
A.1.2 Layer-Level RTL Implementation . . . 51
A.1.3 Simulation Results . . . 51

**Appendix B:Timeline** **53**

**Appendix C:IOP Publication Final Draft** **55**

# List of Figures


1.1 Exponential growth of simultaneously recorded neurons over time [47] . . . . . 1
1.2 Power consumption estimates per channel (log scale) for systems recording systems . 2
1.3 Thermal constraints for brain implants to ensure tissue health . . . . . 3

2.1 Possibilities of neuromorphic computing based systems. [44] . . . . . 5
2.2 (A) Position-dependent extracellular spike waveforms during an action potential. Black dots indicate electrode positions. Spike traces are normalized and color-coded by the magnitude of the negative peak. (B) Corresponding somatic membrane potential during the action potential. Dashed lines show the alignment of the maximum somatic potential with the extracellular waveform (C) at the position marked with an asterisk in (A). [31] . 6
2.3 Neural recording in a mouse and its spike amplitudes. [56] . . . . . 7
2.4 Fully connected network . . . . . 7
2.5 Convolutional neural network . . . . . 9
2.6 Illustration of a Recurrent Neural Network (RNN) architecture, showing how the hidden state $h_t$ is propagated across time steps. . . . . 10
2.7 LSTM cell composed of forget gate $f_t$, input gate $i_t$, and output gate $o_t$ . . . . . 11
2.8 Comparison between (a) a traditional ANN neuron and (b) a spiking neuron in an SNN. ANNs operate in the spatial domain with continuous activations, while SNNs operate in the spatio-temporal domain with binary spikes[51]. . . . . 12

3.1 Schematic diagram of the encoder-decoder network architecture showing the nonlinear ($\Phi$ and $\Phi$') latent vector and reconstruction transformation. . . . . 18

4.1 High-Level architecture for our Neuromorphic Compression System. The system processes action potentials which are apmlified and filtered from a Multielectrode Array (MEA). . . . . 20
4.2 The top panel (orange) depicts the reconstructed waveform through cumulative summation while the bottom panel (blue) shows delta-modulated level crossing events over time. 21
4.3 Knowledge distillation process in teacher – student architecture. . . . . 22
4.4 Illustration of a binary input system processing signals through a neuron model. The membrane potential (MP) is compared against a threshold to generate binary outputs. . 23
4.5 (a) Stage 1: student encoder training using the teacher's latent space. (b) Stage 2: student decoder training using the reconstruction loss . . . . . 24
4.6 Comparison of (a) fully connected network and (b) pruned network with grouping (in L1) and interleaving (in L2). . . . . 25
4.7 Illustration of (a) weight matrix for input layer L1 and (b) weight matrix for input layerL2 . 25

6.1 Comparison of original (blue) and reconstructed waveforms (orange) for different spikes in the hippocampus dataset (hippo_dat1) ((a) & (b)) and cerebellum dataset (cere_dat2) ((c) & (d)) using our neuromorphic autoencoder compression system. . . . . 32
6.2 Comparative performance of teacher architectures . . . . . 33
6.3 Comparison of student network compression ratios and accuracy . . . . . 34
6.4 Comparison of teacher RNN network with different layer sizes. . . . . 35
6.5 Comparison of teacher RNN network with different latent dimensions . . . . . 36
6.6 Comparison of student spiking network with different layer sizes . . . . . 37
6.7 Compression ratio (CR) across different precision levels for the hippo_dat1 and cere_dat2 datasets. . . . . 39
6.8 RMSE across different precision levels for the hippo_dat1 and cere_dat2 datasets. . . . 39

1 Neuron level model black-box framework . . . . . 50

2 Internal implementation of neuron . . . . . 50
3 RTL logic design block diagram . . . . . 51
4 Neuron level RTL simulation . . . . . 52
5 Layer level RTL simulation . . . . . 52
6 Timeline for Inception to Results . . . . . 53

# List of Tables

6.1 Summarized performace and comparsion to STATE-OF-THE-ART (SOTA) neural compression systems. . . . . 31
6.2 Types of student network connection explored for hardware aware optimization . . . . . 38
6.3 Hardware aware student analysis - bits used vs compression ratio vs reconstruction fidelity on different datasets. . . . . 38

# Nomenclature

## Abbreviations

| Abbreviation | Definition |
|---|---|
| ADC | Analog-to-Digital Converter |
| BCI | Brain-Computer Interface |
| CNN | Convolutional Neural Network |
| CR | Compression Ratio |
| LC-ADC | Level-Crossing Analog-to-Digital Converter |
| LFP | Local Field Potentials |
| LSTM | Long Short-Term Memory |
| MAC | Multiply-And-Accumulate |
| MP | Membrane Potential |
| NCT | Neural Compressive Telemetry |
| RMSE | Root Mean Square Error |
| RNN | Recurrent Neural Network |
| SNN | Spiking Neural Network |
| SNDR | Signal-to-Noise and Distortion Ratio |

## Symbols

| Symbol | Definition |
|---|---|
| $x(t)$ | Continuous neural signal |
| $E_n(t)$ | n-th epoch of the signal |
| $\Delta$ | Change between consecutive signal values |
| $y$ | Reconstructed data |
| $\eta$ | Transformation basis |
| $\Phi$ | Sampling matrix |
| $W$ | Weight parameter of the encoder |
| $W'$ | Weight parameter of the decoder |
| $\varphi$ | Nonlinearity function |
| RMSE | Root Mean Square Error |
| SNDR | Signal-to-Noise and Distortion Ratio (dB) |

# 1

# Introduction

*This chapter introduces the thesis. First, the motivation of the thesis is discussed, followed by the problem statement. Then, the thesis contribution is presented and finally, the thesis organization is briefed.*

## 1.1. Motivation and Current Challenge

Exploration of the human brain and the way it processes and decodes information has long been a topic of profound interest. Over the last few decades, developments in the field of Intra-cortical Brain-Computer Interfaces (iBCIs). particularly in both invasive approaches like intra-cortical BCIs, which involve placing electrodes directly into the brain's cortex, and less invasive methods such as electro-corticography (ECoG), which place electrodes on the surface of the cortex, have enabled researchers to bridge the gap between our understanding of brain function and the technology that can record and interpret neural signals. These advancements have had a significant impact on medicine—such as treating disabilities, epilepsy, and Parkinson's disease—as well as on neuroscience research and engineering disciplines [20],[5].

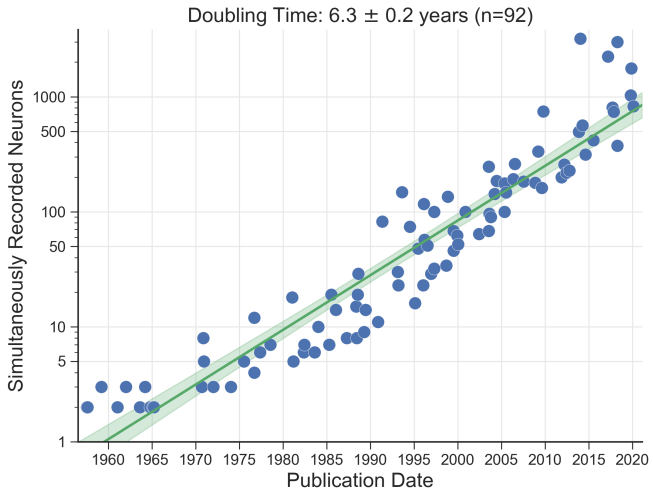



**Figure 1.1:** Exponential growth of simultaneously recorded neurons over time [47]

In today's BCI landscape, a significant challenge lies in processing many action potentials recorded from numerous electrodes. As demonstrated in a study by Stevenson and Kording [47], advancements

in neural recording techniques over the past fifty years have resulted in the number of simultaneously recorded neurons doubling roughly every seven years, paralleling Moore's law (as shown in Fig. 1.1).

This exponential increase has led to a huge influx of data, creating substantial stress in data transmission and recording systems. Additionally, studies in [13] have shown that the most power-consuming part in many conventional commercialized systems and academic prototypes available today is the transmitter, hence justifying the need for high bandwidth BCIs and the need for the compression of recorded neural signals (as shown in Fig. 1.2 wherein comparisons have been made for systems in [14],[6],[57],[24],[33],[12],[27],[1],[4]).

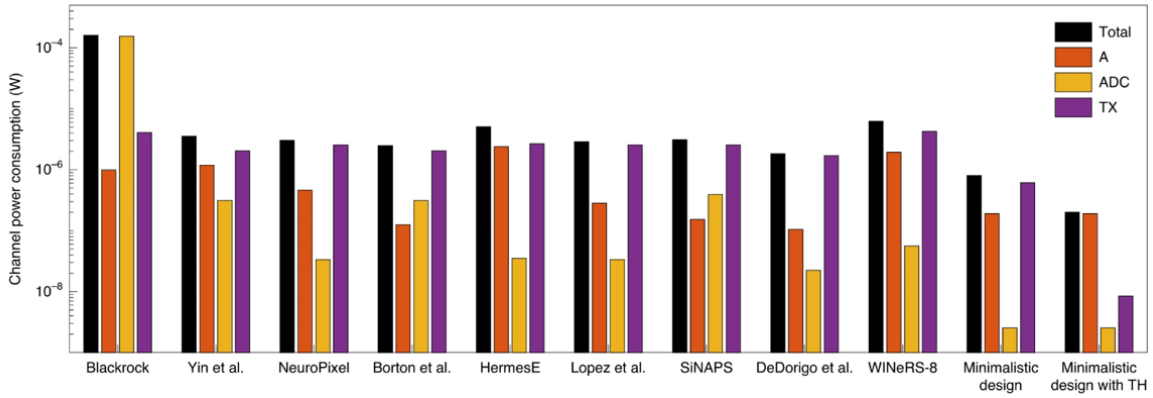



**Figure 1.2:** Power consumption estimates per channel (log scale) for systems recording systems

Furthermore, spike sorting—essential for distinguishing individual neuron activity—requires high-frequency sampling to capture the detailed shape of each spike [39]. This further amplifies the volume of data generated, exacerbating the strain on current systems. Additionally, N Even-Chen et al. [61] have shown that the most power-consuming part in many conventional commercialized systems and academic prototypes available today is the wireless transmitter, hence justifying the need for high bandwidth BCIs and the need for the compression of recorded neural signals.

For instance, the CMOS-based high-density neural probe described in [40] produces a data rate of 144 Mbps from 384 channels, each operating at a sampling rate of 30 kHz with a 12-bit ADC resolution. This significant data output presents considerable challenges for data storage and transmission, underscoring the need for sophisticated processing capabilities and high-bandwidth communication channels.

To address the challenges of processing large volumes of neural data within the constraints of implantable devices, researchers are increasingly adopting neuromorphic computing [23][37]. Neuromorphic systems emulate biological neural networks, enabling online data processing with minimal energy consumption. These systems incorporate event-based sensors that, like neurons, respond only to significant environmental changes, reducing unnecessary data processing [30]. Given the need to minimize implant volume and power consumption—essential for reducing invasiveness and preventing heat-related tissue damage—traditional methods like parallelization or multiplexing are impractical for implants. Maintaining a temperature increase of less than 1ºC for surrounding brain tissue is crucial for long-term health, with this requiring implant electronics to stay below a heat flux of 40 mW/cm² surface area Wolf et. al[54]. (as shown in Fig. 1.3)

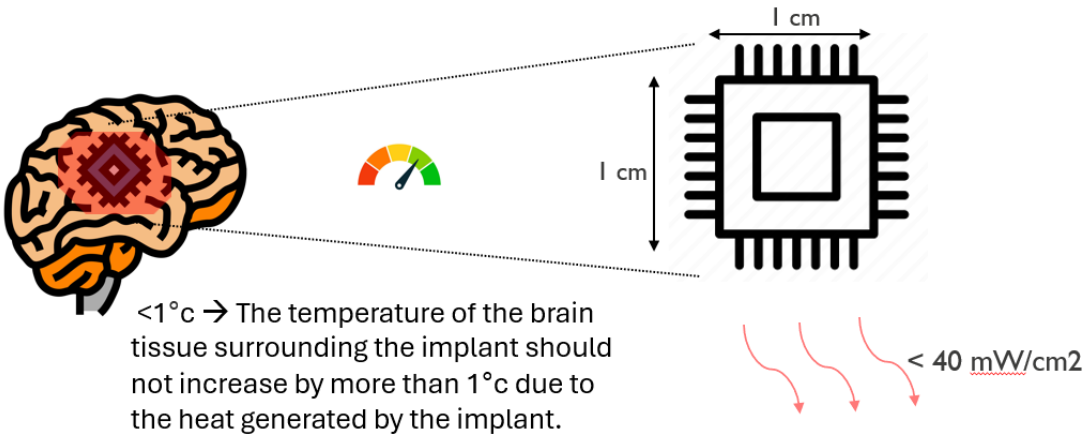



**Figure 1.3:** Thermal constraints for brain implants to ensure tissue health

## 1.2. Problem Statement

Hence the contemporary landscape of electrophysiology demands sophisticated methods to handle the voluminous data generated by high-channel-count devices, particularly in applications like brain-machine interfaces (BMIs). My thesis explored a novel neuromorphic two-step data compression system designed for extreme-edge environments, such as stand-alone neural implants. My innovation encapsulates a power-efficient approach to manage the enormous data throughput exceeding 100s Mbps, emphasizing the critical need for energy-efficient technologies in medical implants to prevent tissue damage and ensure long-term functionality.

To address the overarching challenge in this field mainly in compression, my research is guided by the following question:

*"Is there a way to develop an energy-efficient data compression system for high-channel-count electrophysiology recording devices that balances compression ratio, computational complexity, and power consumption while ensuring the integrity and safety of recorded neural data in constrained implant environments?"*

## 1.3. Contributions

My research introduces a novel spiking neural network (SNN) based compression approach to address the data challenges in iBCIs. Neural networks excel in this context due to their ability to model complicated, nonlinear relationships within large datasets, which is relevant for handling the intricacies of neural signals [26]. SNNs, a form of neuromorphic computing, enhance this approach by mimicking the brain's natural, event-driven processing [23], making them particularly energy-efficient and well-suited for on-device processing in BCIs, where minimizing power consumption and heat generation is critical to prevent tissue damage and ensure implant longevity [22], [43].

This approach achieves a compression ratio of 94× significantly reducing data rates while preserving the accuracy of recorded action potentials, significantly reducing data rates without sacrificing the accuracy of spike sorting. It also minimizes memory usage, cutting down the required memory per channel by almost half, and enhances computational efficiency by substantially reducing both the number of computations and memory accesses per spike[11].

This thesis tries to present a possible breakthrough in neural data compression, balancing the trade-offs between compression rate, computational complexity, and power consumption. It paves the way for advanced BCIs and other edge devices, proposing a scalable solution that could revolutionize our approach to handling large-scale neural data in constrained environments.

The development of this data compression system aims to represent a significant stride towards more sustainable and practical implementations of neural recording technologies, ensuring that they can operate within the stringent limits of power, space, and processing capabilities required by next-generation biomedical devices.

## 1.4. Organization

The remainder of this thesis is structured as follows:

- **Chapter 2 & 3** provides a comprehensive background and review of related work in neural signal compression techniques, highlighting the challenges faced in high-density neural recording and the solutions proposed in previous studies.
- **Chapter 4** details the proposed neuromorphic compression system, including the integration of delta modulation with Spiking Neural Networks (SNNs) and the key components of the system, such as the Amplifier, ADC, and Neuromorphic Autoencoder. This chapter also introduces the experimental setup and methodologies used to validate the proposed system, with a focus on datasets, preprocessing steps, and the training process of the autoencoder system.
- **Chapter 5** This section discusses the key metrics employed to evaluate the performance of the proposed model, focusing on reconstruction fidelity and providing a basis for comparison with other state-of-the-art methods. The evaluation metrics include Compression Ratio (CR), Signal-to-Noise and Distortion Ratio (SNDR), Root Mean Square Error (RMSE), and memory efficiency. These metrics are critical for assessing the system's ability to effectively compress and reconstruct neural signals while maintaining high accuracy and efficiency in comparison to existing approaches.
- **Chapter 6** presents the results of the experiments, including the ablation study for the justification of the teacher network, latent dimensions, and student network configurations, along with the analysis of pruning and quantization impacts on performance.
- **Chapter 7** offers a discussion on the implications of the findings, comparing the proposed system with existing state-of-the-art methods in terms of performance and efficiency.
- **Chapter 8** concludes the paper, summarizing the key contributions and suggesting potential avenues for future research in neuromorphic systems and brain-computer interfaces.

# 2

# Background

*This chapter provides the background for the thesis, exploring the field of neuromorphic computing and deep learning algorithms. It begins with an overview of basic extracellular recordings and then transitions into a discussion of the neural networks, algorithms, and hardware optimizations that are employed in this research.*

## 2.1. Neuromorphic Computing and Fundamentals of Neural Recording

Neuromorphic computing, first conceptualized by Carver Mead in the late 1980s, refers to the design of systems that mimic the architecture and functionality of biological neural networks [37]. The goal is to develop hardware that processes information like the human brain, leveraging the advantages of parallel processing, low power consumption, and fault tolerance. Neuromorphic systems use analog, digital, and mixed-signal circuits to simulate the properties of neurons and synapses.

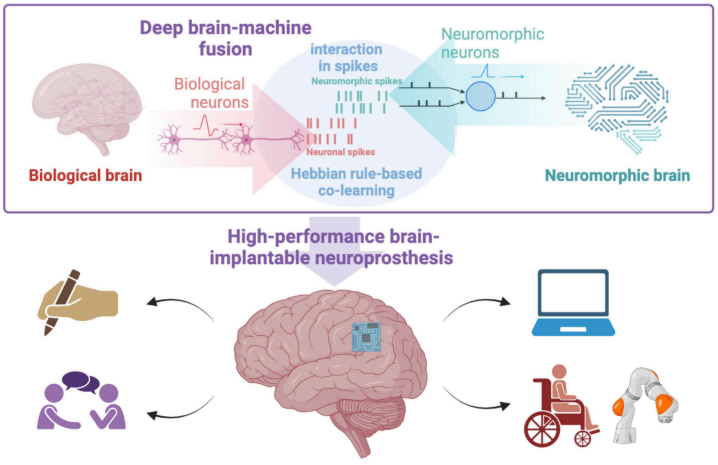



**Figure 2.1:** Possibilities of neuromorphic computing based systems. [44]

In recent years, neuromorphic computing has gained significant traction in the development of brain-machine interfaces (BMIs), where it is used to process and interpret signals from the brain for various applications, including neuroprosthetics and cognitive computing [44] (as shown in Fig. 2.1). In BMIs, neuromorphic architectures are employed to compress and process neural signals in real-time, enabling high-speed and efficient communication between the brain and external devices [9].

### 2.1.1. Neural Recording Process

Neural recording involves using electrodes, such as Neuropixels [40], to monitor the electrical activity of neurons, particularly during action potentials. When a neuron fires, depolarization occurs due to the influx of sodium ions ($Na^+$), followed by repolarization as potassium ions ($K^+$) are expelled. This process generates a negative extracellular voltage change, which is captured by extracellular recordings [31].

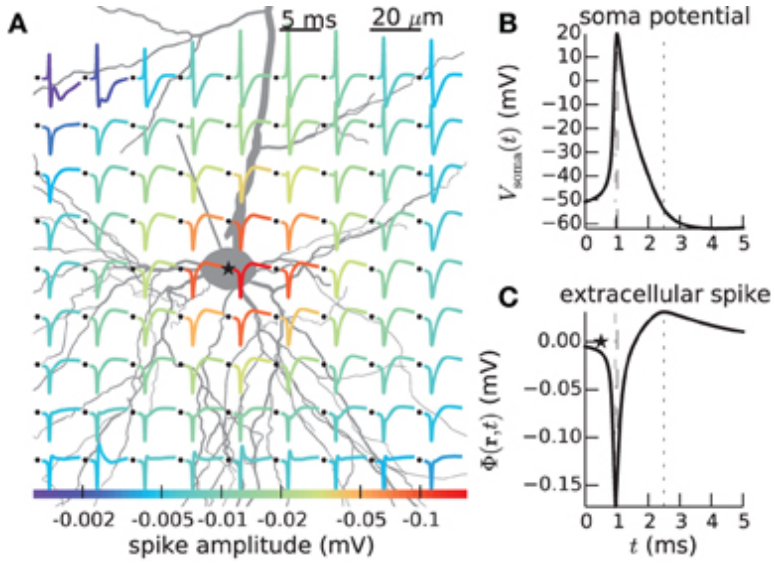



**Figure 2.2:** (A) Position-dependent extracellular spike waveforms during an action potential. Black dots indicate electrode positions. Spike traces are normalized and color-coded by the magnitude of the negative peak. (B) Corresponding somatic membrane potential during the action potential. Dashed lines show the alignment of the maximum somatic potential with the extracellular waveform (C) at the position marked with an asterisk in (A). [31]

To detect these extracellular signals, electrodes are placed near the neurons, often using a probe that contains multiple electrodes capable of simultaneously monitoring the activity of many neurons. This probe is based on CMOS silicon technology, enabling high-density and precise neural recordings [28]. The data used in this study was collected using a Neuropixels (NPX) probe, known for its high sensitivity and precision in capturing neural activity [40]. Although the data presented here comes from rat experiments, the principles of neural activity and spike behavior are consistent across different species, making the findings broadly applicable [24]. More about the type of data used will be explained in section 4.1

The data is sampled at a high frequency of 30 kHz, which is essential for accurately capturing the rapid voltage changes associated with action potentials. After amplification, typical neural spikes exhibit amplitudes between -65 and -70 mV, whereas, before amplification, these signals range from 50 to 300 microvolts (μV). Fig. 2.3 shows how the spike amplitudes change after amplification (Amp).

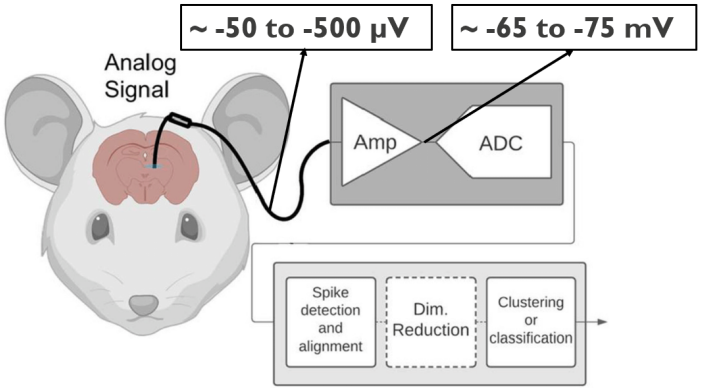



**Figure 2.3:** Neural recording in a mouse and its spike amplitudes. [56]

## 2.2. Basics of Deep Neural Network Topologies

In this section, I will briefly about the different types of neural networks that were used in the context of this thesis, namely Fully Connected Networks, Convolutional Neural Networks (CNNs), Long-term (LSTM), Recurrent Neural Networks (RNNs), Spiking Neural Networks.

### 2.2.1. Fully Connected Network

The simplest form of a feed-forward neural network is the Fully Connected Network. Fig. 2.4 illustrates a typical Fully Connected Network. This type of network is composed of an input layer, one or more hidden layers, and an output layer. As implied by its name, every neuron in a fully connected layer is linked to every activation from the preceding layer.

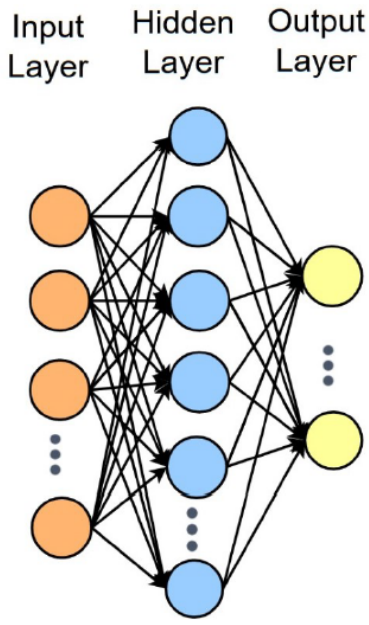



**Figure 2.4:** Fully connected network

**Input Layer** — The input layer receives the input vector $(x_1, \ldots, x_m)$. These values are standardized before being passed on to the next layer.

**Hidden Layer** — In the hidden layer, each input neuron's value is multiplied by a corresponding weight $w_{ij}$, and a bias $b_i$ is added to the result. The sum of these weighted inputs is then passed through an activation function $f$, producing the output of the hidden neuron $h_j$.

$$h_j = f\left(\sum_i (x_i \cdot w_{ij} + b_i)\right)$$

**Output Layer** — The output layer is designed according to the number of output classes. To represent $N$ output classes, the layer contains $N$ neurons. Each value from the hidden neurons in the previous layer is multiplied by a weight $w_{jk}$, with a bias $b_j$ added to the result. The weighted sum is then passed through an activation function $f$, producing the output value $y_k$. This activation function may vary across hidden layers. The resulting $y$ values form the network's output, with a softmax function applied to determine the classification output.

$$y_k = f\left(\sum_j (h_j \cdot w_{jk} + b_j)\right) \tag{2.1}$$

### 2.2.2. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a specialized type of neural network designed for processing data that has a grid-like topology, such as images. Unlike Fully Connected Networks, CNNs leverage the spatial structure of the data by applying convolutional filters, which are small, learnable matrices, across the input. This allows CNNs to capture local patterns such as edges, textures, and shapes, which are then combined to recognize more complex structures in deeper layers.

**Input Layer** — The input to a CNN is typically an image represented as a multi-dimensional array. Each dimension corresponds to a spatial feature, such as width (M), height (N), and color channels (C) for RGB images. The input is processed through convolutional filters that slide over the entire image to generate feature maps.

**Convolutional Layer** — In the convolutional layer, each filter is convolved across the input image, producing a feature map. The convolution operation for a filter $f$ applied to the input $x$ can be expressed as:

$$z_{i,j,k} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{c=0}^{C-1} f_{m,n,c,k} \cdot x_{i+m,j+n,c} \tag{2.2}$$

where $z_{i,j,k}$ is the resulting feature map, $M$ and $N$ are the filter dimensions, $C$ is the number of channels, and $k$ indexes the output feature map. An activation function (Act(.)), such as ReLU, is typically applied to introduce non-linearity into the model:

$$h_{i,j,k} = \mathsf{Act}(z_{i,j,k}) \tag{2.3}$$

**Pooling Layer** — The pooling layer is used to reduce the spatial dimensions of the feature maps, which helps in lowering the computational load and reducing overfitting. In max pooling, the operation can be expressed as:

$$p_{i,j,k} = \max_{(m,n)\in \mathsf{pool}} h_{i+m,j+n,k} \tag{2.4}$$

where $p_{i,j,k}$ is the pooled feature map and the max function operates over a pooling window.

**Fully Connected Layer** — After several convolutional and pooling layers, the resulting feature maps are flattened into a one-dimensional vector $\mathbf{v}$ and passed through one or more fully connected layers. The output of a fully connected layer can be expressed as:

$$o_i = f\left(\sum_j v_j \cdot w_{ji} + b_i\right) \tag{2.5}$$

where $w_{ji}$ are the weights, $b_i$ is the bias, and $f$ is the activation function.

**Output Layer** — Similar to Fully Connected Networks, the output layer in a CNN is configured according to the number of output classes. For classification tasks, a softmax function is often applied to the final fully connected layer to generate probabilities for each class:

$$\hat{y}_k = \frac{e^{o_k}}{\sum_j e^{o_j}} \tag{2.6}$$

where $\hat{y}_k$ represents the probability of the $k$-th class. The typical flow of a CNN is shown in Fig. 2.5.

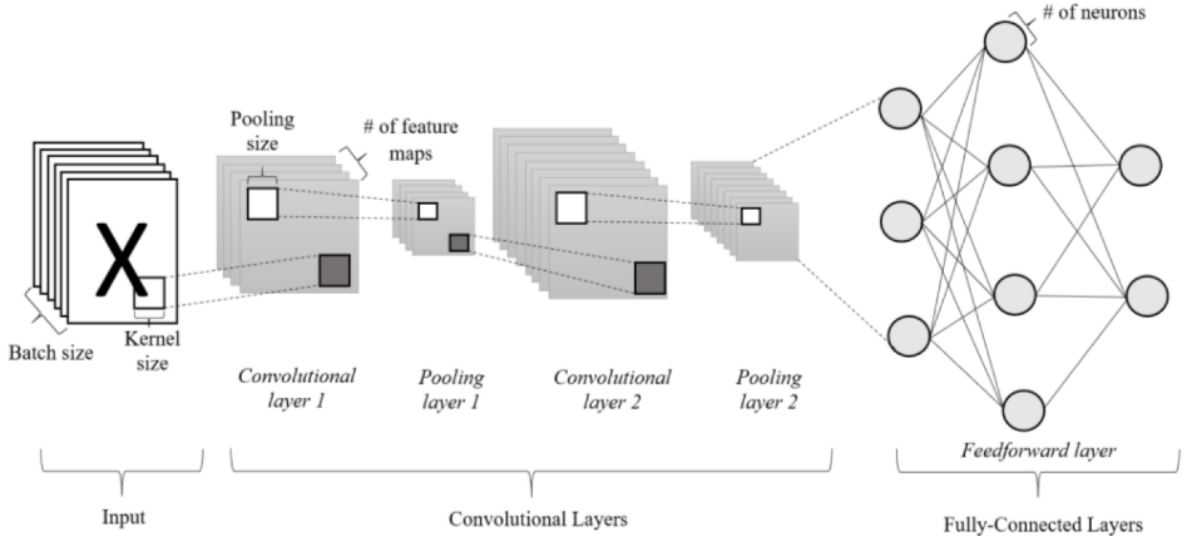



**Figure 2.5:** Convolutional neural network

### 2.2.3. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a class of neural networks designed to process sequential data, where the output from the previous time step is fed as an input to the network for the current time step. This allows RNNs to maintain a memory of previous inputs and outputs, making them particularly effective for tasks where context or temporal dependencies are important, such as language modeling, time-series prediction, and sequence classification.

In an RNN, the hidden state $h_t$ at time step $t$ is computed based on the input at the current time step $x_t$ and the hidden state from the previous time step $h_{t-1}$. The hidden state $h_t$ serves as the memory of the network, encapsulating information from all previous time steps. The output $o_t$ at each time step is computed using the hidden state $h_t$.

The mathematical operations for a single forward pass through an RNN can be described by the following equations:

$$h_t = \sigma(W_h \cdot x_t + U_h \cdot h_{t-1}) \tag{2.7}$$

$$o_t = \sigma(W_o \cdot h_t) \tag{2.8}$$

Here, $W_h$ and $U_h$ are the weight matrices that connect the input $x_t$ and the previous hidden state $h_{t-1}$ to the current hidden state $h_t$, respectively. $W_o$ is the weight matrix that connects the hidden state $h_t$ to the output $o_t$. The function $\sigma$ typically represents a non-linear activation function, such as a sigmoid or tanh.

RNNs are particularly powerful for modeling sequential data because they allow information to persist across time steps. However, traditional RNNs can suffer from issues such as vanishing and exploding gradients during training, which limit their ability to learn long-term dependencies. Despite these challenges, RNNs have been successfully applied in various domains, including natural language processing, speech recognition, and video analysis.

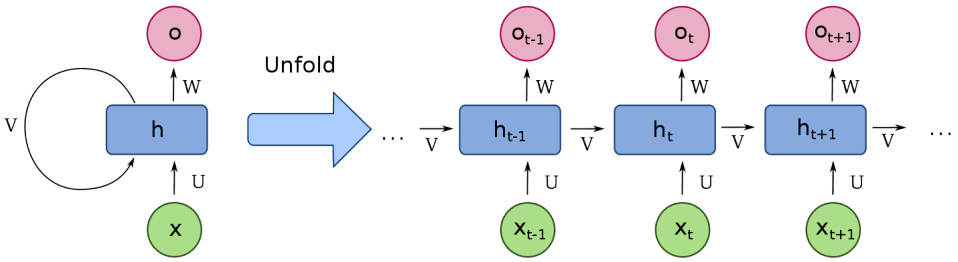



**Figure 2.6:** Illustration of a Recurrent Neural Network (RNN) architecture, showing how the hidden state $h_t$ is propagated across time steps.

### 2.2.4. Long-Short-Term-Memory Networks (LSTM)

Long Short-Term Memory (LSTM) networks are an advanced version of traditional Recurrent Neural Networks (RNNs) that have largely replaced them in many applications. Traditional RNNs often struggle to maintain long-term dependencies and context from earlier time steps in a sequence. They are also prone to issues such as exploding and vanishing gradients, which can hinder the training process. LSTMs address these challenges by incorporating memory cells equipped with gates.

A typical LSTM memory cell, as shown in Fig. 2.7, is composed of three types of gates: the forget gate $f_t$, the input gate $i_t$, and the output gate $o_t$. The forget gate $f_t$ determines the amount of information to retain or discard from the memory. The input gate $i_t$ controls the information that should be added to the memory cell. The output gate $o_t$ regulates when the information stored in the memory cell should be used. The cell state, denoted by $C_t$, carries information across different layers and time steps, enabling the network to maintain long-term dependencies.

The mathematical operations for a single forward pass through an LSTM cell are described by the following equations:

$$f_t = \sigma(U_f \cdot x_t + W_f \cdot h_{t-1}) \tag{2.9}$$

$$i_t = \sigma(U_i \cdot x_t + W_i \cdot h_{t-1}) \tag{2.10}$$

$$a_t = \tanh(U_a \cdot x_t + W_a \cdot h_{t-1}) \tag{2.11}$$

$$o_t = \sigma(U_o \cdot x_t + W_o \cdot h_{t-1}) \tag{2.12}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot a_t \tag{2.13}$$

$$h_t = o_t \odot \tanh(C_t) \tag{2.14}$$

In these equations, $U_f, U_i, U_a, U_o$ are the weight matrices associated with the input vector, and $W_f, W_i, W_a, W_o$ are the weight matrices associated with the hidden state from the previous time step. The operations $\sigma$

and $\tanh$ represent the sigmoid and hyperbolic tangent activation functions, respectively. The element-wise multiplication is denoted by $\odot$.

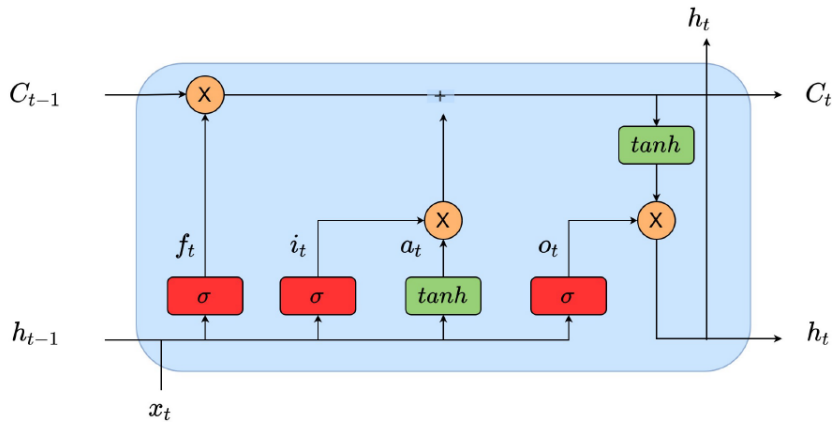



**Figure 2.7:** LSTM cell composed of forget gate $f_t$, input gate $i_t$, and output gate $o_t$

### 2.2.5. Spiking Neural Networks (SNNs)

Spiking Neural Networks (SNNs) represent a significant departure from traditional artificial neural networks (ANNs) by mimicking the way biological neurons communicate through discrete events known as spikes. Unlike ANNs, where neurons communicate using continuous activations, SNNs operate in the spatio-temporal domain with neurons generating spikes only when a certain membrane potential threshold is reached. This event-driven mechanism allows SNNs to be highly energy-efficient, as computation occurs only during spikes, making them particularly suitable for low-power neuromorphic hardware.

In SNNs, the basic processing unit is a spiking neuron, typically modeled by integrating incoming spikes (which are weighted) over time. When the accumulated membrane potential $u(t)$ crosses a certain threshold $u_{th}$, the neuron generates an output spike, which is then propagated to connected neurons. The neuron then resets its potential, starting a new cycle of integration. The entire process can be mathematically described as:

$$u(t) = \sum_i w_i \cdot S_i(t) \tag{2.15}$$

$$S(t) = \begin{cases} 1 & \text{if } u(t) \geq u_{th} \\ 0 & \text{otherwise} \end{cases} \tag{2.16}$$

Here, $w_i$ are the synaptic weights, and $S_i(t)$ represents the incoming spikes from presynaptic neurons. Fig. 2.8 illustrates the key differences between a traditional ANN neuron and a spiking neuron, highlighting the continuous activation in ANNs versus the binary spike behavior in SNNs.

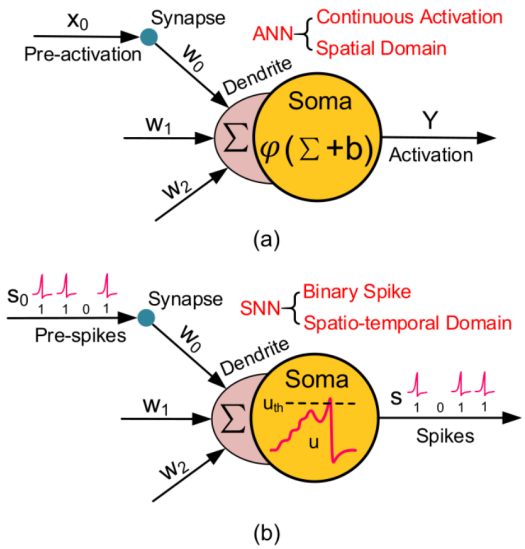



**Figure 2.8:** Comparison between (a) a traditional ANN neuron and (b) a spiking neuron in an SNN. ANNs operate in the spatial domain with continuous activations, while SNNs operate in the spatio-temporal domain with binary spikes[51].

### Converting Traditional Networks to Spiking Counterparts

Converting conventional networks like CNNs, RNNs, and LSTMs into their spiking counterparts (SCNN, SRNN, SLSTM) involves adapting the neurons and network architecture to handle spike-based communication and computation.

**Spiking Convolutional Neural Networks (SCNNs)** — To convert a Convolutional Neural Network (CNN) into a Spiking CNN (SCNN), the activation functions in the convolutional layers are replaced with spiking neurons that emit spikes when the membrane potential exceeds a threshold. Additionally, the inputs (typically images) are converted into spike trains, either using rate coding, where the intensity of a pixel is translated into the frequency of spikes, or temporal coding, where the timing of spikes encodes information.

**Spiking Recurrent Neural Networks (SRNNs)** — In Spiking RNNs (SRNNs), the recurrent connections are preserved, but the neurons operate in the spiking domain. The hidden states in traditional RNNs, which are continuous, are replaced by spike trains. SRNNs can maintain temporal dependencies by accumulating spikes over time, similar to how they maintain context in traditional RNNs.

**Spiking Long Short-Term Memory Networks (SLSTMs)** — Converting an LSTM into a Spiking LSTM (SLSTM) is more complex due to the presence of gates and continuous cell states. However, this can be achieved by discretizing the gating mechanisms, allowing them to operate with spikes. The cell state can be implemented as a leaky integrate-and-fire neuron, which maintains the memory through spiking behavior.

The primary challenge in converting these networks into their spiking counterparts lies in encoding information efficiently in the form of spikes and ensuring that the temporal dynamics of the spiking neurons align with the task-specific requirements. However, when done correctly, these spiking networks can inherit the advantages of both traditional neural networks and the energy efficiency of SNNs, making them suitable for deployment in neuromorphic hardware.

## 2.3. Hardware Aware Optimizations

In this section, I will discuss how I employed two key techniques—batch normalization folding and quantization—to enhance the hardware efficiency of my compression system.

### 2.3.1. Batch Normalization Folding in Neural Networks

Batch normalization is a technique widely used in neural networks to stabilize and accelerate training by normalizing inputs to each layer, ensuring they have zero mean and unit variance. This process helps mitigate issues like internal covariate shift. During training, batch normalization introduces additional layers, but these layers can be ”folded” into the preceding layers during inference to simplify the network and reduce computational overhead.

The process begins with calculating the mean $\mu_{\text{batch}}$ and variance $\sigma^2_{\text{batch}}$ over a mini-batch:

$$\mu_{\text{batch}} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.17}$$

$$\sigma^2_{\text{batch}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\text{batch}})^2 \tag{2.18}$$

The input is then normalized:

$$\hat{x}_i = \frac{x_i - \mu_{\text{batch}}}{\sqrt{\sigma^2_{\text{batch}} + \epsilon}} \tag{2.19}$$

where $\epsilon$ is a small constant for numerical stability. This normalized input is scaled and shifted using learnable parameters $\gamma$ and $\beta$:

$$y_i = \gamma\hat{x}_i + \beta \tag{2.20}$$

During inference, batch statistics are replaced by running averages computed during training. The batch normalization parameters $\gamma$, $\beta$, $\mu_{\text{running}}$, and $\sigma^2_{\text{running}}$ can be folded into the preceding layer's weight $W$ and bias $b$, computed as:

$$W' = \frac{\gamma}{\sqrt{\sigma^2_{\text{running}} + \epsilon}} \cdot W \tag{2.21}$$

$$b' = \frac{\gamma}{\sqrt{\sigma^2_{\text{running}} + \epsilon}} \cdot (b - \mu_{\text{running}}) + \beta \tag{2.22}$$

Folding batch normalization into the network reduces its complexity, eliminating the need for separate batch normalization operations during inference. This reduction in computational load and memory usage is particularly beneficial for edge devices, such as smartphones and IoT devices, where computational resources and power are limited. By integrating batch normalization directly into the layers, the model becomes more efficient, enabling real-time processing and deployment on resource-constrained hardware without sacrificing performance.

### 2.3.2. Quantization in Neural Networks and Symmetric Quantization

Quantization in neural networks is a process where continuous (floating-point) values of weights, biases, and activations are mapped to discrete levels, typically represented by lower precision integers. This conversion reduces the memory footprint and computational complexity, enabling efficient deployment of neural networks on hardware with limited resources, such as edge devices. Quantization can also speed up inference by allowing the use of integer arithmetic, which is faster and more energy-efficient than floating-point operations.

In symmetric quantization, the range of values is symmetrically distributed around zero, meaning that the quantization levels for positive and negative values are mirrored. This is particularly useful in neural networks because weights and activations are often centered around zero, allowing for a more uniform distribution of quantization errors. Symmetric quantization is mathematically defined as:

$$q(x) = \text{round}\left(\frac{x}{\Delta}\right) \cdot \Delta \tag{2.23}$$

where $\Delta$ is the quantization step size, determined by:

$$\Delta = \frac{\text{max}(|x_{\text{min}}|, |x_{\text{max}}|)}{2^b - 1} \tag{2.24}$$

Here, $b$ represents the number of bits used for quantization, and $x_{\text{min}}$ and $x_{\text{max}}$ are the minimum and maximum values of the input range.

When quantizing spiking neural networks (SNNs), the neuron's membrane potential and spike generation threshold must be carefully scaled to maintain the accuracy of spike timing and the overall network behavior. In SNNs, a neuron generates a spike when its membrane potential $u(t)$ exceeds a threshold $u_{\text{th}}$. After quantization, both $u(t)$ and $u_{\text{th}}$ must be scaled to the reduced precision format:

$$\hat{u}(t) = \text{round}\left(\frac{u(t)}{\Delta_u}\right) \cdot \Delta_u \tag{2.25}$$

$$\hat{u}_{\text{th}} = \text{round}\left(\frac{u_{\text{th}}}{\Delta_u}\right) \cdot \Delta_u \tag{2.26}$$

where $\Delta_u$ is the quantization step size for the membrane potential. It's crucial that $\hat{u}_{\text{th}}$ is accurately scaled to prevent deviations in spike timing, which could lead to significant errors in the network's output.

In practice, symmetric quantization in SNNs simplifies the hardware implementation, as the same scaling factor can be used for both positive and negative values. Additionally, the uniformity in quantization step size ensures consistent behavior across different layers of the network. Proper scaling of thresholds during quantization helps maintain the integrity of spike-based communication, which is essential for preserving the functionality of the SNN after quantization.

# 3

# Related Work

*This chapter presents a comprehensive review of the current state-of-the-art compression techniques, highlighting their key methodologies, advancements, and inherent limitations. By examining these approaches, we aim to identify the challenges that persist in the field and set the stage for potential improvements and innovations in compression technology.*

Over the past decade, significant advancements have been made in the field of neural signal compression, aimed at addressing the critical issues of transmission bandwidth and power consumption. This section will delve into several noteworthy methods proposed and developed for compressing neural signals, focusing on those most pertinent to our research.

## 3.1. Transformations

This type of compression aims to transform the neural signal into a different domain where the signal is sparse and a reduced number of parameters can be efficiently identified and quantized to represent action potentials and thus more suitable for compression and transmission (such as frequency or wavelet domain). Commonly used techniques include, e.g. Principal Component Analysis (PCA), Discrete Cosine Transform (DCT), wavelet, linear approximations, etc [55].

Zhang [59] mentions that the hardware requirements for such transformations are substantial, as they often necessitate complex digital filters and additional memory operating at high speeds, which significantly increases the circuit area and on-chip power consumption. Moreover due to the signal-agnostic natures of the mentioned methods they are more often not more optimized for compression resulting in lack of representational power in the transformed domains leading to limited compression [55]. This makes these methods less feasible for large-scale neural recording systems that require efficiency and scalability.

## 3.2. Delta Modulation for Neural Recording Systems

Delta Modulation is particularly beneficial for sparse action potential signals, such as those found in electrophysiology, because it inherently reduces data rates by focusing on difference or ‘delta’ between successive signal samples rather than transmitting redundant information. In the context of sparse neural signals, where action potentials (spikes) occur intermittently, Delta Modulation only records and transmits the significant changes (the spikes) while ignoring the long periods of inactivity. This selective sampling is highly efficient, reducing the amount of data that needs to be processed and transmitted, which is crucial for resource-constrained systems like Brain-Computer Interfaces (BCIs).

He et al. [16] expanded upon this concept by integrating a Delta-modulated Analog-to-Digital Converter (Δ-ADC) within an event-based framework. The Δ-ADC triggers an event only when the difference (Δ) between consecutive samples exceeds a predefined threshold, minimizing the generation of unneces-

sary data, where x(ti) and x(ti-1) are the signal values at consecutive time points, and Δ is the change that prompts data transmission. By focusing on significant changes in the signal, He's system efficiently compresses the data while preserving essential information.

$$\Delta = x(t_i) - x(t_{i-1}) \tag{3.1}$$

In our work, we leverage the efficiency of He's Δ-ADC as a foundational component within our system. In addition, we integrate Spiking Neural Networks (SNNs), which enables our system to compress the already sparse and event-driven data from the Δ-ADCs further.

By aligning the event-driven properties of the Δ-ADCs with the event-based processing capabilities of SNNs, my system can maintain a high signal-to-noise and distortion ratio (SNDR), ensuring that the compressed data remains highly representative of the original neural signals[34].

## 3.3. Block-based compression

The block-based method (epoch) is a well-established technique in neural signal processing, particularly in Brain-Computer Interfaces (BCIs). It involves segmenting continuous neural data into discrete time intervals called epochs, which align with specific events or periods of interest. This segmentation allows for focused analysis and compression of the most relevant data segments, reducing overall data volume and computational load [7].

In BCIs, epochs are often selected to coincide with specific stimuli or actions, such as motor imagery tasks or event-related potentials (ERPs). For example, an epoch might begin just before a visual cue and extend through the duration of the neural response to capture the most pertinent data [35]. This targeted segmentation reduces the overall data volume and allows for more efficient processing, as only the relevant portions of the signal are analyzed. Let's denote the continuous neural signal as $x(t)$, where $t$ represents time. The signal is divided into $N$ epochs, each of duration $T$ seconds. The $n$-th epoch, $E_n(t)$, can be defined as in Equation (3.2).

$$E_n(t) = x(t) \times w_n(t) \tag{3.2}$$

where $w_n(t)$ is a window function that extracts the $n$-th epoch. This window function is defined as:

$$w_n(t) = \begin{cases} 1, & \text{if } nT \leq t < (n+1)T \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

While effective, the epoch method has limitations, such as the challenge of capturing continuous, non-time-locked activities and the potential for significant memory usage, as noted by Biederman et al. [3], where each channel demands around 1500 bits for processing the incoming action potentials, potentially increasing system complexity and power consumption.

In our work, we use the epoch method as a baseline to highlight the advantages of our proposed system, which integrates Delta encoding with Spiking Neural Networks (SNNs).

## 3.4. Compressed Sensing

Compressed Sensing (CS) has emerged as a prominent technique to address the limitations of traditional sampling methods, particularly in scenarios requiring high efficiency and low resource utilization. Zhang et al. [59] explored the application of Compressed Sensing in neural signal acquisition systems, demonstrating its potential to significantly reduce the amount of data required for accurate signal reconstruction. The CS method works by acquiring 'Sub-Nyquist' samples, where the signals are sampled at a frequency lower than the Nyquist rate, under the assumption that the signals are sparse in some

domain. This is achieved using a sampling matrix (Ø) that compresses the signal directly during the acquisition process, thereby circumventing the need to store all samples before compression.

Zhang's work involves transforming the traditionally sampled signals into sparse representations using a transformation basis $(\eta)$, which is then compressed to form a much smaller dataset (as seen in 3.4). The compressed data is then processed to reconstruct the original signal by solving an optimization problem to find the sparse solution, a process which can be computationally intensive and is typically performed off-chip.

$$\begin{aligned} y &= \Phi x \\ &= \Phi(\eta s) \end{aligned} \tag{3.4}$$

However, Zhang's method, while effective, is limited by its complexity in real-time applications, as the optimization required for signal reconstruction poses challenges for on-chip implementation. This is particularly evident in the peak SNDR (Signal-to-Noise and Distortion Ratio) performance, where Zhang's method struggles to maintain high fidelity in signal reconstruction, particularly at higher compression ratios.

In comparison to state-of-the-art methods, our method significantly advances beyond Zhang's approach by integrating Delta and SNN (Spiking Neural Networks) encoding techniques, achieving higher compression ratios. Our work overcomes the limitations of Zhang's approach by eliminating the need for off-chip optimization, enabling real-time processing.

## 3.5. Autoencoders

Autoencoders, a class of deep neural network frameworks, are employed for data compression by transforming input data into a lower-dimensional representation. This transformation is governed by the parameter $W$ and its associated nonlinearity $\Phi$. The encoder, which constitutes the first stage of the autoencoder, generates this compressed representation, enabling efficient data compression. Subsequently, the decoder, defined by the transformation parameter $W'$ and nonlinearity $\Phi'$, reconstructs the original data from the compressed representation, as described by Equation (3.5). The primary objective is to minimize the loss between the reconstructed output and the original input, thereby optimizing the parameters of both the encoder and decoder ($W$ and $W'$). To achieve this minimization, traditional optimization techniques, such as gradient descent, are employed.

$$y = \Phi'(W'\Phi(Wx)) \tag{3.5}$$

This encoder-decoder structure is utilized for its inherent design of compression, which effectively reduces the dimensionality of the input data while retaining the most critical information. This capability makes the encoder-decoder network particularly suitable for applications requiring efficient data representation and reconstruction. A high-level diagram of this architecture is shown in Fig. 3.1.

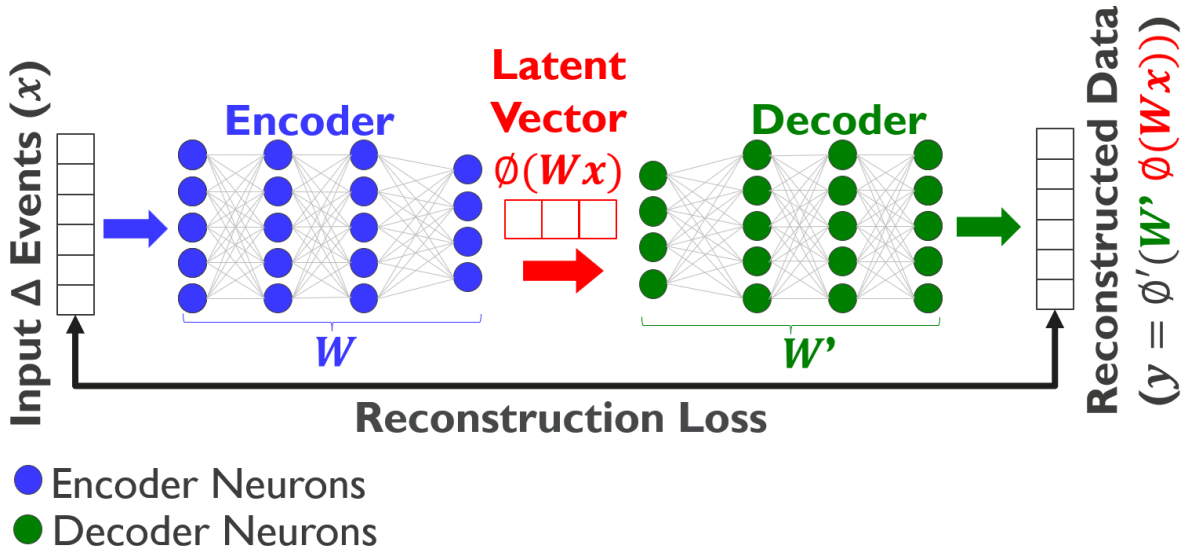



**Figure 3.1:** Schematic diagram of the encoder-decoder network architecture showing the nonlinear ($\Phi$ and $\Phi$') latent vector and reconstruction transformation.

Previous research on autoencoders for neural compression has primarily focused on EEG and ECG signals or, when using extracellular recordings, has utilized local field potentials (LFPs) as input data [52], [45], [50]. In contrast, this work employs extracellular in vivo spiking data, which provides significant advantages in decoding accuracy and temporal resolution compared to LFPs [10]. The only comparable work is by Wu et al. [55], which uses a CNN-based asymmetric autoencoder.

Wu's method, while effective in using CNNs for neural data compression, faces significant challenges due to its high computational and memory demands—requiring 79.25K MAC operations and 237.75K memory accesses per spike. This makes it less suitable for resource-constrained environments like implantable devices, where power efficiency and low latency are crucial.

# 4

# Proposed Neuromorphic Compression System

*This chapter provides an in-depth overview of the proposed Neuromorphic Compression System, detailing the individual components, the training methodology, and the hardware optimizations implemented.*

The proposed system represents a notable advancement in neural data compression and processing, especially for resource-constrained environments like implantable devices. Unlike traditional methods such as compressed sensing [59], epoch[3] and CNN-based autoencoders[55], which often struggle with hardware complexity, computational demands, and the need for significant buffering that increases memory usage, our system is optimized for power efficiency and real-time processing.This makes it especially well-suited for on-implant applications, where minimizing both computational load and memory footprint is critical.

The Neuromorphic Compression system, shown in Fig. 4.1, is at the heart of this innovation, integrating two key components: an Delta-modulated Analog-to-Digital Converter (Δ-ADC)[16] and a Neuromorphic Autoencoder. It achieves nearly five times greater data reduction and significantly reduces computational demands compared to current methods, enhancing storage efficiency and data transmission speed. Our approach is further refined by using knowledge distillation, where the on-implant SNN is trained in two stages with a more powerful teacher network. The encoder first compresses the data, and the decoder is fine-tuned to maintain signal fidelity.Additionally, by grouping inputs from nearby electrodes, we reduce memory usage with minimal impact on signal reconstruction. These advancements position our system as a highly efficient solution for neural data processing, ideal for advancing implantable device technology.

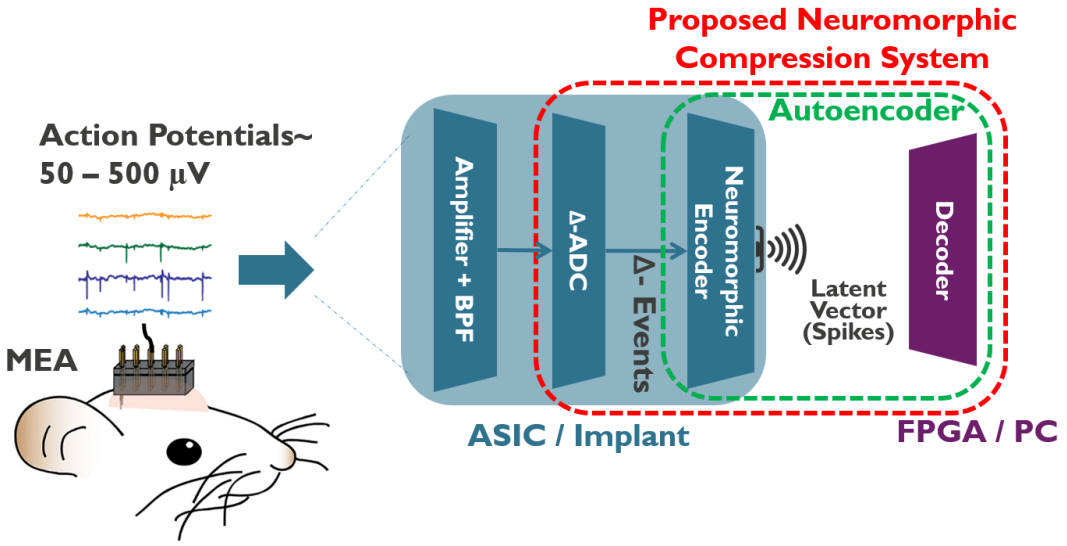



**Figure 4.1:** High-Level architecture for our Neuromorphic Compression System. The system processes action potentials which are apmlified and filtered from a Multielectrode Array (MEA).

The primary function of the amplifier and filter before the proposed neuromorphic compression system is to enhance the raw signals from the microelectrode array (MEA), as shown in Fig. 4.1. The raw data is very low in amplitude (50-500 µV) and hence needs to be amplified before it can be processed. Additionally, a band-pass filter is applied to isolate the spikes (300 Hz – 6 kHz)[42].

## 4.1. Delta-modulated Analog-to-Digital Converter (Δ-ADC)

The Δ-ADC is crucial in converting amplified analog action potentials into digital signals, facilitating the transformation of raw action potentials into delta (Δ) - modulated events as detailed in [16]. This delta modulation technique efficiently represents action potentials in a digital format, essential for subsequent processing steps.

Neural data can be divided into two categories: spikes, which are fast-changing, short-lasting signals from a single neuron, and local field potentials (LFPs), which are slow-changing, long-lasting activities from the collective signals of large neural populations. While both types of signals carry valuable information, spikes are particularly rich in spatial and temporal details, making them more critical for precise neural decoding [58].

In our system, Local Field Potentials (LFPs) are effectively filtered out using a bandpass filter, allowing us to focus exclusively on spikes, which contain the most relevant information. The subsequent delta modulation process computes the difference between consecutive samples, generating non-zero delta values only when significant changes occur. This approach ensures that events are generated only when thresholds are crossed, aligning with an event-based paradigm. This technique effectively reduces the dynamic range of the data and ensures that it remains sparse over time, optimizing both the data compression and the system's overall efficiency. By concentrating on spikes and filtering out LFPs, we enhance the system's ability to capture and process the most informative aspects of neural activity, leading to more accurate and efficient neural data processing.

Fig. 4.2 illustrates the process of delta modulation, where the top panel shows the raw action potentials (orange) and the bottom panel displays the detected level delta events (blue). This Δ-ADC is selected for its significant advantages in managing the challenges of intracortical brain-computer interfaces (BCIs). It addresses the high data volume from numerous recording channels by implementing a high compression ratio ($CR_{\triangle ADC}$) in the range of 10-100$\times$ on extracellular neural recordings, substantially reducing the data that needs to be transferred. Additionally, it offers low power consumption, with

each channel consuming only 1 $\mu$W, and maintains spike waveform fidelity with a Signal-to-Noise and Distortion Ratio (SNDR) of 36 dB even for spike amplitudes as low as 31 $\mu$V, ensuring accurate neural signal representation.

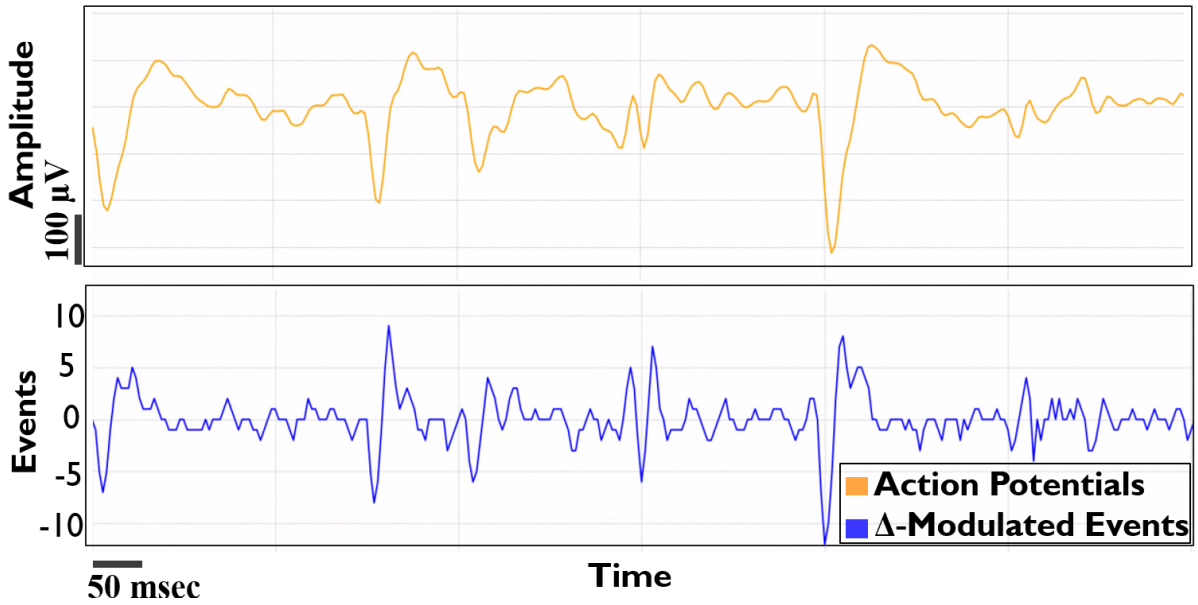



**Figure 4.2:** The top panel (orange) depicts the reconstructed waveform through cumulative summation while the bottom panel (blue) shows delta-modulated level crossing events over time.

## 4.2. Neuromorphic Hybrid Autoencoder

The Neuromorphic Hybrid Autoencoder is the core of our system, designed to compress and process neural data with exceptional efficiency. Starting with the event-based output from the $\Delta$-ADC, the autoencoder's primary function is to significantly reduce data size while preserving essential features, optimizing both storage and transmission. This is accomplished through a robust encoder-decoder network which is meticulously trained using gradient descent to minimize reconstruction loss.

To further enhance efficiency, knowledge distillation is employed to transfer the learned features from the teacher model to a more efficient, event-driven Spiking Neural Network (SNN). This transfer occurs in two stages, with the encoder and decoder being trained successively. The result is a final SNN that inherits the accuracy of the larger model while operating with significantly lower computational and power demands, making it ideal for edge devices and resource-constrained environments.

The following subsections will provide detailed explanations of the encoder-decoder network's architecture and training process, the application of knowledge distillation, and the final implementation of the Spiking Neural Network (SNN). These components work together to deliver the high levels of compression and efficiency that define the crux of our system.

### 4.2.1. Encoder − Decoder Network

The encoder-decoder architecture is a widely utilized sequence-to-sequence transformation network [17], which we have adapted to process the input data—specifically, the output from the $\Delta$-ADC in our case—by compressing it into a significantly lower-dimensional representation, commonly referred to as the latent space or latent vector. The decoder then reconstructs the original input from this latent space.

As described in Equation 3.5 and illustrated in Fig. 3.1, the reconstructed data y is a nonlinear function derived from the latent space. The weight parameters W and W′ are traditionally optimized using reconstruction loss, which is minimized by applying the gradient descent algorithm to fine-tune these

parameters [26]. Our implementation employs a three-hidden layer encoder network with a symmetrically identical decoder. The rationale for this design choice will be discussed in detail in Chapter 6.

### 4.2.2. Knowledge Distillation

Knowledge distillation, also known as the teacher-student architecture, is a powerful technique for transferring knowledge from a large, complex neural network (the teacher) to a smaller, more lightweight model (the student) [17]. This approach is essential for achieving compatibility with edge devices due to the constraints on computational resources and power consumption.

In our proposed method, we utilize a very large and powerful teacher network. Both the encoder and decoder neurons in the teacher network (as shown in Fig. 4.3) have been tested with various types of cells, including Convolutional Neural Networks (CNN), Spiking Neural Networks (SNN), Long Short-Term Memory (LSTM), and Recurrent Neural Networks (RNN). Our results, as we will see in Section 6.1, indicate that RNN and LSTM cells are the most effective.

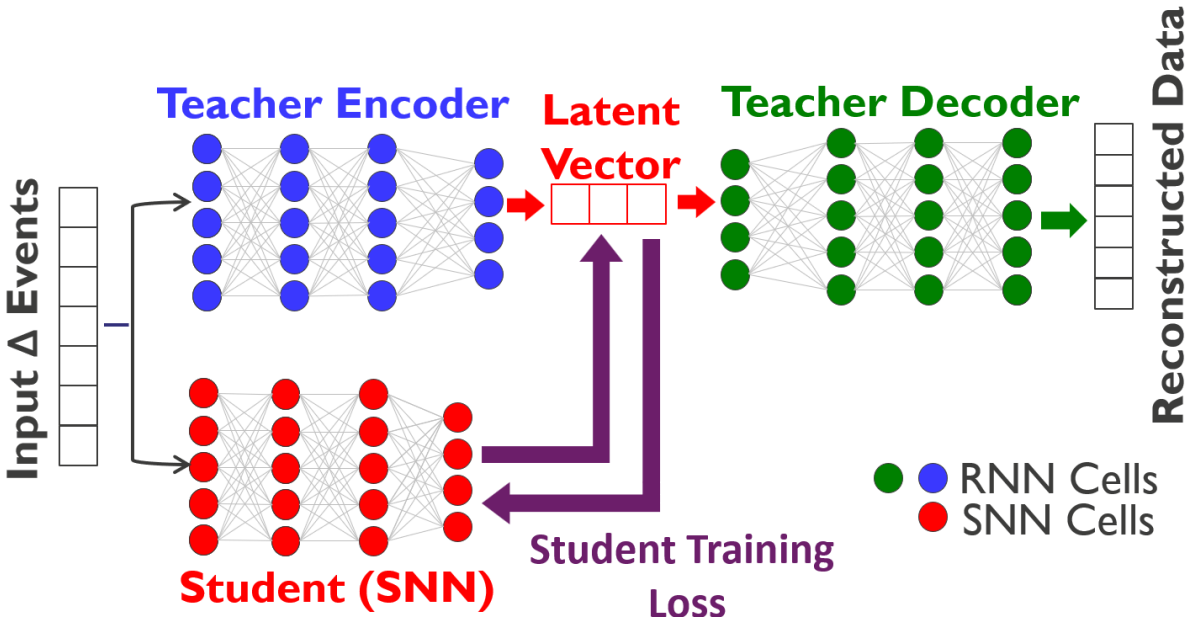



**Figure 4.3:** Knowledge distillation process in teacher – student architecture.

The reconstruction error and compression ratio of the network can be tuned by adjusting the size of the latent vector. After training the teacher autoencoder network, the latent vector and decoder weights are fixed to train the student network. The visual representation of this process is illustrated in Fig. 4.3.The subsequent section will elaborate on the rationale behind choosing an SNN for the student network.

### 4.2.3. Spiking Neural Network

The final component that completes our system is the implementation of the student network as a Spiking Neural Network (SNN). This choice leverages the unique operational characteristics of SNNs, which closely mimic the behavior of biological neurons, thereby achieving high power efficiency. Unlike traditional neural networks that use continuous values for layer-to-layer communication, SNNs utilize binary values, known as spikes (0 or 1). This transition simplifies the traditional Multiply-And-Accumulate (MAC) operation to a mere accumulate operation, facilitating more efficient event-driven hardware operations [46]. Due to the sparsely distributed nature of inputs, SNNs can significantly reduce both power consumption and hardware overhead. Fig. 4.4 illustrates the activity of a single neuron in an SNN.

In an SNN cell, the neuronal state is represented by an intrinsic state called the membrane potential

(MP). The MP changes in response to incoming inputs and, upon reaching a specific threshold, the neuron generates a spike to the subsequent layer while resetting the MP to zero (hard reset). This temporal storage mechanism allows the SNN to capture the dynamic temporal characteristics of input data, reacting only to events. Since the output of the Analog-to-Digital Converter (ADC) is also obtained through event-based sampling, this alignment enhances the overall efficiency of the system.

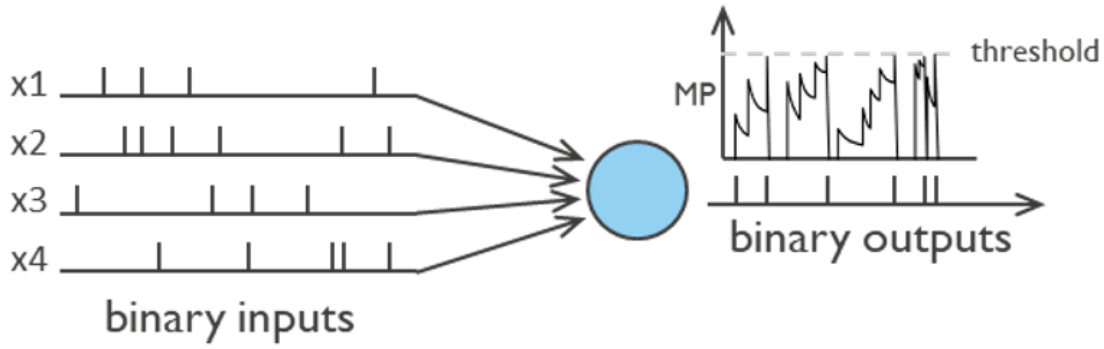



**Figure 4.4:** Illustration of a binary input system processing signals through a neuron model. The membrane potential (MP) is compared against a threshold to generate binary outputs.

Additionally, the decision to choose SNNs was influenced by their event-based nature, particularly due to the integration-and-fire operation. As the output from the ADC (referenced from [16]) is sampled in an event-based fashion, spikes are generated in the SNN only when an event occurs. Once the spiking neuron accumulates these events to a certain threshold, it fires. This event-based sampling nature of the ADC and the subsequent accumulation operation inherently introduces a degree of compression even without dimensionality reduction.

Despite the inherent hardware efficiency and biological plausibility of Spiking Neural Networks (SNNs, their learning capabilities can be limited by the temporal precision required for event-based communication. In our system, some information loss may occur due to the use of a lower frequency clock, which may not fully capture the temporal dynamics encoded in the spikes [43].To address this issue, we utilize knowledge transfer from a larger and more complex network, as discussed in section 4.2.2 (and illustrated in Fig. 4.3). This approach aims to recover a substantial amount of accuracy, thereby making SNNs suitable for implementation on edge devices.

## 4.3. Training the Spiking Neural Network

Our system utilizes two primary innovative strategies to train the teacher-student autoencoder structure. These strategies are designed to ensure that the spiking student network optimally compresses the events from the $\Delta$-ADC. Detailed explanations of these strategies are provided in the subsequent subsections.

### 4.3.1. 2 Stage Student Encoder-Decoder Training

After training the teacher network, we fix its weights and proceed with the training of the student network in two distinct stages. This staged approach allows the student network to effectively capture and replicate the essential features learned by the teacher network, ensuring that the critical aspects of the data compression process are preserved. This method is inspired by layer-wise training techniques, which have been shown to improve the stability and performance of deep networks by isolating the learning process of different components[17][53]. By fixing the encoder weights after initial training, we ensure that the student network benefits from a robust feature extraction process before fine-tuning the decoder.

In the first stage, we train only the encoder of the student network while utilizing the teacher's decoder

with its weights fixed. The loss function used during this stage is the mean squared error (MSE) between the latent spaces of the student and teacher networks. This strategy ensures that the encoder of the student network closely mimics the behavior of the teacher network, effectively performing layer-by-layer training. This process is illustrated in Fig 4.5(a).

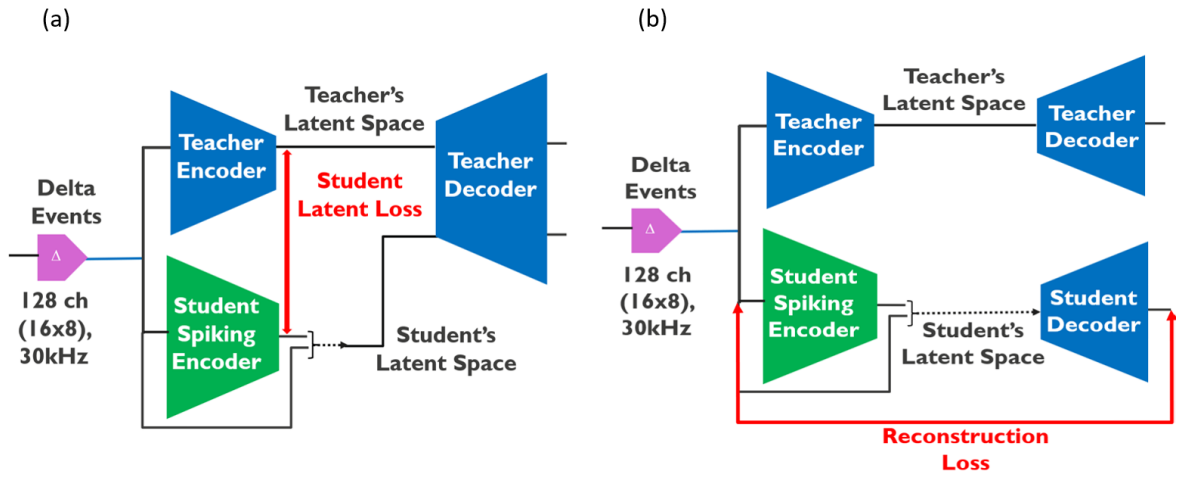



**Figure 4.5:** (a) Stage 1: student encoder training using the teacher's latent space. (b) Stage 2: student decoder training using the reconstruction loss

In the second stage, we focus on training the student decoder, which is initialized with the pre-trained weights of the teacher decoder. During this phase, we fix the weights of the student encoder, allowing the decoder to adapt to the latent space learned in the first stage. This step fine-tunes the student decoder to its corresponding latent space, ensuring cohesive and efficient compression. The detailed process of this stage is depicted in Fig. 4.5(b).

By adopting this two-stage training method, we ensure that the student network effectively learns to compress data by leveraging the knowledge distilled from the teacher network, leading to significant improvements in performance and efficiency. Through simulations, we demonstrate that this approach achieves a remarkable 9.4 × compression in the final spiking layer of the encoder, highlighting the substantial benefits of the two-stage training process. These results underscore the effectiveness of our method in optimizing data compression while maintaining high levels of accuracy, making it an ideal solution for resource-constrained environments.

### 4.3.2. Hardware Aware Optimizations

To enhance the system's efficiency for deployment across various hardware platforms, it was essential to optimize the number of weights in the student Spiking Neural Network (SNN) designed for the implant. This optimization process involved both pruning and quantization of the weights. Pruning effectively streamlined the network by eliminating weights that were deemed less important. Subsequently, the remaining weights were quantized, given that PyTorch, by default, utilizes the float32 datatype. By converting these weights to int6, we improved memory efficiency, achieving nearly 1.5× reduction in bits per channel compared to the state-of-the-art deep learning model used for compression (will see in Section 6.7). These optimizations resulted in a more efficient and compact model, making it highly suitable for hardware implementation.

Leveraging the principle that neurons primarily communicate with their nearest neighbors, we focused on pruning weights connected to more distant neurons. This neuromorphic approach mirrors biological neural networks, where synaptic inputs from nearby neurons are prioritized over those from farther ones. By emulating this natural efficiency, we significantly reduced the computational burden of our SNN. Pruning distant connections helped eliminate redundant or less influential synapses, which in turn reduced the complexity and energy consumption of the network. This biologically inspired strategy ensured that the core functionality and performance of the SNN were preserved while minimizing the

hardware resources required for its implementation [21],[25] .

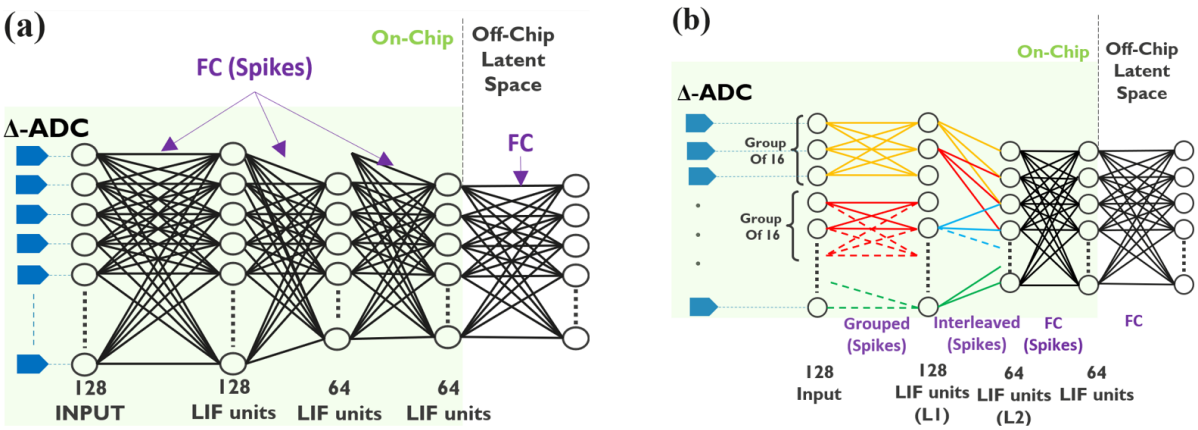



**Figure 4.6:** Comparison of (a) fully connected network and (b) pruned network with grouping (in L1) and interleaving (in L2).

As illustrated in Fig. 4.6, we have employed various strategies for pruning the network. The ADC inputs are directly mapped to the input layer, with the weights connecting to the first hidden layer (L1) pruned in a manner that mirrors the ADC's information processing. This process is organized into eight groups of 16 electrodes each. Within each group, the 16 electrodes are closely correlated due to their spatial proximity, as depicted in Fig. 4.7a, which shows the grouping from the input layer to the first hidden layer (L1).

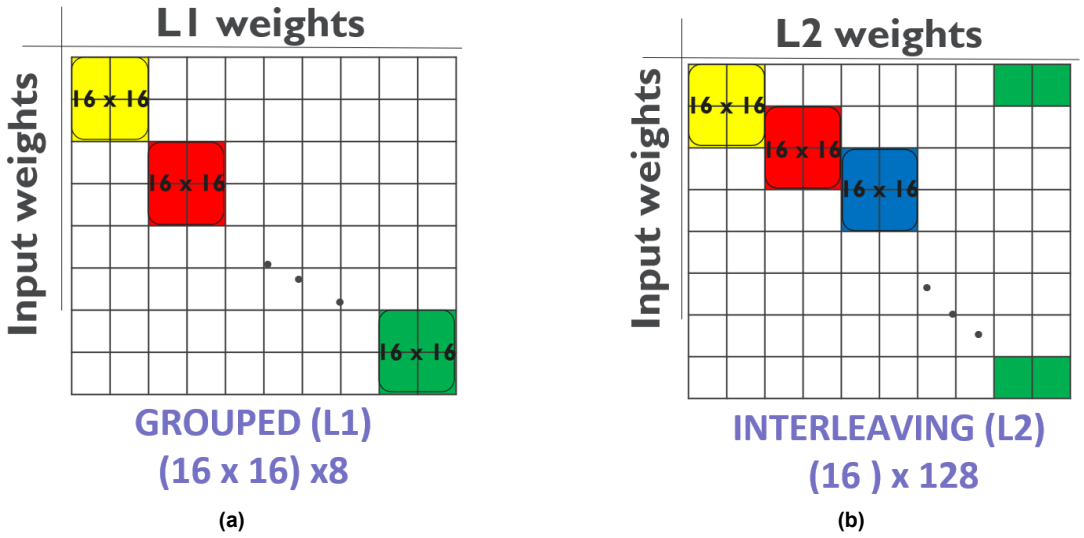



**Figure 4.7:** Illustration of (a) weight matrix for input layer L1 and (b) weight matrix for input layerL2

The interleaving strategy employed from the second layer onward significantly enhances the neural network's capacity to identify and learn intricate patterns across different inputs, such as those originating from multiple electrodes. This sophisticated approach is visually represented in Figure 4.7b, where the synaptic weights are meticulously organized within a 16×16 matrix. Each neuron in the second layer is connected not to all neurons in the subsequent layer, but rather to a strategically selected subset of 16 neurons, with these connections exhibiting a deliberate overlap to foster comprehensive input coverage. For instance, the first neuron in the second layer is connected to neurons 1 through 16 in the next layer, forming a concentrated cluster of connections. The second neuron, however, is linked to

neurons 8 through 24, creating an overlapping pattern that ensures continuity and connectivity across adjacent input regions. This overlap continues across the matrix, with subsequent neurons connected to incrementally shifted groups of neurons, as illustrated by the distinct colored blocks in the matrix. The diagonal arrangement of these blocks within the matrix underscores the sequential and overlapping nature of the interleaving process, allowing the network to capture subtle shifts and relationships in the input data that span across neighboring neurons. This method ensures that the synaptic weights are interspersed throughout the matrix, facilitating the network's ability to discern complex, multi-dimensional patterns while avoiding redundancy. Consequently, the interleaving strategy not only enhances pattern recognition and generalization but also optimizes the efficiency of the network by ensuring that each neuron receives a diverse yet relevant set of inputs, thereby improving overall performance in tasks involving spatially or temporally structured data.

# 5

# Metrics and Performance Evaluation

*This chapter introduces the metrics that we used to evaluate the system as a whole and compare it with other state-of-the-art methods.*

To thoroughly evaluate the performance of the proposed neuromorphic network architecture, we employ six key metrics: Compression Ratio (CR), Signal-to-Noise and Distortion Ratio (SNDR), Root Mean Square Error (RMSE), Memory per Channel (bits/ch), Multiply and Accumulate Operations (MAC), and Memory Access. These metrics provide a comprehensive quantitative basis for assessing both the computational efficiency and the signal reconstruction accuracy of the network. It is important to note, however, that absolute power consumption and computation latency are platform-dependent, making it challenging to directly compare these aspects with previous works. Therefore, our focus remains on metrics that offer a fair and consistent evaluation across different platforms.

## 5.1. Compression Ratio (CR)

Since neuromorphic systems process data with sparse events, the Compression Ratio (CR) in our work is clearly defined by considering the difference between conventional Nyquist sampling and event-based sampling. It is defined in two stages: first, the compression ratio from the original input data (which is equivalent to being sampled by the conventional Nyquist ADC without delta modulation) to the output of the $\Delta$-ADC, and second, the compression ratio from the $\Delta$-ADC output to the final compressed data output from the SNN encoder.

**CR from $\Delta$-ADC ($CR_{\Delta ADC}$):** This is the ratio of the original data rate, represented by the bits generated based on conventional Nyquist sampling without delta modulation, to the event rate after it has been converted by the $\Delta$-ADC. This step includes the level-crossing event generation and the encoding of address event representation (AER). Even with the overhead from the AER protocol, the original data volume (assuming from Nyquist ADC) can still be effectively reduced by focusing only on the changes (i.e., delta) in the signal [16].

$$\mathrm{CR}_{\Delta\mathrm{ADC}} = \frac{\text{Input Data Rate (Nyquist ADC)}}{\text{Output Data Rate } (\Delta\text{-ADC})} \tag{5.1}$$

**CR from SNN encoder ($CR_{SNN}$):** This ratio quantifies the compression achieved by the SNN encoder itself, defined as the ratio of the event rate after the $\Delta$-ADC (which serves as the input to the SNN encoder) to the final compressed data rate produced by the SNN encoder, i.e., the autoencoder's latent space.

$$\mathrm{CR}_{\mathrm{SNN}} = \frac{\text{Input Data Rate } (\Delta\text{-ADC})}{\text{Output Data Rate (SNN)}} \tag{5.2}$$

$$\mathrm{CR} = \mathrm{CR}_{\Delta\mathrm{ADC}} \times \mathrm{CR}_{\mathrm{SNN}} \tag{5.3}$$

The overall Compression Ratio (CR) for the entire system can be expressed as the product of these two ratios as given by Equation 5.3.

## 5.2. Signal-to-Noise and Distortion Ratio (SNDR)

Signal-to-Noise and Distortion Ratio (SNDR) is a critical metric used to evaluate the accuracy of signal reconstruction within neural network architectures, particularly in the context of neuromorphic systems. SNDR quantifies how closely the reconstructed signal matches the original input signal by comparing the strength of the original signal to the magnitude of the reconstruction error. This ratio provides a robust measure of signal fidelity, indicating how well the network preserves the essential features of the input data during the compression and reconstruction processes.

The higher the SNDR value, the more accurately the network has reconstructed the original signal, with lower levels of noise and distortion. It is defined as:

$$\text{SNDR} = 20 \times \log_{10} \left( \frac{\|X\|_2}{\|X - \hat{X}\|_2} \right) \tag{5.4}$$

where:

- $X$ is the original input signal,
- $\hat{X}$ is the reconstructed output signal,
- $\|\cdot\|_2$ denotes the L2 norm, calculated as $\|X\|_2 = \sqrt{\sum_{i=1}^{n} X_i^2}$ for the original signal and $\|X - \hat{X}\|_2 = \sqrt{\sum_{i=1}^{n} \left( X_i^2 - \hat{X}_i^2 \right)^2}$ for the error signal.

In our specific implementation, SNDR measures the similarity between the cumulative delta events obtained from the input and output of the decoder, providing a meaningful indication of signal fidelity. Unlike other works where the comparison is made directly on the waveform, our approach processes delta events as inputs, making direct waveform comparison impractical. Therefore, we consider SNDR calculated from the cumulative events to be a fair and appropriate metric for evaluating the signal reconstruction accuracy in our system. This approach ensures that we maintain a consistent and reliable measure of how well our network performs in preserving the integrity of the original signal throughout the compression and reconstruction stages.

## 5.3. Root Mean Squared Error (RMSE)

Root Mean Square Error (RMSE) is a widely used metric for evaluating the accuracy of predictions made by neural networks, particularly during the training phase. RMSE quantifies the difference between the values predicted by the model and the actual values from the dataset, providing a clear measure of the model's prediction accuracy. Specifically, RMSE calculates the square root of the average squared differences between the predicted and actual values, thereby penalizing larger errors more heavily. This makes RMSE a sensitive and effective metric for identifying how well the model is learning during training. It is mathematically defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{5.5}$$

where:

- $n$ is the total number of delta-modulated events,
- $y_i$ represents the actual value at the $i$-th data point, and
- $\hat{y}_i$ represents the predicted or reconstructed value at the $i$-th data point.

In our work, RMSE was primarily utilized during the evaluation of the model in the training phase. This metric provided a fair and reliable measure of model performance as we iteratively refined the network's parameters. By focusing on RMSE during training, we were able to build a strong foundation for the model, ensuring that it learned to accurately reconstruct the input data before moving on to more advanced evaluation metrics.

However, it is important to note that while RMSE is a valuable tool during training, it is not commonly reported in final performance evaluations across different studies. Consequently, we have chosen not to include RMSE in our state-of-the-art comparison Table 6.1, as it may not provide a consistent basis for comparison with other works. Instead, we rely on other metrics more commonly used in the literature to benchmark our model's performance against the state of the art.

## 5.4. Memory per Channel (bits/ch)

Memory per Channel (bits/ch) refers to the number of bits allocated to each channel in the network. This metric is crucial for assessing the memory efficiency of the architecture, as it reflects the amount of data storage required for synapses, membrane potentials, decay parameters, and other channel-specific components to maintain the desired performance.

This metric allows for a fair comparison with other state-of-the-art methods, such as those by Biedermann[3] and Wu[55]. It provides an indication of the buffering required for specific data processing steps, such as spike alignment or detection. Additionally, normalizing the total bits per channel offers a more universal metric, enabling comparison across works with varying numbers of channels.

It is important to note that memory access significantly influences overall power consumption, while the memory footprint predominantly determines the total chip area [48]. These factors make Memory per Channel a critical consideration in the design of neuromorphic systems, especially for applications where power efficiency and chip area are of paramount importance.

## 5.5. Multiply and Accumulate Operations per Second (MAC)

In the context of Spiking Neural Networks (SNNs), the estimation of computational efficiency is crucial for understanding the network's performance, especially when compared to traditional artificial neural networks (ANNs). One key metric for evaluating this efficiency is the number of effective Multiply-Accumulate (MAC) operations. The effective MACs in an SNN can be described as the sum of the operations required for both inter-layer connections and intra-neuron (decay parameter) dynamics.

The first component of the effective MAC metric quantifies the operations required to propagate spikes between layers. In a network with $L$ layers, where the spiking rate of layer $l$ is denoted as $r_l$, the number of neurons in the $l$-th layer as $d_l$, and the number of neurons in the preceding layer as $d_{l-1}$, the inter-layer MACs are given by:

$$\text{MAC}_{\text{inter-layer}} = \sum_{l=1}^{L} (r_l \times d_{l-1} \times d_l) \tag{5.6}$$

Secondly, each neuron in an SNN typically has additional operations related to its internal dynamics, such as membrane potential decay and bias accumulation. These operations can be quantified as:

$$\text{MAC}_{\text{intra-neuron}} = \sum_{l=1}^{L} (\text{decay} + \text{bias}_l) \times d_l \tag{5.7}$$

In this context, decay represents the computational cost of maintaining the membrane potential dynamics, and $\text{bias}_l$ represents the accumulation of bias terms in the neurons of layer $l$. Therefore, the total

effective MAC operations for the SNN can be expressed as:

$$\text{Total MACs} = \text{MACs}_{\text{inter-layer}} + \text{MACs}_{\text{intra-neuron}} \tag{5.8}$$

This metric, inspired by [29], provides a comprehensive measure of the computational cost associated with the network, taking into account the sparsity of spikes, the presence of recurrent connections, and the intrinsic dynamics of spiking neurons. However, as we utilize a spiking neural network in our work, a more accurate term would be ”Accumulate” (ACC) operations instead of Multiply-Accumulate (MAC) operations, reflecting the event-driven nature of our SNN. In general state-of-the-art comparisons, the metric remains a MAC, but in the context of our specific implementation, it is better referred to as ACC. This ensures that the metric remains general enough to benchmark against other state-of-the-art methods while accurately representing the computational efficiency in our neuromorphic system.

## 5.6. Memory Access

Memory Access (MA) is a critical metric that quantifies the rate at which memory is accessed during the operation of the network. This metric is essential for a comprehensive estimation of the network’s energy consumption, as memory read-and-write operations typically require one to two orders of magnitude more energy than arithmetic operations performed by linear units [49], [19].

Given that most of the energy consumption in neural architectures stems from these memory operations, simply reporting the effective arithmetic operations without including memory access would provide an incomplete assessment of the network’s efficiency. For instance, it has been reported in [29] that during inference, there can be approximately 10 times more memory reads than effective operations, emphasizing the significant energy impact of memory access.

In line with previous studies [29], we conservatively estimate the number of memory accesses by assuming that each MAC operation involves three memory loads and one store (this includes the memory read for bias since we don’t use one in our student network), and each Accumulate (ACC) operation involves two memory loads and one store (since in our network we don’t have a bias). This estimate is further adjusted to account for the sparseness of network activity, providing a realistic measure of memory access in the system. The total number of memory accesses can be expressed as:

$$\text{MA} = (3 \text{ Loads} + 1 \text{ Store}) \times \text{MACs} + (2 \text{ Loads} + 1 \text{ Store}) \times \text{ACCs} \tag{5.9}$$

This equation ensures that the evaluation of the network’s energy efficiency is both accurate and comprehensive by accounting for the energy impact of memory operations relative to computational operations. In our work, we exclusively utilize ACC operations, resulting in a total of three memory operations. In contrast, other state-of-the-art methods that rely on MAC operations require four memory operations in total. This difference highlights the inherent efficiency of our approach in minimizing memory access, which is a significant contributor to overall energy consumption.

# 6

# Results and Experiments

*This chapter explores the key results obtained from benchmarking our system against defined metrics, as well as the findings from subsequent ablation studies.*

In this section, we present the main results of our Autoencoder System, followed by detailed ablation studies that explore the influence of different teacher networks, network sizes, and configurations. These experiments were conducted using multi-electrode recordings from two datasets, each comprising 30 minutes of data from different mice. The first dataset, containing hippocampal recordings, was obtained from University College London (UCL) [2], and the second, consisting of cerebellar recordings, was sourced from Erasmus Medical Center [41]. To ensure robustness and generalizability across various conditions and subjects, we refer to these datasets as *hippo_dat1* and *cere_dat2*, respectively.

## 6.1. Performance Summary and State-of-the-Art (SoTA) Comparison

We compared various state-of-the-art compression techniques, as outlined in Section 3, against the metrics defined in Section 5 to obtain a comprehensive assessment of our system's efficiency.

| **Reference** | He [16] | Biederman [3] | Zhang [58] | Wu [55] | **This Work** |
|---|---|---|---|---|---|
| **Publication** | TBioCAS'24 | JSSC'15 | IOP'15 | IOP'18 | - |
| **No. of channels** | 128 | 64 | 4 | 256 | 128 |
| **Encoder Type** | Delta mod. | Epoch | Compresssed Sensing | CNN | Delta mod. + SNN |
| **Compression Ratio** | 10 × | 8 × | 8 × - 16 × | 20 × | **94 ×** |
| **Peak SNDR** | 36 dB | >50 dB | ~ 13 dB | 10 – 13 dB | **22 dB** |
| **Memory (bit /ch)** | 27 | 1500 | - | 576[A] | **384** |
| **Computations (MACs/spike)** | - | - | - | 79.25K | 18.07K |
| | | | | 309[B] | **141[B]** |
| **Memory Access per spike** | - | - | - | 237.75K | 54.21K |
| | | | | 1238[B] | **423[B]** |

[A]This number excludes the additional data buffer required for spike detection and alignment, which requires extra ~1000b/channel of memory for 3ms of spike data.
[B] This number has been normalized to number of channel.

**Table 6.1:** Summarized performace and comparsion to STATE-OF-THE-ART (SOTA) neural compression systems.

As shown in Table. 6.1, the proposed compression system achieves superior compression, 4.7× better

than the closest existing work [55], while demonstrating 9-12 dB better reconstruction quality in SNDR. This significant improvement is attributed to the combination of the delta modulation, the event-based neuromorphic encoder, and the proposed two-stage teacher-student network training. It also requires only approximately 5× less memory usage compared to [3], which is attributed to buffer-less real-time processing of neuromorphic encoders and meticulous pruning techniques implemented in this work.

Additionally, using the neuromorphic encoder enabled us to minimize multiplication operations, reducing the effective MAC operations by nearly 4× compared to the closest neural network-based compression systems. This reduction also led to a corresponding decrease in the number of memory read and write operations by the same factor, further enhancing the overall efficiency of the proposed compression system. Fig. 6.1 illustrates a detailed visual comparison between the cumulative waveforms of an original input signal from the Δ-ADC and its corresponding reconstructed signal our Neuromorphic Autoencoder for the *hippo_dat1* and *cere_dat2* datasets. The original waveform (blue) and the reconstructed waveform (orange) closely align, indicating the decoder's high fidelity in preserving the overall signal trend.

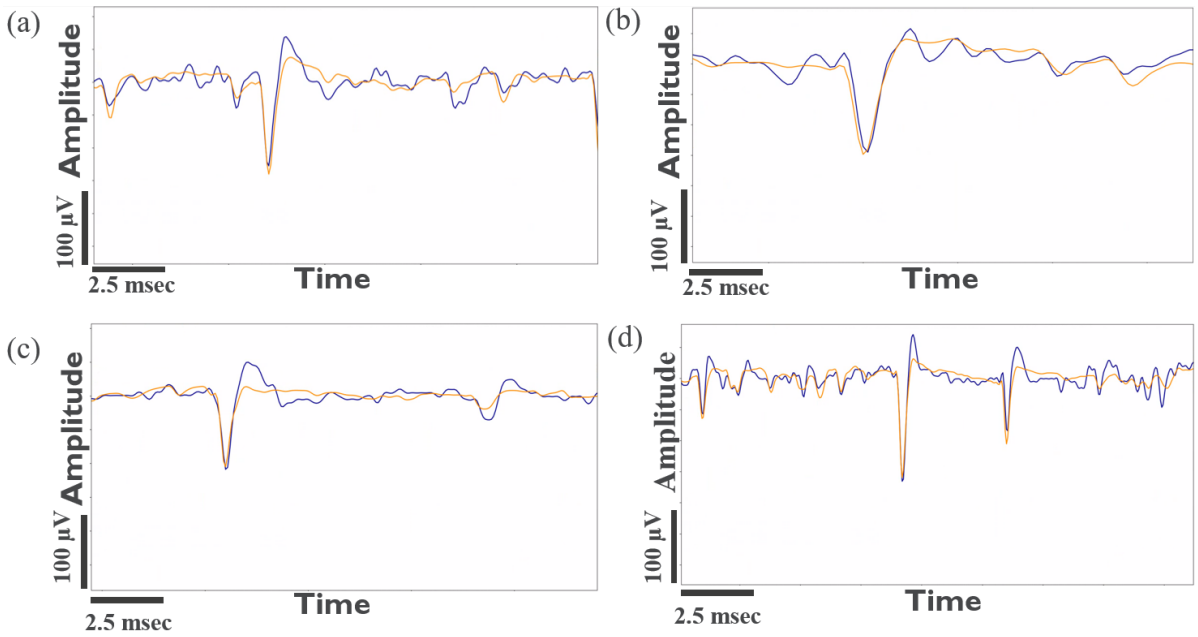



**Figure 6.1:** Comparison of original (blue) and reconstructed waveforms (orange) for different spikes in the hippocampus dataset (hippo_dat1) ((a) & (b)) and cerebellum dataset (cere_dat2) ((c) & (d)) using our neuromorphic autoencoder compression system.

It is important to note that the figures presented in Table. 6.1 were derived from the *hippo_dat1 dataset* [2], which is publicly available and can be reproduced if needed, ensuring the robustness and reproducibility of the presented results.

## 6.2. Influence of Teacher Network Architectures on Performance

To optimize the training of our spiking-based student network, we conducted an ablation study on various teacher network architectures using the *hippo_dat1* dataset. We evaluated Spiking Neural Networks (SNN), Convolutional Neural Networks (CNN), Spiking Convolutional Neural Networks (SCNN), Long Short-Term Memory networks (LSTM), Spiking Long Short-Term Memory networks (SLSTM), Recurrent Neural Networks (RNN), and Spiking Recurrent Neural Networks (SRNN).

As illustrated in Fig. 6.2, other network architectures outperformed the SNN, evidenced by lower Root Mean Squared Error (RMSE) loss during training. This observation underscores the critical importance

of knowledge distillation in effectively training spiking-based student networks.

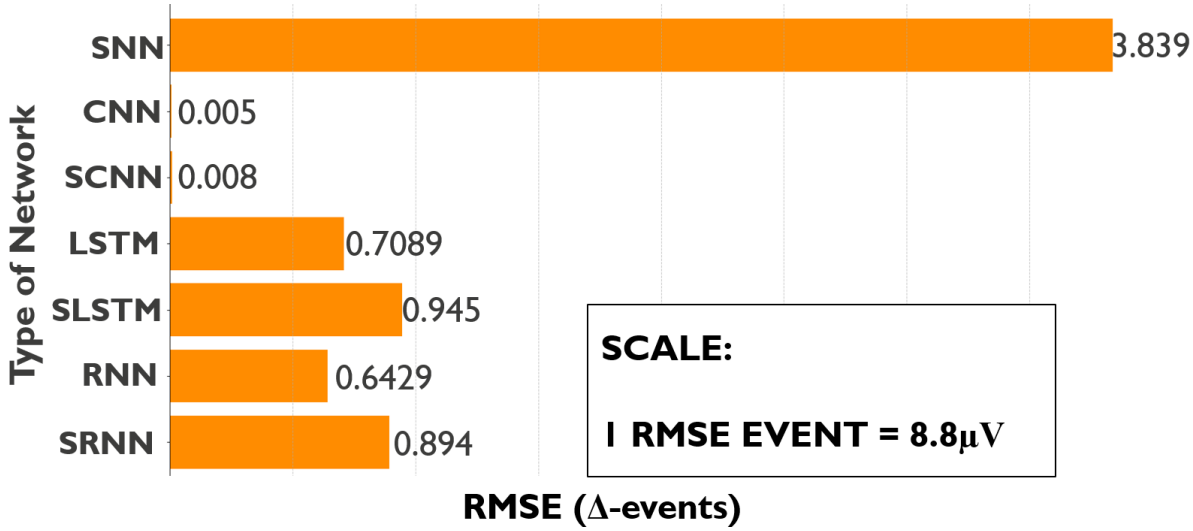



**Figure 6.2:** Comparative performance of teacher architectures

Our findings show that non-spiking architectures generally outperform spiking ones, with CNNs achieving the best performance in terms of training loss. However, when training the student network using the two-stage encoder-decoder method described earlier in Section 4.3.1, RNN and LSTM models more effectively preserved waveform fidelity, despite the CNN's favorable compression ratios (as illustrated in Fig. 6.3). The spiking variants performed significantly worse than the architectures shown in Fig. 6.2, primarily due to inherent limitations such as lower frequency clocks, which may not fully capture the temporal dynamics encoded in spikes[43].

These results indicate that LSTM and RNN architectures are more effective as teacher models for training spiking neural networks, given their ability to handle temporal sequences that align better with the spiking network's requirements. For the remainder of this paper, we work with RNN as the teacher network.

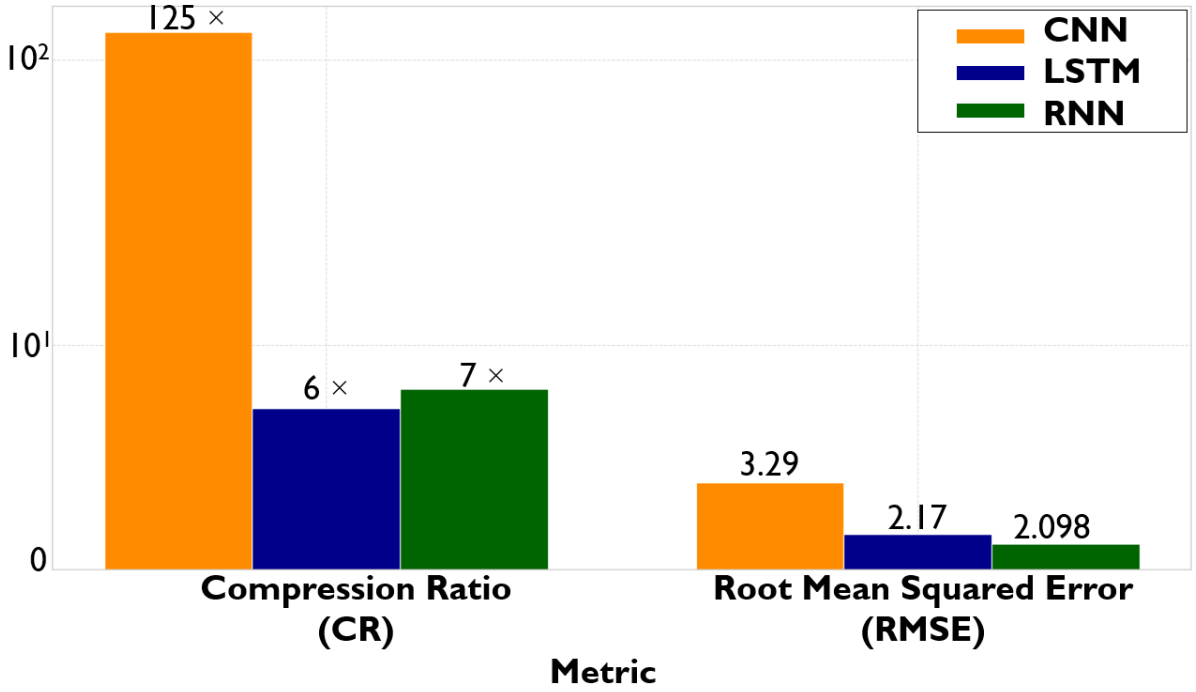



**Figure 6.3:** Comparison of student network compression ratios and accuracy

## 6.3. Impact of Teacher Network Depth

We conducted an ablation study to determine the optimal number of layers for the RNN-based teacher network, aiming to balance model complexity and accuracy. As illustrated in Fig. 6.4, we evaluated various layer configurations across the *hippo_dat1* and *cere_dat2* datasets.

The results reveal a clear trend: increasing the number of layers beyond three leads to a significant increase in Root Mean Square Error (RMSE) for both datasets. This observation suggests that adding more layers increases the risk of overfitting, where the model becomes too tailored to the training data and loses its ability to generalize effectively to new data. Specifically, the RMSE sharply rises with five or more layers, indicating that excessive complexity hampers the network's performance rather than enhancing it, consistent with findings in related studies [15].

Conversely, a configuration with three or four layers consistently delivers robust performance with minimal RMSE, striking the optimal balance between complexity and accuracy. These configurations provide sufficient depth to capture the underlying patterns effectively without the drawbacks of overfitting.

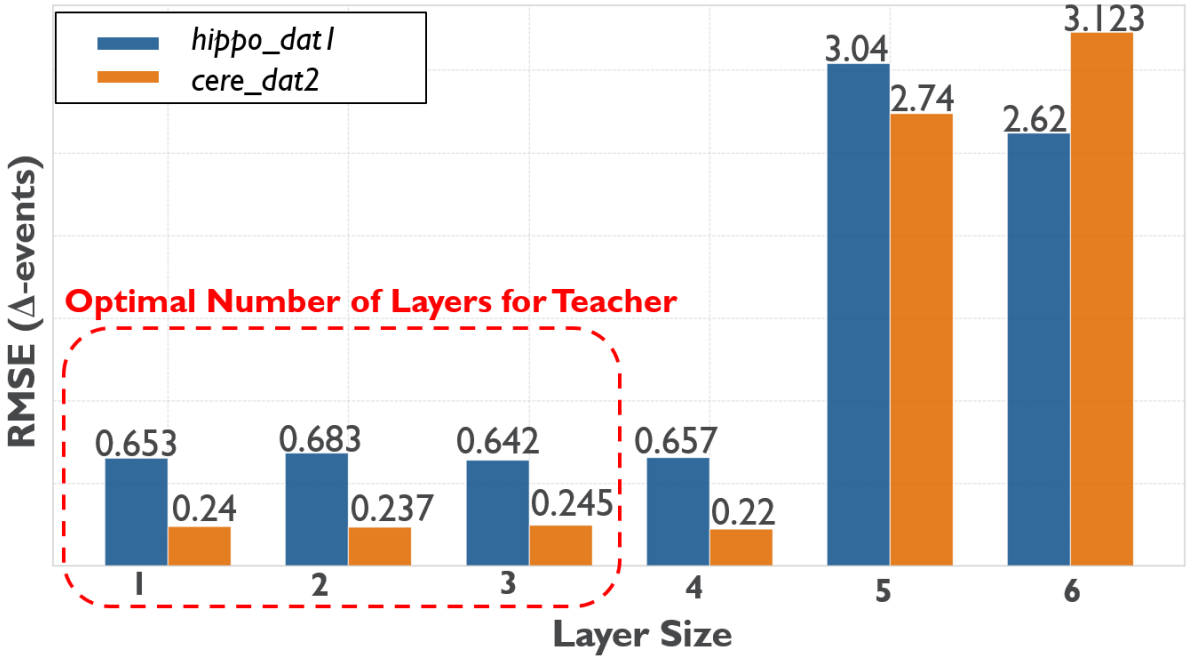



**Figure 6.4:** Comparison of teacher RNN network with different layer sizes.

However, reducing the layer count below three may introduce the risk of the spiking student model's ability to generalize. As shown in the lower RMSE values for the two-layer configuration compared to a single layer, the model's capacity to capture complex temporal dynamics improves with additional layers, which is crucial for effectively training recurrent models[18]. This suggests that a three- or four-layer teacher network offers a solid foundation for training, effectively balancing generalization and performance. While these findings provide valuable guidance, further analysis is necessary to conclusively determine the impact of reducing layers on the student network's performance as will be elaborated in subsequent sections.

## 6.4. Optimization of Latent Space Dimensionality

To further enhance the performance of our Recurrent Neural Networks (RNNs) as a teacher model, we conducted an analysis of the impact of different latent space dimensions on the network's ability to capture and generalize from the underlying data. This step is crucial as it informs the selection of an appropriate latent space dimension, which directly affects the model's capacity to represent complex patterns within the datasets.

To ensure consistent and accurate loss calculation during training the student, particularly when computing the latent loss during training (as explained in Section 4.3.1), an additional layer of 64 units (off-chip layer as shown in Fig. 4.6) is included as the final layer. However, this last layer is not considered part of the spiking layers; instead, it functions as an RNN cell. This design choice is critical for maintaining a seamless alignment between the latent representations of the student and teacher networks, thereby facilitating effective knowledge transfer through knowledge distillation.

The initial spiking layers, with their 128 and 64 units respectively (as shown in Fig. 6.5), provide sufficient depth and capacity to encode complex features while maintaining computational efficiency. The inclusion of the RNN cell at the end serves to refine the output, ensuring that the latent space is accurately captured and utilized in the loss calculation. This structure not only enhances the model's performance but also ensures that the training process remains robust and aligned with the objectives of efficient and accurate data compression within the autoencoder framework.

As illustrated in Fig. 13, we evaluated the RMSE performance across various latent space dimensions—specifically, 8, 16, 32, 64, and 128—using both the *hippo_dat1* and *cere_dat2* datasets. The results demonstrate a clear downward trend in RMSE as the latent dimension increases, indicating that a larger latent space allows the model to better encode the intricate relationships present in the data.

For the *hippo_dat1* Dataset, the RMSE decreases sharply from around 3.0 with an 8-dimensional latent space to 0.65 with a 64-dimensional latent space. Similarly, the *cere_dat2* Dataset exhibits a reduction in RMSE from 2.45 to around 0.22 across the same range. This significant improvement underscores the importance of a higher-dimensional latent space in providing the necessary capacity for the RNN to accurately model the complexities of the datasets.

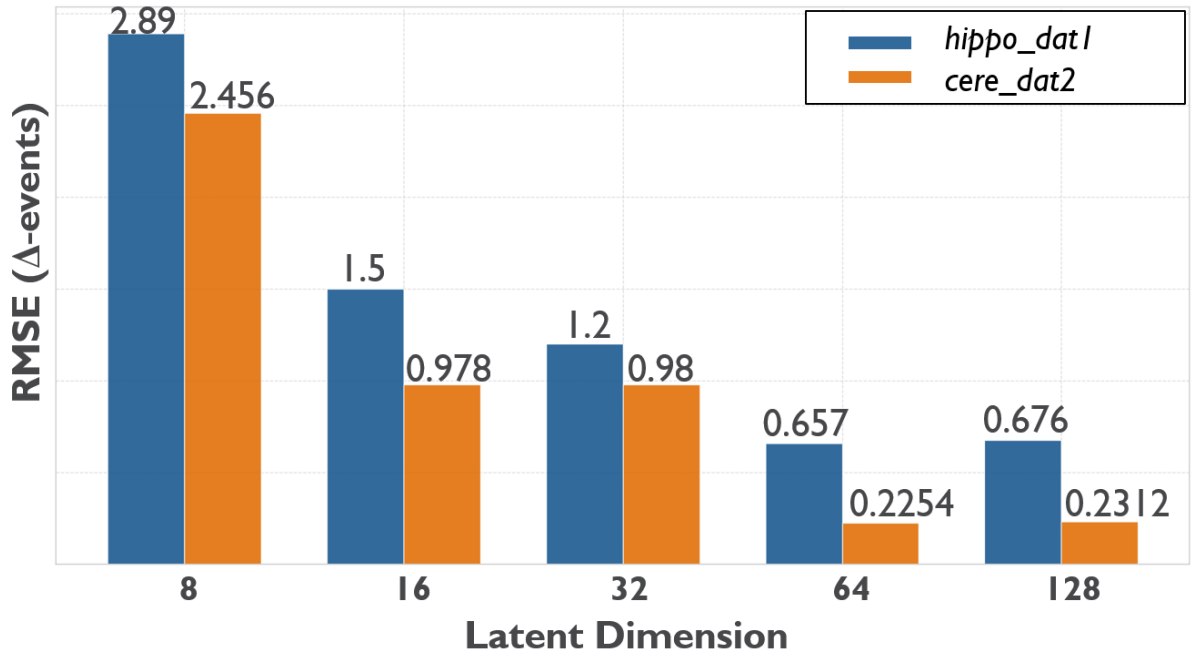



**Figure 6.5:** Comparison of teacher RNN network with different latent dimensions

Notably, while the RMSE values for both datasets converge at the higher latent dimensions, the *cere_dat2* dataset consistently achieves lower RMSE values, particularly at the larger latent space sizes. This difference suggests that the *cere_dat2* dataset may exhibit more straightforward patterns, which can be efficiently captured even as model complexity increases. In contrast, the *hippo_dat1* dataset appears to benefit more significantly from an expanded latent space, likely due to its more complex signal characteristics.

Moreover, the superior performance of the *cere_dat2* dataset compared to the *hippo_dat1* dataset at this stage might be attributed to differences in data collection methods. Recordings from the cerebellum are characterized by a higher firing rate compared to those from the hippocampus, providing more frequent data points from the cerebellum [36]. Additionally, hippocampal recordings typically have larger signal amplitudes, as the electrodes are placed deeper within the brain. This depth results in more significant RMSE errors due to inherent amplitude reductions in the reconstructed waveforms, as indicated in Fig. 6.5 [8].

Based on these findings, a 64-dimensional latent space is justified as the optimal choice for the teacher network within the context of our study. This dimension provides a robust capacity for the model to learn and generalize effectively across both datasets, minimizing RMSE and thereby improving the accuracy of the model's predictions. However, it is worth noting that while a 64-dimensional latent space offers

the best performance in this scenario, it also introduces additional computational complexity.

## 6.5. Evaluation of Student Network Depth

From the insights from our initial analysis, we further examined the impact of varying the number of layers (hidden layers) in the student network, focusing specifically on configurations ranging from one to three layers. As shown in Fig. 6.6, the RMSE values decrease progressively as the number of layers increases from one to three across both the *hippo_dat1* and *cere_dat2* datasets for the student spiking network.

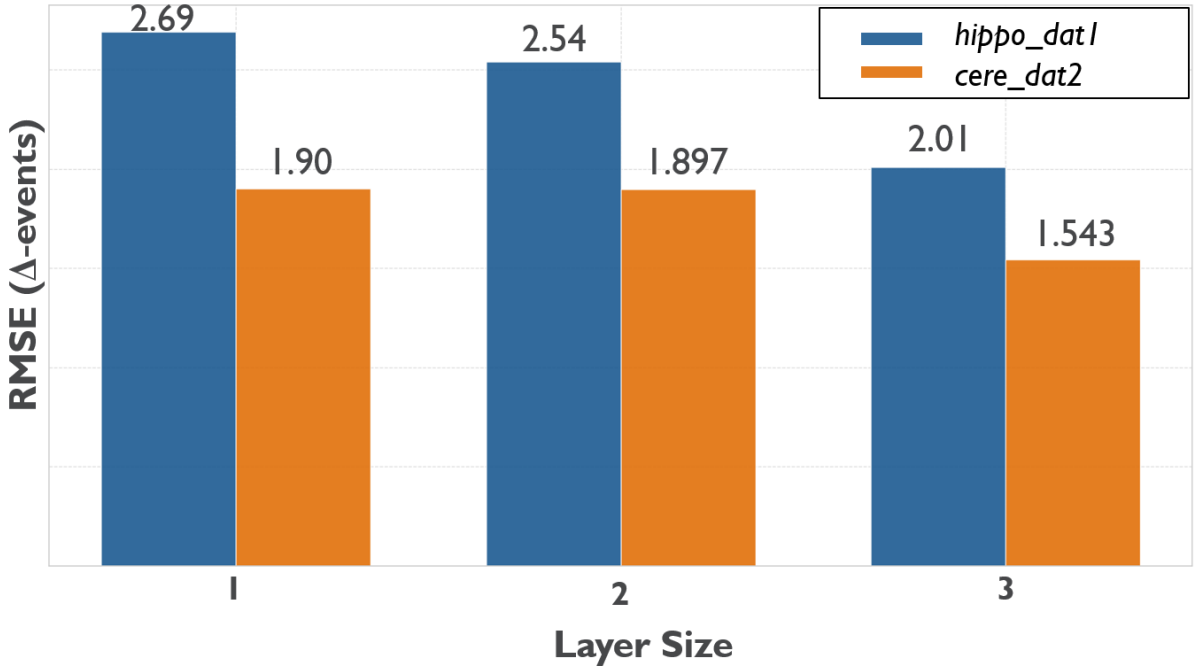



**Figure 6.6:** Comparison of student spiking network with different layer sizes

For the *hippo_dat1* Dataset, the RMSE decreases from 2.69 with a single layer to 2.01 with three layers, while for the *cere_dat2* Dataset, the RMSE reduces from 1.90 to 1.543 over the same range. This reduction in RMSE highlights the importance of deeper spiking network architectures, at least within this tested range, in effectively capturing the intricate patterns present in the data.

Notably, while both datasets exhibit a similar downward trend, the RMSE values for the *cere_dat2* Dataset consistently remain lower than those for the *hippo_dat1* Dataset across all tested layer sizes. This difference likely reflects the varying characteristics and complexities inherent in the two datasets. The *cere_dat2* Dataset appears to be ”cleaner,” with patterns that are more readily captured by the RNN, even with fewer layers, whereas the *hippo_dat1* Dataset may require a more complex model to achieve comparable accuracy.

However, it is not the only viable option; depending on specific application constraints, a one- or two-layer structure might be preferred, trading off a slight increase in RMSE for reduced model complexity and lower computational overhead. While a three-layer (hidden) structure appears to deliver fair reconstruction performance, the choice of network depth should ultimately align with the broader goals and limitations of the particular application.

## 6.6. Pruned Student Model Configuration

Following extensive experimentation, we identified a three-hidden-layer student spiking neural network with a 64-dimensional latent space as the most suitable configuration. This model, detailed in Table

6.2, balances memory efficiency and accuracy effectively.

As discussed in Section 4.3.2, we employed input grouping and interleaving techniques to reduce hardware complexity, gradually pruning the network layer by layer, starting from the fully connected configuration as shown in Table 6.2.

| Connection Type | | | Network No. |
|---|---|---|---|
| **Input Layer** | **2nd Hidden Layer** | **3rd Hidden Layer** | |
| Fully Connected | Fully Connected | Fully Connected | 1 |
| Grouped | Fully Connected | Fully Connected | 2 |
| Grouped | Interleaved | Fully Connected | 3 |
| Grouped | Interleaved | Interleaved | 4 |

**Table 6.2:** Types of student network connection explored for hardware aware optimization

This approach resulted in a nearly 7× reduction in memory footprint per channel from the fully connected model to the final pruned model, as seen in Table 6.3. However, this reduction came with a trade-off in RMSE values, which gradually increased due to information loss during pruning. A similar effect was observed in the decreasing SNDR values.

| Data | Network No. | Memory (kbit/ch) | $CR_{SNN}$ | RMSE | SNDR (dB) |
|---|---|---|---|---|---|
| *hippo_data1* | 1 | 2.55 | 5.48 | 2.01 | 24.47 |
| | 2 | 1.64 | 7.63 | 2.137 | 23.94 |
| | 3 | 0.384 | 9.4 | 2.185 | 23.75 |
| | 4 | 0.24 | 19.7 | 6.217 | 10.67 |
| *cere_dat2* | 1 | 2.55 | 6.12 | 1.543 | 24.19 |
| | 2 | 1.64 | 7.98 | 1.78 | 22.95 |
| | 3 | 0.384 | 10.2 | 1.83 | 22.7 |
| | 4 | 0.24 | 12.9 | 5.667 | 10.94 |

**Table 6.3:** Hardware aware student analysis - bits used vs compression ratio vs reconstruction fidelity on different datasets.

To strike the best balance between memory efficiency and accuracy, we identified Network No. 3 as the optimal model. In this configuration, pruning was applied only up to the penultimate spiking layer, while the final spiking layer remained fully connected. This approach minimized the trade-off between memory usage and accuracy.

## 6.7. Quantization

In the previous section, we identified Network No. 3 as the optimal model due to its balanced trade-off between memory efficiency and accuracy. Building on this, we proceeded to further optimize this model through quantization. In the process of optimizing our neuromorphic encoder model, we employed symmetric quantization to reduce the precision of weights and activations from floating-point 32 (FP32) to integer formats (INT8, INT6, and INT4).

Symmetric quantization was chosen for its ability to maintain a balanced range of positive and negative values around zero, which is particularly effective for the types of data encountered in our model [60]. This approach led to a similar compression ratio as in FP32, as highlighted in Fig. 6.7. However, as depicted in Fig. 6.8, a reduction in precision increased RMSE, particularly at the INT4 level, indicating a trade-off between compression and signal reconstruction accuracy.

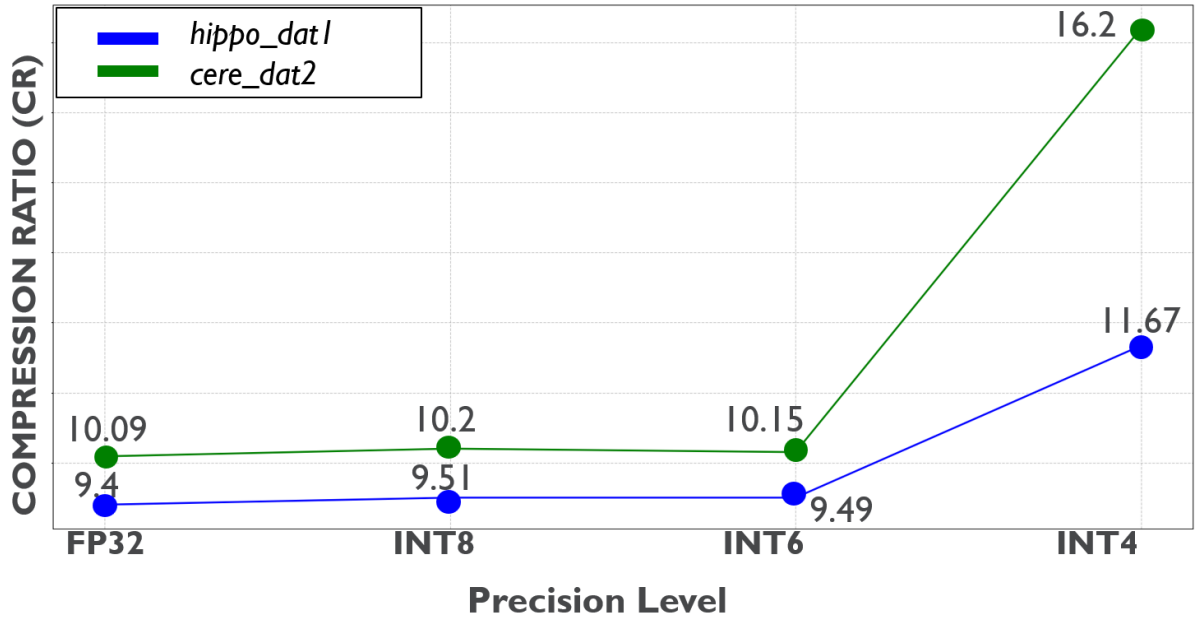



**Figure 6.7:** Compression ratio (CR) across different precision levels for the hippo_dat1 and cere_dat2 datasets.

To further enhance the efficiency of our quantization process, we integrated a technique known as BatchNorm folding [60]. BatchNorm layers, which are typically used during training to stabilize and accelerate convergence, were folded into the preceding convolutional or fully connected layers during the inference phase. This folding process involves incorporating the BatchNorm parameters (mean and variance) into the weights and biases of the corresponding layers, effectively reducing the computational overhead and memory requirements during inference. By doing so, we were able to maintain the benefits of BatchNorm during training while simplifying the model for deployment on resource-constrained hardware.

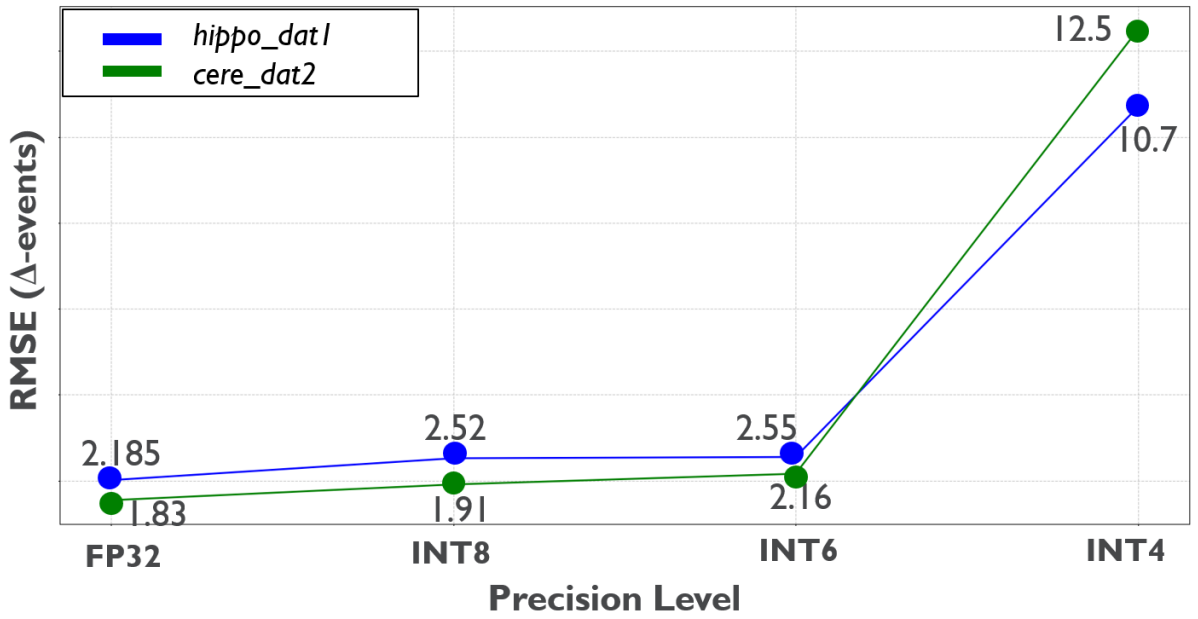



**Figure 6.8:** RMSE across different precision levels for the hippo_dat1 and cere_dat2 datasets.

This combination of symmetric quantization and BatchNorm folding not only improved the overall memory efficiency of our model giving almost a 1.5 × fold reduction in memory (bit/ch) but also ensured that

it remained compact and suitable for real-time applications in neuromorphic systems [32].

# 7

# Discussions and Conclusion

*This penultimate chapter highlights key findings that warrant further discussion and provides a summary of all our results.*

## 7.1. Discussions

Our system addresses several key bottlenecks encountered by existing state-of-the-art methods. Zhang's et.al approach [59] faces challenges with the complexity of optimizing signal reconstruction, as evidenced by its low SNDR value, which is nearly 2.5 times lower than that achieved by our method. Furthermore, Zhang's method requires additional buffer memory to store the sampling and transformation matrices, which complicates its implementation. Similarly, the methods proposed by Biedermann et.al [3] and Wu et.al [55] also necessitate extra buffer memory, as they rely on spike detection and alignment, requiring nearly 1500b per channel of memory for 3 ms of spike data. Wu's study is the closest to our work, given its use of neural networks; however, it is burdened by the large number of parameters inherent in CNNs. Our approach, which employs an event-based neuromorphic design, significantly reduces memory and computation costs by nearly twofold, while also achieving a signal fidelity that is approximately 10 dB higher than Wu's work.

However, the Fig. 6.1(d) we can observe minor amplitude shrinkages in the reconstructed waveform, which may occur due to the use of a lower frequency clock, potentially failing to fully capture the temporal dynamics encoded in the spikes[43]. These discrepancies are minimal and occur at certain critical points, as shown in the boxed areas. Despite these small variations, which are further quantified through the RMSE analysis earlier, the decoder effectively replicates the signal with high accuracy. The identification and visualization of these discrepancies underscore the robust performance of the decoder while also acknowledging the areas where slight deviations occur, thereby providing a comprehensive evaluation of the signal reconstruction process.

Despite these advancements, there remain opportunities for further research. Future studies could explore the real-time implementation of our system and assess its performance in dynamic environments. Additionally, applying this compression architecture to other domains within neuromorphic computing may yield further insights and enhancements.

The neuromorphic compression system presented in this paper establishes a new benchmark for neural signal processing in intra-cortical Brain-Computer Interfaces (iBCIs). It offers a scalable and energy-efficient solution that not only meets the current demands of high-density neural data processing but also paves the way for future advancements in neural prosthetics and neuro-engineering. The findings and methods developed here represent a critical step toward the practical realization of advanced Brain-Computer Interface technologies, making a substantial contribution to the field.

## 7.2. Conclusion

This paper presents a significant advancement in neuromorphic systems, particularly for intra-cortical Brain-Computer Interfaces (iBCIs), by introducing a novel compression architecture that integrates delta modulation with Spiking Neural Networks (SNNs). The proposed system achieves a noteworthy compression ratio of$> 94\times$, far surpassing existing state-of-the-art methods, and effectively addresses the critical challenges of data transmission and storage in high-density neural recording environments.

Our experimental results, validated on hippocampal and cerebellar datasets [2], [41], demonstrate the superiority of our approach in terms of compression efficiency, Signal-to-Noise and Distortion Ratio (SNDR), and memory usage compared to existing state-of-the-art techniques[59][3][55]. Notably, our system not only achieves a peak SNDR of 24 dB but also significantly reduces memory usage to 384 bits per channel, with computations per spike minimized to 18.07K MACs and memory access per spike reduced to 54.21K—highlighting its efficiency for resource-constrained implantable devices.

The ability to drastically reduce data throughput while preserving the essential characteristics of neural signals underscores the potential of SNNs in creating highly efficient and scalable neuromorphic systems. Moreover, the low power consumption and minimal heat generation of the system make it particularly well-suited for implantable devices, where resource constraints are a primary concern.

The effectiveness of this work lies in its innovative application of neuromorphic principles, leveraging the event-driven nature of SNNs to achieve both high compression and energy efficiency. Moreover, the training method used hardware-aware optimization techniques, including pruning and quantization, to further enhance the system's applicability in real-world, resource-limited settings. These techniques enabled the reduction of synaptic weights and the associated memory requirements, contributing to the overall compactness and efficiency of the system.

# 8

# Future Scope

*This final chapter outlines potential future directions for this thesis, briefly mentioning key findings and exploring opportunities for further research based on these results.*

Although our system demonstrates superior computational and storage efficiency compared to other state-of-the-art methods (as shown in Table 6.1), there are still areas that can be further explored and refined to enhance its capabilities:

**1. Reducing the Number of Student Layers:** This involves optimizing the layers within the student network to improve computational efficiency while maintaining performance. Initial investigations have shown that a single-layer student SNN can achieve the same compression ratio as a three-layer structure, though this comes at the expense of some reconstruction quality. Future work could focus on fine-tuning this trade-off, perhaps by exploring hybrid architectures or by incorporating adaptive layer structures that can dynamically adjust based on input complexity or desired output fidelity.

**2. Asymmetrical Autoencoder:** An asymmetrical autoencoder structure presents an interesting avenue for improving the current architecture. This involves using a highly capable student decoder, which may not necessarily be an SNN. Preliminary experiments with a powerful RNN-based student architecture have yielded promising results. However, further validation is necessary to establish its effectiveness and generalizability across different datasets and tasks. Future research could also explore the integration of other deep learning models, such as transformers or generative networks, to create a more versatile and powerful decoding mechanism.

**3. Optimizing SNN Hardware Implementation:** The hardware implementation of Spiking Neural Networks (SNNs) offers significant potential for efficiency gains, particularly in neuromorphic computing systems where power consumption and thermal management are critical. One promising direction explored during this research was the replacement of the decay operation in SNNs with a shifter, which eliminates the need for a multiplier block, thus reducing hardware complexity and power usage. Future work could extend this approach by exploring other hardware-friendly optimizations, such as using approximate computing techniques or custom digital circuits tailored to the specific characteristics of SNNs. More detailed discussions on these implementations can be found in Appendix 8.

**4. Exploring Spike-Timing-Dependent Plasticity (STDP):** STDP is a biologically inspired learning mechanism that could further enhance the parallelism within SNNs by allowing the FIRE logic to execute concurrently with the Update phase. This would not only improve processing efficiency but could also lead to more biologically plausible models. Future research could focus on developing more advanced STDP algorithms that can be effectively integrated into large-scale SNNs, potentially leading to breakthroughs in both performance and real-time processing capabilities.

**5. Advanced Quantization Schemes:** Sophisticated quantization techniques, such as Adaround [38], offer opportunities to further reduce memory requirements and computational costs, making the system more suitable for resource-constrained environments. In this thesis, the quantization was limited

to INT6 to balance reconstruction quality with compression. However, exploring more advanced quantization methods could allow for even lower bit-widths, such as INT4, while maintaining or improving precision. This could be particularly valuable in deploying SNNs on edge devices or other platforms with stringent memory and power constraints.

**6. Integration with Emerging Technologies:** The rapid development of technologies such as quantum computing, photonic processors, and memristive devices opens new possibilities for SNNs. Future research could investigate how these emerging technologies can be leveraged to enhance the performance and capabilities of SNNs. For instance, quantum computing could be used to optimize learning algorithms or accelerate inference, while photonic processors could offer ultra-fast and energy-efficient computation. Exploring these integrations could position SNNs at the forefront of next-generation AI systems.

These directions provide a comprehensive roadmap for future research, aiming to enhance the current system's efficiency, scalability, and applicability in practical scenarios. By addressing these challenges and opportunities, future work could significantly advance the field, bringing us closer to realizing the full potential of SNNs in both academic and industrial applications.

# References


[1] G.N. Angotzi et al. “SiNAPS: an implantable active pixel sensor CMOS-probe for simultaneous large-scale neural recordings”. In: *Biosensors and Bioelectronics* 126 (2019), pp. 355–364. DOI: 10.1016/j.bios.2018.10.027.

[2] Anonymous. *Recording with a Neuropixels probe*. University College London, Feb. 2024. DOI: 10.5522/04/25232962.v2.

[3] W. Biederman et al. “A 4.78 mm 2 Fully-Integrated Neuromodulation SoC Combining 64 Acquisition Channels With Digital Compression and Simultaneous Dual Stimulation”. In: *IEEE Journal of Solid-State Circuits* 50.4 (2015), pp. 1038–1047. DOI: 10.1109/JSSC.2014.2384736.

[4] Blackrock Microsystems. *Cerebus: Instructions for Use*. https://www.blackrockmicro.com/wp-content/ifu/LB-0028-15.00-Cerebus-Instructions-for-Use.pdf. 2020.

[5] M. Bockbrader. “Brain Computer Interfaces in Rehabilitation Medicine”. In: *PM&R* (2018). Accessed: Aug. 07, 2024. URL: https://onlinelibrary.wiley.com/doi/full/10.1016/j.pmrj.2018.05.028.

[6] D.A. Borton et al. “An implantable wireless neural interface for recording cortical circuit dynamics in moving primates”. In: *Journal of Neural Engineering* 10.2 (2013), p. 026010. DOI: 10.1088/1741-2560/10/2/026010.

[7] G. Buzsáki. *Rhythms of the brain*. Oxford University Press, 2006. DOI: 10.1093/acprof:oso/9780195301069.001.0001.

[8] György Buzsáki. “Large-scale recording of neuronal ensembles”. In: *Nature Neuroscience* 7.5 (2004), pp. 446–451. DOI: 10.1038/nn1233.

[9] X. Chen et al. “Brain-machine interfaces: from basics to clinical translation and beyond”. In: *Neuroscience Bulletin* 35.6 (2019), pp. 1001–1016.

[10] M. R. Cohen and A. Kohn. “Measuring and interpreting neuronal correlations”. In: *Nature Neuroscience* 14.7 (2011), pp. 811–819. DOI: 10.1038/nn.2842.

[11] M. Davies et al. “Loihi: A neuromorphic manycore processor with on-chip learning”. In: *IEEE Micro* 38.1 (2018), pp. 82–99. DOI: 10.1109/MM.2018.112130359.

[12] D. De Dorigo et al. “Fully immersible subcortical neural probes with modular architecture and a Delta-Sigma ADC integrated under each electrode for parallel readout of 144 recording sites”. In: *IEEE Journal of Solid-State Circuits* 53.11 (2018), pp. 3111–3125. DOI: 10.1109/JSSC.2018.2845694.

[13] N. Even-Chen et al. “Power-saving design opportunities for wireless intracortical brain–computer interfaces”. In: *Nature Biomedical Engineering* 4.10 (2020), pp. 984–996. DOI: 10.1038/s41551-020-0595-9.

[14] H. Gao et al. “HermesE: a 96-channel full data rate direct neural interface in 0.13 µm CMOS”. In: *IEEE Journal of Solid-State Circuits* 47.4 (2012), pp. 1043–1055. DOI: 10.1109/JSSC.2012.2184972.

[15] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[16] Y. He et al. “An event-based neural compressive telemetry with >11× loss-less data reduction for high-bandwidth intracortical brain computer interfaces”. In: *IEEE Transactions on Biomedical Circuits and Systems* (Mar. 2024). DOI: 10.1109/TBCAS.2024.3378973.

[17] G. Hinton, O. Vinyals, and J. Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (Mar. 2015). DOI: 10.48550/arXiv.1503.02531.

[18] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[19] M. 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)*. IEEE. Feb. 2014, pp. 10–14. DOI: 10.1109/ISSCC.2014.6757323.

[20] H. Hosseini-Nejad, A. Jannesari, and A. M. Sodagar. “Data compression in brain-machine/computer interfaces based on the Walsh–Hadamard transform”. In: *IEEE Transactions on Biomedical Circuits and Systems* 8 (2014), pp. 129–137. DOI: 10.1109/TBCAS.2013.2258669.

[21] *How do neurons work?* Accessed: Aug. 08, 2024. 2024. URL: https://qbi.uq.edu.au/brain-basics/brain/brain-physiology/how-do-neurons-work.

[22] Paul Hueber et al. “Benchmarking of hardware-efficient real-time neural decoding in brain–computer interfaces”. In: *Neuromorphic Computing and Engineering* 4 (May 2024). DOI: 10.1088/2634-4386/ad4411.

[23] G. Indiveri and S.-C. Liu. “Memory and information processing in neuromorphic systems”. In: *Proceedings of the IEEE* 103.8 (2015), pp. 1379–1397. DOI: 10.1109/JPROC.2015.2444094.

[24] J.J. Jun et al. “Fully integrated silicon probes for high-density recording of neural activity”. In: *Nature* 551.7679 (2017), pp. 232–236. DOI: 10.1038/nature24636.

[25] S. Lall. *How do neurons communicate (so quickly)?* Accessed: Aug. 08, 2024. Feb. 2019. URL: https://mcgovern.mit.edu/2019/02/28/ask-the-brain-how-do-neurons-communicate/.

[26] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[27] B. Lee et al. “An inductively-powered wireless neural recording and stimulation system for freely-behaving animals”. In: *IEEE Transactions on Biomedical Circuits and Systems* 13.3 (2019), pp. 413–424. DOI: 10.1109/TBCAS.2019.2909146.

[28] Michael S. Lewicki. “A review of methods for spike sorting: the detection and classification of neural action potentials”. In: *Network: Computation in Neural Systems* 9.4 (1998), R53–R78. DOI: 10.1088/0954-898X_9_4_001.

[29] J. 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)*. IEEE. June 2022, pp. 134–137. DOI: 10.1109/AICAS54282.2022.9869846.

[30] P. Lichtsteiner, C. Posch, and T. Delbruck. “A 128×128 120 dB 15 µs Latency Asynchronous Temporal Contrast Vision Sensor”. In: *IEEE Journal of Solid-State Circuits* 43.2 (Feb. 2008), pp. 566–576. DOI: 10.1109/JSSC.2007.914337.

[31] Henrik Lindén et al. “LFPy: a tool for biophysical simulation of extracellular potentials generated by detailed model neurons”. In: *Frontiers in Neuroinformatics* 7 (2014). ISSN: 1662-5196. DOI: 10.3389/fninf.2013.00041. URL: https://www.frontiersin.org/journals/neuroinformatics/articles/10.3389/fninf.2013.00041.

[32] S. Liu and Y. Yi. “Quantized neural networks and neuromorphic computing for embedded systems”. In: *IntechOpen* (2020). DOI: 10.5772/intechopen.91835.

[33] C.M. Lopez et al. “A neural probe with up to 966 electrodes and up to 384 configurable channels in 0.13 µm SOI CMOS”. In: *IEEE Transactions on Biomedical Circuits and Systems* 11.3 (2017), pp. 510–522. DOI: 10.1109/TBCAS.2016.2638688.

[34] W. Maass. “Networks of spiking neurons: The third generation of neural network models”. In: *Neural Networks* 10.9 (1997), pp. 1659–1671. DOI: 10.1016/S0893-6080(97)00011-7.

[35] S. Makeig et al. “Mining event-related brain dynamics”. In: *Trends in Cognitive Sciences* 8 (2004), pp. 204–10. DOI: 10.1016/j.tics.2004.03.008.

[36] D. A. McCormick and R. F. Thompson. “Neuronal responses of the rabbit cerebellum during acquisition and performance of a classically conditioned nictitating membrane-eyelid response”. In: *Journal of Neuroscience* 4.11 (Nov. 1984), pp. 2811–2822. DOI: 10.1523/JNEUROSCI.04-11-02811.1984.

[37] C. Mead. “Neuromorphic electronic systems”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1629–1636.

[38] Markus Nagel et al. “Up or Down? Adaptive Rounding for Post-Training Quantization”. In: *CoRR* abs/2004.10568 (2020). arXiv: 2004.10568. URL: https://arxiv.org/abs/2004.10568.

[39] J. Navajas et al. “Minimum requirements for accurate and efficient real-time on-chip spike sorting”. In: *Journal of Neuroscience Methods* 230 (2014), pp. 51–64. DOI: 10.1016/j.jneumeth.2014.04.018.

[40] “Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings”. In: *Science* (2024). Accessed: Aug. 19, 2024. URL: https://www.science.org/doi/10.1126/science.abf4588.

[41] Department of Neuroscience. *Contact - Department of Neuroscience*. https://neuro.nl/contact. Accessed: Aug. 21, 2024.

[42] Sung-Yun Park et al. “Dynamic Power Reduction in Scalable Neural Recording Interface Using Spatiotemporal Correlation and Temporal Sparsity of Neural Signals”. In: *IEEE Journal of Solid-State Circuits* 53.4 (2018), pp. 1102–1114. DOI: 10.1109/JSSC.2017.2787749.

[43] M. Pfeiffer and T. Pfeil. “Deep learning with spiking neurons: Opportunities and challenges”. In: *Frontiers in Neuroscience* 12 (2018). DOI: 10.3389/fnins.2018.00774.

[44] Yu Qi, Jiajun Chen, and Yueming Wang. “Neuromorphic computing facilitates deep brain-machine fusion for high-performance neuroprosthesis”. In: *Frontiers in Neuroscience* 17 (2023). ISSN: 1662-453X. DOI: 10.3389/fnins.2023.1153985. URL: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2023.1153985.

[45] X. Ran et al. “A hybrid autoencoder framework of dimensionality reduction for brain-computer interface decoding”. In: *Computers in Biology and Medicine* 148 (Sept. 2022), p. 105871. DOI: 10.1016/j.compbiomed.2022.105871.

[46] A. Sengupta et al. “Going deeper in spiking neural networks: VGG and residual architectures”. In: *Frontiers in Neuroscience* 13 (Mar. 2019). DOI: 10.3389/fnins.2019.00095.

[47] I. H. Stevenson and K. P. Kording. “How advances in neural recording affect data analysis”. In: *Nature Neuroscience* 14.2 (2011), pp. 139–142. DOI: 10.1038/nn.2731.

[48] J. Stuijt et al. “µBrain: An event-driven and fully synthesizable architecture for spiking neural networks”. In: *Frontiers in Neuroscience* 15 (May 2021). DOI: 10.3389/fnins.2021.664208.

[49] V. Sze et al. *Efficient processing of deep neural networks*. In Synthesis Lectures on Computer Architecture. Springer International Publishing, 2020. DOI: 10.1007/978-3-031-01766-7.

[50] C. Tan et al. “Autoencoder-based transfer learning in brain–computer interface for rehabilitation robot”. In: *International Journal of Advanced Robotic Systems* 16 (Apr. 2019), p. 172988141984086. DOI: 10.1177/1729881419840860.

[51] Tonic. *Introduction to Spiking Neural Networks (SNNs)*. https://tonic.readthedocs.io/en/latest/reading_material/intro-snns.html. Accessed: August 24, 2024.

[52] D. Valencia, P. P. Mercier, and A. Alimohammad. “Efficient in vivo neural signal compression using an autoencoder-based neural network”. In: *IEEE Transactions on Biomedical Circuits and Systems* 18.3 (June 2024), pp. 691–701. DOI: 10.1109/TBCAS.2024.3359994.

[53] Pascal Vincent et al. “Extracting and composing robust features with denoising autoencoders”. In: Jan. 2008, pp. 1096–1103. DOI: 10.1145/1390156.1390294.

[54] P.D. Wolf et al. “Thermal considerations for the design of an implanted cortical brain–machine interface (BMI)”. In: *NCBI Bookshelf* (2024). Accessed: Aug. 08, 2024. URL: https://www-ncbi-nlm-nih-gov.tudelft.idm.oclc.org/books/NBK3932/.

[55] T. Wu et al. “Deep compressive autoencoder for action potential compression in large-scale neural recording”. In: *Journal of Neural Engineering* 15.6 (Dec. 2018), p. 066019. DOI: 10.1088/1741-2552/aae18d.

[56] Jinho Yi et al. “Multichannel Many-Class Real-Time Neural Spike Sorting With Convolutional Neural Networks”. In: *IEEE Open Journal of Circuits and Systems* 3 (Jan. 2022), pp. 168–179. DOI: 10.1109/OJCAS.2022.3184302.

[57] M. Yin et al. “Wireless neurosensor for full-spectrum electrophysiology recordings during free behavior”. In: *Neuron* 84.6 (2014), pp. 1170–1182. DOI: 10.1016/j.neuron.2014.11.010.

[58] S. Zanos et al. “Relationships between spike-free local field potentials and spike timing in human temporal cortex”. In: *Journal of Neurophysiology* 107.7 (Apr. 2012), pp. 1808–1821. DOI: 10.1152/jn.00663.2011.

[59] J. Zhang et al. “A closed-loop compressive-sensing-based neural recording system”. In: *Journal of Neural Engineering* 12.3 (2015). Accessed: Aug. 08, 2024. URL: https://eprints.gla.ac.uk/117586/.

[60] X. Zhao et al. “Linear symmetric quantization of neural networks for low-precision integer hardware”. In: *International Conference on Learning Representations*. Accessed: Aug. 16, 2024. Sept. 2019. URL: https://openreview.net/forum?id=H1lBj2VFPS.

[61] T. Zhou et al. “Power-saving design opportunities for wireless intracortical brain computer interfaces”. In: *PMC* (2024). Accessed: Aug. 08, 2024. URL: https://www-ncbi-nlm-nih-gov.tudelft.idm.oclc.org/pmc/articles/PMC8286886/.

# Appendix A: RTL Implementation of SNN

*This appendix section describes how I implemented the RTL design for the SNN network based on future scope*.

## A.1 Hardware Implementation

After quantizing the final model, the network weights are fixed based on the results obtained from GPU simulations from the earlier sections. The focus then shifts to designing the hardware implementation tailored for brain-machine interfaces. This design process is divided into two primary components: neuron-level implementation and layer-level implementation. For RTL-level simulations of both components, I utilized Cadence Virtuoso.

### A.1.1 Neuron-Level RTL Implementation

Starting with the neuron-level simulations, the inputs and outputs are defined within a black-box framework (as seen in Fig. 1):

1. **i_spike**: This input varies depending on the neuron being implemented. For the input layer, i_spike is a 6-bit `int8` value, which is the output of the ADC used. The 6-bit input is divided into the most significant bit (MSB), which indicates the direction of events (up or down), and the remaining 5 bits, which represent the value. In the intermediate layers, i_spike is a 1-bit spiking value (0 or 1).
2. **i_weight**: This input loads the weights, which are fixed based on the GPU simulations. This is a 6-bit value, as previously quantized during the GPU simulation phase.
3. **reset_asyn**: This asynchronous reset signal acts as an on/off switch, with implementation starting only when this signal is high.
4. **clock**: The clock signal is maintained at a frequency similar to that of the electrodes sampling the neuronal signals. Given that neuronal spikes range from 300 Hz to 6 kHz, the Nyquist theorem dictates that a sampling frequency at least twice or more than this range is required to faithfully capture samples without aliasing.
5. **o_spike**: This is the output spike from our neuron, represented as a 1-bit value (0 or 1).

The behavior of the neuron can be mathematically described by the following equations:

$$U_{t+1} = (1 - \beta)U_t(1 - O_t) + (Wx_{t+1}) \tag{1}$$

$$O_{t+1} = \text{cmp}(U_{t+1}, V_{th}) \tag{2}$$

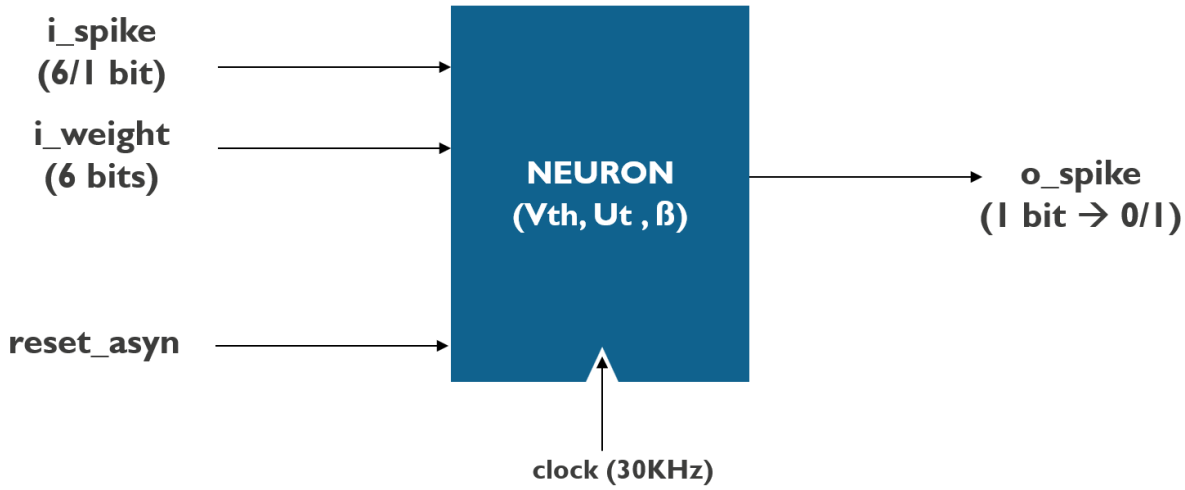



**Figure 1:** Neuron level model black-box framework

Here, $U_{t+1}$ represents the membrane potential at the next time step, $U_t$ is the current membrane potential, $O_t$ is the current output spike, $W$ is the weight, $x_{t+1}$ is the input at the next time step, $\beta$ is the leakage parameter, and $V_{th}$ is the threshold voltage. The first equation updates the membrane potential by considering the current potential, the leakage, and the new input weighted by $W$. The second equation generates the output spike $O_{t+1}$, which is 1 if the updated membrane potential $U_{t+1}$ exceeds the threshold $V_{th}$; otherwise, it is 0.

Once the high-level implementation of the neuron model was designed, we proceeded to examine its internal workings based on the governing equation 1. As illustrated in Fig. 2, the neuron receives 16 inputs (as detailed in section 4.3.2), which are organized into 16 fully connected groups, each containing 8 inputs. These inputs are then accumulated, and the resulting value is multiplied by the decay factor, represented as either $(1 - \beta)$ or $\tau$. Finally, the output is compared against the threshold value to determine whether the neuron generates a spike.

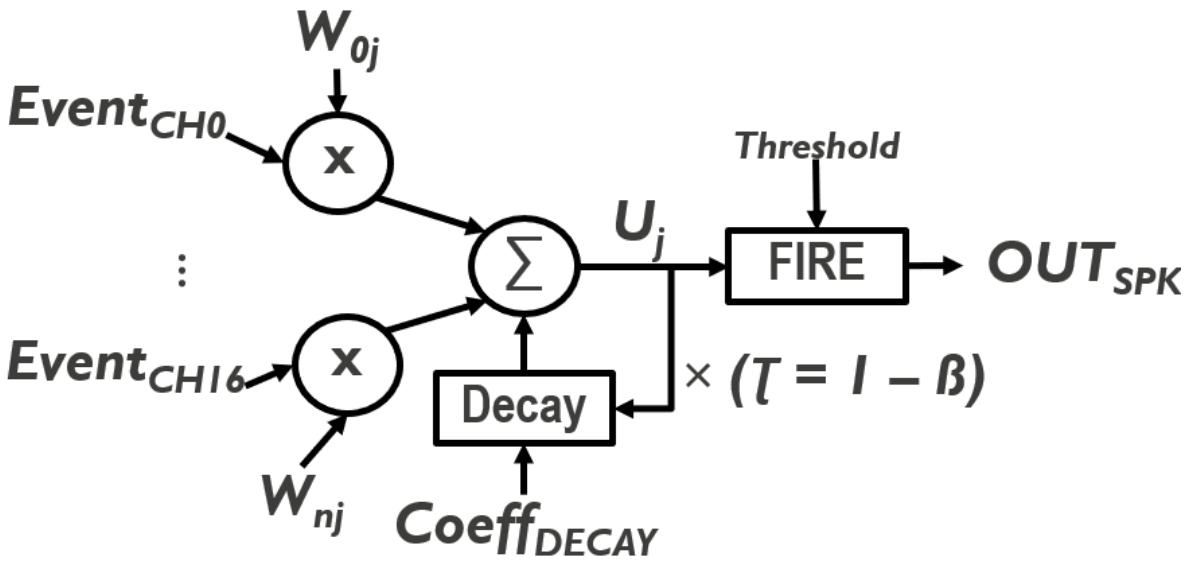



**Figure 2:** Internal implementation of neuron

As shown in Fig. 3, various blocks were utilized to simulate the operations described by Equation 1. These blocks are detailed as follows, starting from the leftmost weights block:

1. **Input and Weights Loading:** The neuron, along with all other neurons, parallel loads the weights in a single operation. This operation is directly related to the memory access metric defined in Section 5.6, which pertains to one of the load operations. If the input is from the ADC, a multiplexer is used to select the value based on the most significant bit (MSB), followed by a multiplication operation.
2. **Accumulation and Decay:** The multiplied values are then accumulated with the difference between $u_t$ and its right-shifted value. This operation is equivalent to multiplying the membrane potential by $(1-\beta)$, but instead of performing a direct multiplication, we use a right shift operation to optimize the process. For example:

$$\begin{aligned} 0.11111\,(\text{which is } 0.96875 \text{ in decimal}) \times u_t &= (1 - 0.00001) \times u_t \\ &= u_t - 0.00001 \times u_t \\ &= u_t - (u_t \text{ right-shifted by 5 units}) \end{aligned} \tag{3}$$

3. **Comparison and Spiking:** The resulting value, $u_{t+1}$, is then compared with the quantized threshold value to determine whether the output is 0 or 1. If the value is less than the threshold, the $u_{t+1}$ value is stored back into memory.

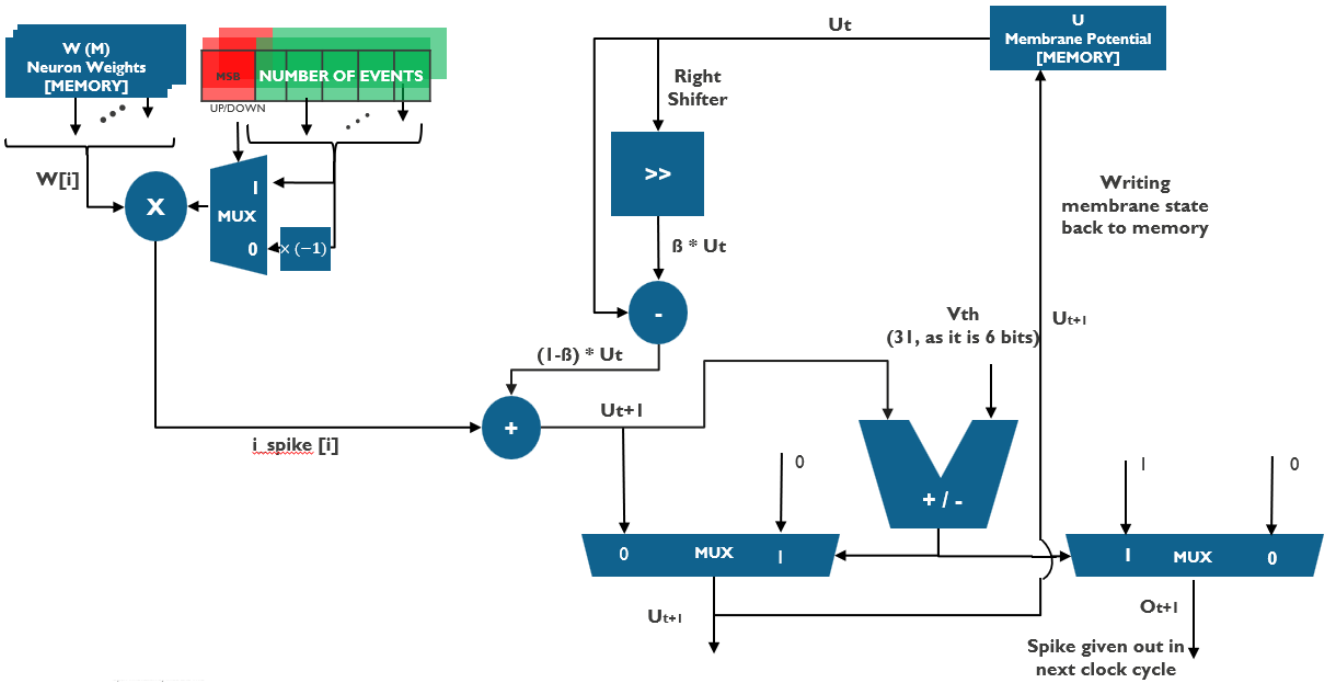



**Figure 3:** RTL logic design block diagram

## A.1.2 Layer-Level RTL Implementation

Once the neuron-level implementation was completed, a generate loop was used to group the neurons and form the layer. A testbench was created to load the weights and verify the results against those obtained from the RTL simulations.

## A.1.3 Simulation Results

In the neuron-level simulation (as shown in Fig. 4), the x-axis represents time in nanoseconds (ns), and the y-axis represents the membrane potential (in red) and spiking output. The membrane potential is shown as a red waveform, which increases or decreases based on the input stimuli and decay factor (0.96875). The spiking behavior of the neuron is determined by comparing the membrane potential to a predefined threshold value.

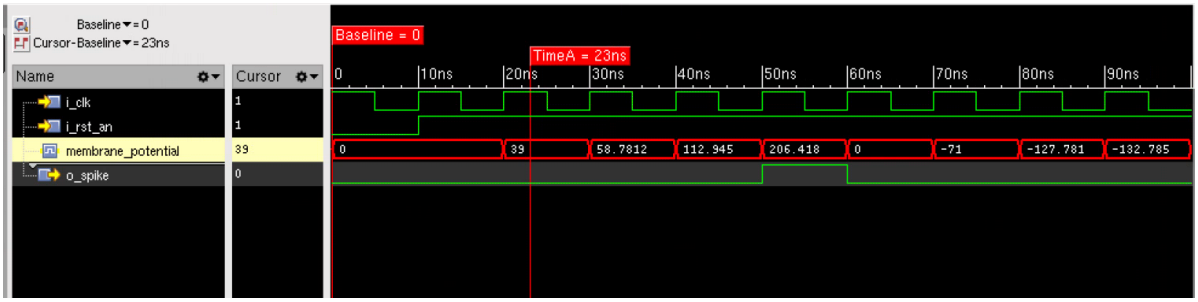



**Figure 4:** Neuron level RTL simulation

The threshold value is relatively high, which is due to the quantization of weights to 8 bits in the hardware implementation. This quantization results in larger discrete steps in the membrane potential, requiring a higher threshold for spiking. When the membrane potential exceeds this threshold, the neuron produces a spike, which is indicated by a transition in the output waveform.

In the waveform shown, the membrane potential increases incrementally as inputs are integrated over time. However, due to the decay or inhibitory inputs, the potential may decrease as well. When the potential does not reach the quantized threshold, no spike is generated, as evidenced by the low state of the output spike signal.

The layer-level simulation (as depicted in Fig. 5) also has the x-axis representing time in nanoseconds (ns), while the y-axis shows the spiking activity of multiple neurons within the layer. Each row in the y-axis corresponds to the spike output of a different neuron within the layer.

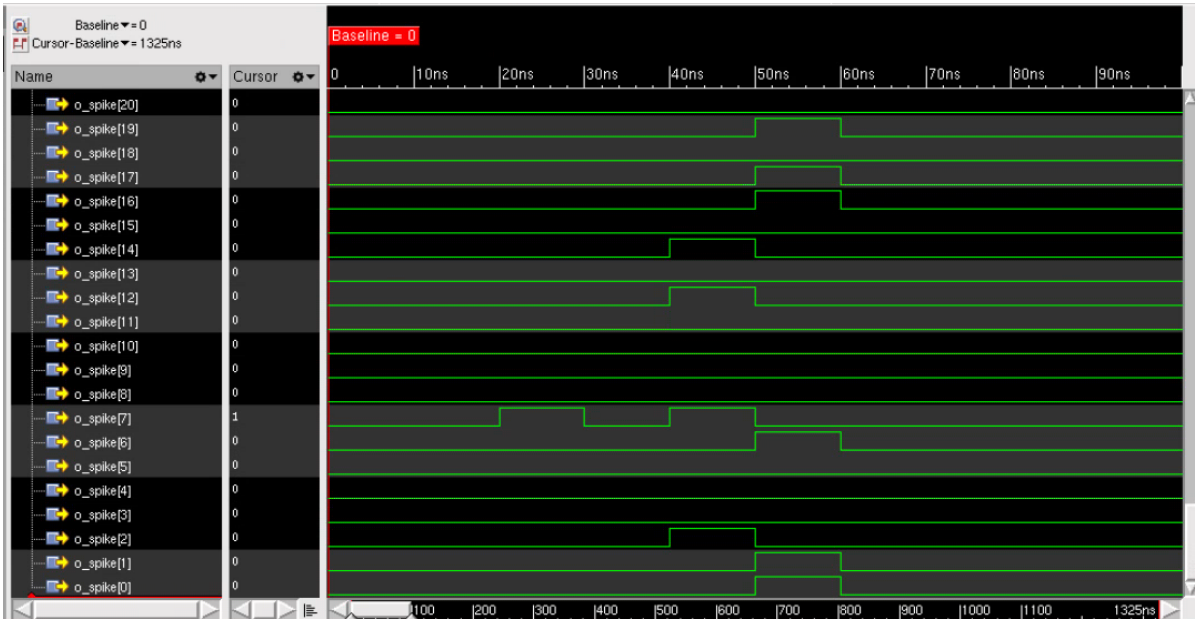



**Figure 5:** Layer level RTL simulation

In this simulation, the neurons exhibit spiking behavior when their respective membrane potentials exceed the threshold, similar to the neuron-level simulation. The timing and pattern of spikes across the neurons reflect the collective processing of input signals by the network layer for each of the 128 channels. This behavior is crucial for understanding the dynamic response of the spiking neural network under different input conditions.

# Appendix B:Timeline

*The following timeline outlines the key stages and milestones of my thesis project, detailing the progression of tasks and their respective timeframes. This structured timeline was designed to ensure systematic progress and timely completion of the thesis.*

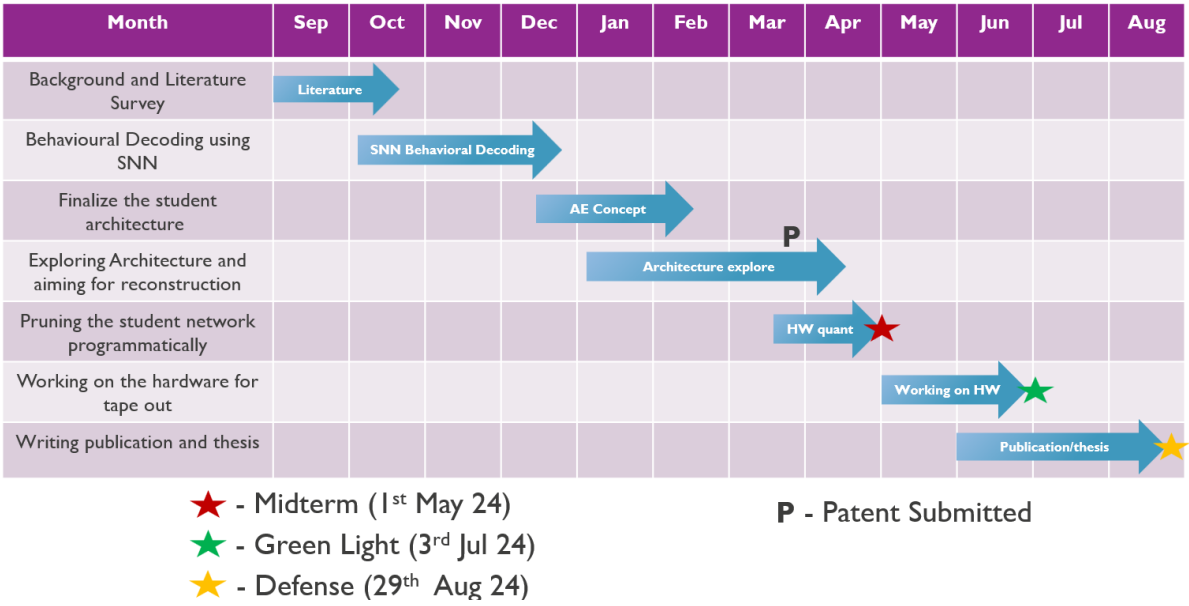



**Figure 6:** Timeline for Inception to Results

1. **Background and Literature Survey (September - October 2023):** The project's initial phase focused on conducting a comprehensive background and literature survey. This involves reviewing existing research and relevant literature to establish a strong theoretical foundation for the thesis. The literature review continued until the end of October.
2. **Behavioral Decoding using SNN (October - December 2023):** Following the literature review, the next task was to explore behavioral decoding using Spiking Neural Networks (SNN). This task involves developing and testing SNN models to understand their effectiveness in behavioral decoding. This project was mainly to familiarize myself with the working of SNNs. This phase spanned from October to December.
3. **Finalizing the Student Architecture (November - December 2023):** During this period, the architecture for the student model, which will be central to the thesis, was finalized. This involved defining the structure and parameters of the model based on the insights gained from the SNN exploration.
4. **Exploring Architecture and Aiming for Reconstruction (January - March 2024):** The next phase involved exploring the architecture further, with a focus on reconstruction. This task covered detailed architecture exploration and testing, leading to a refined model that aligns with the thesis objectives. This exploration continued through March 2024.
5. **Pruning the Student Network Programmatically (March - April 2024):** In this phase, the focus shifted to optimizing the student network by pruning unnecessary connections programmatically. This step was crucial to enhance the model's efficiency and performance.
6. **Working on Hardware for Tape-Out (May - July 2024):** As the thesis progressed toward implementation, the focus was on developing the hardware necessary for the tape-out process. This

stage involves close work on hardware integration and testing, ensuring the model is ready for practical deployment.

7. **Writing Publication and Thesis (July - August 2024):** The final phase of the timeline is dedicated to writing and compiling the thesis and associated publications. This process will involve documenting the research, results, and conclusions in a coherent and academically rigorous manner, culminating in the final thesis defense.

## Milestones

- **Midterm (1st May 2024):** A midterm review was conducted to assess the progress made so far and to make any necessary adjustments to the timeline or project scope.
- **Green Light (3rd July 2024):** The ”Green Light” milestone marks the point where the project was formally approved to proceed to the final stages of thesis writing and hardware completion.
- **Defense (29th August 2024):** The thesis defense is scheduled for late August, marking the culmination of the project and the presentation of the final results.
- **Patent Submitted (March 2024):** During the architecture exploration phase, a patent was also submitted to protect any novel methodologies or designs developed as part of the thesis

# Appendix C:IOP Publication Final Draft

*Please find on the next page the final draft intended for submission to the journal Neuromorphic Computing and Engineering (IOP Science). This draft has undergone multiple reviews by various colleagues at Imec. While it is nearly complete, a few minor modifications will be made before the final submission.*

# Neuromorphic Compression and Distributed Computing for On-Implant Neural Signal Processing in Brain-Computer Interfaces

**1.Author Names**: Lohit Gandham[12], Hua-Peng Liaw[2], Gaungzhi Tang[3], Manolis Sifalakis[2], Yuming He[2], Rajendra Bishnoi[1], Yao-Hong Liu[1, 2]

[1]Computer and Embedded Systems Engineering, TU Delft, Netherlands
[2]IMEC, Netherlands
[3]Department of Advanced Computing Sciences Maastricht University, Netherlands

**Email**:

## Abstract

Neuromorphic systems offer a promising solution to the computational challenges of intra-cortical Brain-Computer Interfaces (iBCIs), leveraging the event-driven nature of biological neural networks for enhanced power efficiency and data scalability. The exponential growth in neural data rates due to advancements in high-density neural recording creates significant hurdles in data transmission and processing. This paper introduces a novel neuromorphic compression architecture that combines delta-modulated level crossing events with Spiking Neural Networks (SNNs) trained through a two-stage process with grouped inputs. The proposed system achieves a compression ratio exceeding 94× and a peak Signal-to-Noise and Distortion Ratio (SNDR) of 22 dB compared to the signal without Local Field Potentials. This substantial improvement over existing methods drastically reduces data throughput making it ideal for implantable devices where resource constraints are critical. Validated on hippocampal and cerebellar datasets, our approach significantly outperforms state-of-the-art techniques in terms of computational and compression efficiency, SNDR, and memory usage—requiring just 384 bits per channel to store necessary data for processing and 18.07K Multiply and Accumulate operations (MACs) per spike. These findings underscore the potential of neuromorphic SNNs as a transformative technology for future iBCIs, advancing neural prosthetics and brain-machine interfaces.

Keywords: Neuromorphic, Spiking Neural Networks, Brain-Computer Interface, Neural Signal Compression, Delta Modulation, Implantable Devices, Signal-to-Noise and Distortion Ratio, Compression, Autoencoders.

## Introduction

Exploration of the human brain and the way it processes and decodes information has long been a topic of profound interest. Over the last few decades, developments in the field of Intra-cortical Brain-Computer Interfaces (iBCIs). particularly in both invasive approaches like intra-cortical BCIs, which involve placing electrodes directly into the brain's cortex, and less invasive methods such as electrocorticography (ECoG), which place electrodes on the surface of the cortex, have enabled researchers to bridge the gap between our understanding of brain function and the technology that can record and interpret neural signals. These advancements have had a significant impact on medicine—such as treating disabilities, epilepsy, and Parkinson's disease—as well as on neuroscience research and engineering disciplines [1],[2].

In today's BCI landscape, a significant challenge lies in processing a large number of action potentials recorded from numerous electrodes. As demonstrated in a study by Stevenson and Kording [3], advancements in neural recording techniques over the past fifty years have resulted in the number of simultaneously recorded neurons doubling

 © xxxx IOP Publishing Ltd

roughly every seven years, paralleling Moore's law. This exponential increase has led to a huge influx of data, challenging data transmission and recording systems.

Furthermore, spike sorting—essential for distinguishing individual neuron activity—requires sampling at frequencies as high as 30kHz to capture the detailed shape of each spike [4]. This further amplifies the volume of data generated, exacerbating the strain on current systems. Additionally, Even-Chen et al.[5] have shown that the most power-consuming part in many conventional, commercial systems and academic prototypes available today is the wireless data transmitter, hence justifying the need for high bandwidth BCIs and compression of recorded neural signals.

For instance, the neural probe described in [6] produces a data rate of 144 Mbps from 384 channels, each operating at a sampling rate of 30 kHz with a 12-bit ADC resolution. This significant data output presents considerable challenges for data storage and transmission, underscoring the need for sophisticated processing capabilities and high-bandwidth communication channels.

To address the challenges of processing large volumes of neural data within the constraints of implantable devices, researchers are increasingly adopting neuromorphic computing. Neuromorphic systems emulate biological neural networks, enabling online data processing with minimal energy consumption [7], [8]. These systems incorporate event-based sensors that, like neurons, respond only to significant environmental changes, reducing unnecessary data processing [9]. Given the need to minimize implant volume and power consumption—essential for reducing invasiveness and preventing heat-related tissue damage—traditional methods like parallelization or multiplexing are impractical for implants. Maintaining a temperature increase of less than 1ºC for surrounding brain tissue is crucial for long-term health, with this requires implant electronics to stay below a heat flux of 40 mW/cm² surface area Wolf et. al[10] .

Our research introduces a novel spiking neural network (SNN) based compression approach to address the data challenges in iBCIs. Neural networks excel in this context due to their ability to model complicated, nonlinear relationships within large datasets, which is relevant for handling the intricacies of neural signals [11]. SNNs, a form of neuromorphic computing, enhance this approach by mimicking the brain's natural, event-driven processing [7], making them particularly energy-efficient and well-suited for on-device processing in BCIs, where minimizing power consumption and heat generation is critical to prevent tissue damage and ensure implant longevity [12], [13].

Our approach achieves a compression ratio of ~94× significantly reducing data rates while preserving the accuracy of recorded action potentials, significantly reducing data rates without sacrificing the accuracy of spike sorting. It also minimizes memory usage, cutting down the required memory per channel by almost half , and enhances computational efficiency by substantially reducing both the number of computations and memory accesses per spike[14].

## Background and related work

Over the past decade, significant advancements have been made in the field of neural signal compression, aimed at addressing the critical issues of transmission bandwidth and power consumption, as outlined below.

### *2.1 Transformations:*

This type of compression aims to transform the neural signal into a different domain where the signal is sparse and reduced number of parameters can be efficently identified and quantized to represent action potentials and thus more suitable for compression and transmission (such as frequency or wavelet domain). Commonly used techniques include , e.g. Principal Component Analysis (PCA), Discrete Cosine Transform (DCT), wavelet, linear approximations, etc [15].

Zhang [16] mentions that the hardware requirements for such transformations are substantial, as they often necessitate complex digital filters and additional memory operating at high speeds, which significantly increases the circuit area and on-chip power consumption. Moreover due to the signal-agnostic natures of the mentioned methods they are more often not more optimized for compression resulting in lack of representational power in the transformed domains leading to limited compression [15]. This makes these methods less feasible for large-scale neural recording systems that require efficiency and scalability .

### *2.2 Delta Modulation for Neural Recording Systems*

Delta Modulation is particularly beneficial for sparse action potential signals, such as those found in electrophysiology, because it inherently reduces data rates by focusing on difference or 'delta' between successive signal samples rather than transmitting redundant information. In the context of sparse neural signals, where action potentials (spikes) occur intermittently, Delta Modulation only records and transmits the significant changes (the spikes) while ignoring the long periods of inactivity. This selective sampling is highly efficient, reducing the amount of data that needs to be processed and transmitted, which is crucial for resource-constrained systems like Brain-Computer Interfaces (BCIs).

He et al. [17] expanded upon this concept by integrating a Delta-modulated Analog-to-Digital Converter (Δ-ADC) within an event-based framework. The Δ-ADC triggers an event only when the difference (Δ) between consecutive samples exceeds a predefined threshold, minimizing the generation of unnecessary data , where $x(t_i)$ and $x(t_{i-1})$ are the signal values at consecutive time points, and Δ is the change that prompts data transmission.

$$\Delta = x(t_i) - x(t_{i-1}) \quad (1)$$

In our work, we leverage the efficiency of He's Δ-ADC as a foundational component within our system. In addition, we integrate Spiking Neural Networks (SNNs), which enables our system to compress the already sparse and event-driven data from the Δ-ADCs further.

### *2.3 Block-based compression:*

The block-based method (epoch) is a well-established technique in neural signal processing, particularly in Brain-Computer Interfaces (BCIs). It involves segmenting continuous neural data into discrete time intervals called epochs, which align with specific events or periods of interest. This segmentation allows for focused analysis and compression of the most relevant data segments, reducing overall data volume and computational load [19].

In BCIs, epochs are typically aligned with specific stimuli or actions, such as motor imagery tasks or event-related potentials (ERPs). For instance, an epoch may begin just before a visual cue and extend through the duration of the neural response, effectively capturing the critical data [20]. Let's denote the continuous neural signal as $x(t)$, where $t$ represents time. The signal is divided into $N$ epochs, each of duration $T$ seconds. The *n-th* epoch, $E_n(t)$, can be defined as in Equation (2).

$$E(t)_n = x(t) \times w_n(t) \quad (2)$$

where $w_n(t)$ is a window function that extracts the *n-th* epoch. This window function is defined as:

$$w_n(t) = \begin{cases} 1, & if\ nT \le t < (n+1)T \\ 0, & otherwise \end{cases} \quad (3)$$

While effective, the epoch method has limitations, such as the challenge of capturing continuous, non-time-locked activities and the potential for significant memory usage, as noted by Biederman et al. [21], where each channel demands around 1500 bits for processing the incoming action potentials, potentially increasing system complexity and power consumption.

In our work, we use the epoch method as a baseline to highlight the advantages of our proposed system, which integrates Delta encoding with Spiking Neural Networks (SNNs).

### *2.4 Compressed Sensing:*

Compressed Sensing (CS) has emerged as a prominent technique to address the limitations of traditional sampling methods, particularly in scenarios requiring high efficiency and low resource utilization. Zhang et al. [16] explored the application of Compressed Sensing in neural signal acquisition systems, demonstrating its potential to significantly reduce the amount of data required for accurate signal reconstruction. The CS method works by acquiring 'Sub-Nyquist' samples, where the signals are sampled at a frequency lower than the Nyquist rate, under the assumption that the signals are sparse in some domain. This is achieved using a sampling matrix (Ø) that compresses the signal directly during the acquisition process, thereby circumventing the need to store all samples before compression.

Zhang's work involves transforming the traditionally sampled signals into sparse representations using a transformation basis (η), which is then compressed to form a much smaller dataset (as seen in Equation. (4)). The compressed data is then processed to reconstruct the original signal by solving an optimization problem to find the sparse solution, a process which can be computationally intensive and is typically performed off-chip.

$$\begin{aligned} y &= \emptyset x \\ &= \emptyset\,(\eta s) \end{aligned} \quad (4)$$

However, while Zhang's method is effective, it faces limitations in real-time applications due to the complexity of the optimization required for signal reconstruction, which presents challenges for on-chip implementation. This is particularly evident in the peak Signal-to-Noise and Distortion Ratio (SNDR) performance, where Zhang's method struggles to maintain high fidelity in signal reconstruction, especially at higher compression ratios. Additionally, the method requires extra memory for storing the parameters associated with the sampling and transformation matrices, further complicating its implementation.

### *2.5 Autoencoders:*

Autoencoders, a class of deep neural network frameworks, are employed for data compression by transforming input data into a lower-dimensional representation. This transformation is governed by the parameter *W* and its associated nonlinearity **Ø**. The encoder, which constitutes the first stage of the autoencoder, generates this compressed representation, enabling efficient data compression. Subsequently, the decoder, defined by the transformation parameter *W'* and non-linearity **Ø'**, reconstructs the original data from the compressed representation, as described by Equation (5). The primary objective is to minimize the loss between the reconstructed output and the original input, thereby optimizing the parameters of both the encoder and decoder (*W* and *W'*). To achieve this minimization, traditional optimization techniques, such as gradient descent, are employed.

$$y = \emptyset'(W'\emptyset\,(Wx)) \tag{5}$$

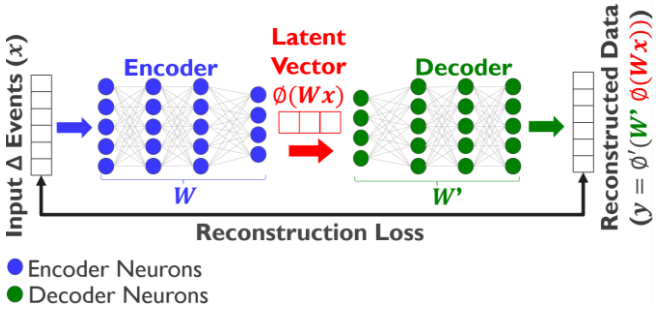



**Figure 1:** Schematic diagram of the encoder-decoder network architecture showing the non linear (**Ø** and **Ø'**) latent vector and reconstruction transformation.

This encoder-decoder structure is utilized for its inherent design of compression, which effectively reduces the dimensionality of the input data while retaining the most critical information. This capability makes the encoder-decoder network particularly suitable for applications requiring efficient data representation and reconstruction. A high-level diagram of this architecture is shown in Fig. 1.

Previous research on autoencoders for neural compression has primarily focused on EEG and ECG or, when using electrophysiology signals, has utilized local field potentials (LFPs) as input data [22], [23], [24]. In contrast, this work employs extracellular in vivo spiking data, which provides significant advantages in decoding accuracy and temporal resolution compared to LFPs [25]. The only comparable work is by Wu et al. [15], which uses a CNN-based asymmetric autoencoder.

Wu's method, while effective in using CNNs for neural data compression, faces significant challenges due to its high computational and memory demands—requiring 79.25K MAC operations and 237.75K memory accesses per spike. This makes it less suitable for resource-constrained environments like implantable devices, where power efficiency and low latency are crucial.

## Neuromorphic Compression System

The proposed system represents a notable advancement in neural data compression and processing, especially for resource-constrained environments like implantable devices. Unlike traditional methods such as compressed sensing [16], epoch[21] and CNN-based autoencoders[15], which often struggle with hardware complexity, computational demands, and the need for significant buffering that increases memory usage, our system is optimized for power efficiency and real-time processing. This makes it especially well-suited for on-implant applications, where minimizing both computational load and memory footprint is critical.

The Neuromorphic Compression system, shown in Fig. 2, is at the heart of this innovation, integrating two key components: an Delta-modulated Analog-to-Digital Converter (Δ-ADC) [13] and a Neuromorphic Autoencoder. It achieves nearly five times greater data reduction and significantly reduces computational demands compared to current methods, enhancing storage efficiency and data transmission speed. Our approach is further refined by using knowledge distillation, where the on-implant SNN is trained in two stages with a more powerful teacher network. The encoder first compresses the data, and the decoder is fine-tuned to maintain signal fidelity.

Additionally, by grouping inputs from nearby electrodes, we reduce memory usage with minimal impact on signal reconstruction. These advancements position our system as a highly efficient solution for neural data processing, ideal for advancing implantable device technology.

The primary function of the amplifier and band pass filter before the proposed neuromorphic compression system is to enhance the raw signals from the microelectrode array (MEA), as shown in Fig. 3. The raw data is very low in amplitude (50-500 µV) and hence needs to be amplified before it can be processed. Additionally, a band-pass filter is applied to isolate the spikes (300 Hz – 6 kHz). [26].

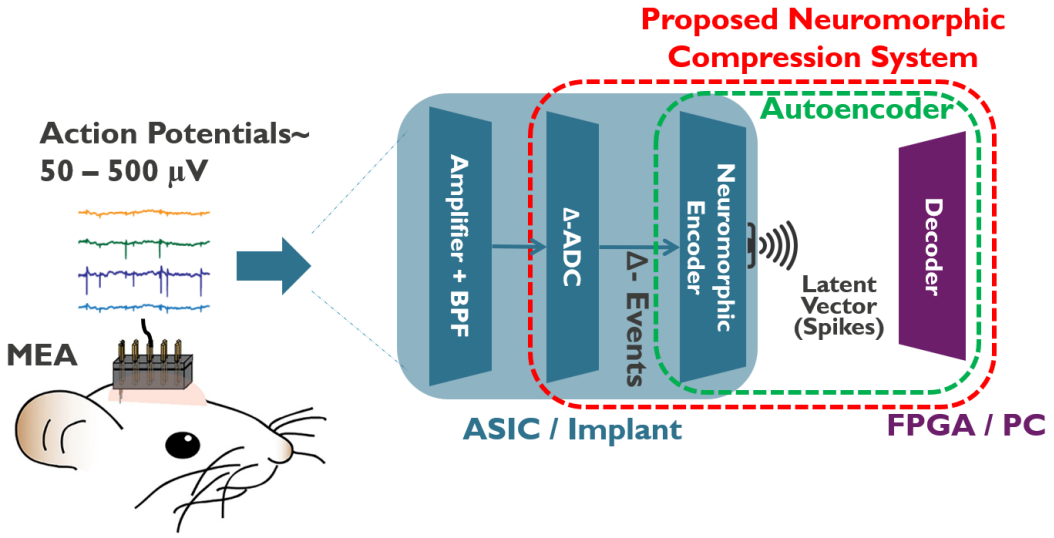



**Figure 2:** High-Level architecture for our Neuromorphic Compression System. The system processes action potentials which are apmlified and filtered from a Multielectrode Array (MEA).

### *3.1 Delta-modulated Analog-to-Digital Converter (Δ-ADC)*

The Δ-ADC is crucial in converting amplified analog action potentials into digital signals, facilitating the transformation of raw action potentials into delta (Δ) - modulated events as detailed in [17]. This delta modulation technique efficiently represents action potentials in a digital format, essential for subsequent processing steps.

Neural data can be divided into two categories: spikes, which are fast-changing, short-lasting signals from a single neuron, and local field potentials (LFPs), which are slow-changing, long-lasting activities from the collective signals of large neural populations. While both types of signals carry valuable information, spikes are particularly rich in spatial and temporal details, making them more critical for precise neural decoding [27].

In our system, Local Field Potentials (LFPs) are effectively filtered out using a bandpass filter, allowing us to focus exclusively on spikes, which contain the most relevant information. The subsequent delta modulation process computes the difference between consecutive samples, generating non-zero delta values only when significant changes occur. This approach ensures that events are generated only when thresholds are crossed, aligning with an event-based paradigm. This technique effectively reduces the dynamic range of the data and ensures that it remains sparse over time, optimizing both the data compression and the system's overall efficiency. By concentrating on spikes and filtering out LFPs, we enhance the system's ability to capture and process the most informative aspects of neural activity, leading to more accurate and efficient neural data processing.

The Fig. 3 illustrates the process of delta modulation, where the top panel shows the raw action potentials (orange) and bottom panel displays the detected level delta events (blue). This Δ-ADC is selected for its significant advantages in managing the challenges of intracortical brain-computer interfaces (BCIs). It addresses the high data volume from numerous recording channels by implementing a high compression ratio ($CR_{\Delta ADC}$) in the range of 10-100× on extracelluar neural recordings, substantially reducing the data that needs to be transferred. Additionally, it offers low power consumption, with each channel consuming only 1 µW, and maintains spike waveform fidelity with a Signal-to-Noise and Distortion Ratio (SNDR) of 36 dB even for spike amplitudes as low as 31 µV, ensuring accurate neural signal representation.

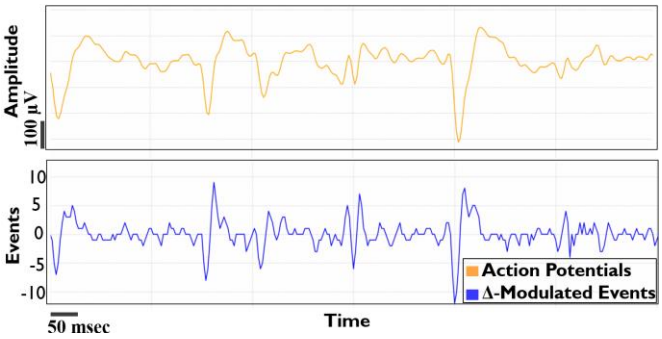



**Figure 3:** The top panel (orange) depicts the reconstructed waveform through cumulative summation while the bottom panel (blue) shows delta-modulated level crossing events over time.

### *3.2 Neuromorphic Hybrid Autoencoder*

The Neuromorphic Hybrid Autoencoder is the core of our system, designed to compress and process neural data with exceptional efficiency. Starting with the event-based output from the Δ-ADC, the autoencoder's primary function is to significantly reduce data size while preserving essential features, optimizing both storage and transmission. This is accomplished through a robust encoder-decoder network which is meticulously trained using gradient descent to minimize reconstruction loss.

To further enhance efficiency, knowledge distillation is employed to transfer the learned features from the teacher model to a more efficient, event-driven Spiking Neural Network (SNN). This transfer occurs in two stages, with the encoder and decoder being trained successively. The result is a final SNN that inherits the accuracy of the larger model while operating with significantly lower computational and power demands, making it ideal for edge devices and resource-constrained environments.

The following subsections will provide detailed explanations of the encoder-decoder network's architecture and training process, the application of knowledge distillation, and the final implementation of the Spiking Neural Network (SNN). These components work together to deliver the high levels of compression and efficiency that define the crux of our system.

#### *3.2.1 Encoder – Decoder Network*

The encoder-decoder architecture is a widely utilized sequence-to-sequence transformation network [25], which we have adapted to process the input data—specifically, the output from the Δ-ADC in our case—by compressing it into a significantly lower-dimensional representation, commonly referred to as the latent space or latent vector. The decoder then reconstructs the original input from this latent space.

As described in Equation (5) and illustrated in Fig. 1, the reconstructed data $y$ is a nonlinear function derived from the latent space. The weight parameters $W$ and $W'$ are traditionally optimized using reconstruction loss, which is minimized by applying the gradient descent algorithm to fine-tune these parameters [26]. Our implementation employs a three hidden layer encoder network with a symmetrically identical decoder. The rationale for this design choice will be discussed in detail in Sections 5.1 and 5.2.

#### *3.2.2 Knowledge Distillation*

Knowledge distillation, also known as the teacher-student architecture, is a powerful technique for transferring knowledge from a large, complex neural network (the teacher) to a smaller, more lightweight model (the student) [28]. This approach is essential for achieving compatibility with edge devices due to the constraints on computational resources and power consumption.

In our proposed method, we utilize a large and powerful teacher network. The neurons in both the encoder and decoder of the teacher network (as depicted in Fig. 5) were tested with various types of architectures, including Convolutional Neural Networks (CNN), Spiking Neural Networks (SNN), Long Short-Term Memory (LSTM) networks, and Recurrent Neural Networks (RNN). Our results, detailed in Section 5.2, indicate that RNN and LSTM architectures are the most effective. They not only demonstrate superior knowledge distillation capabilities compared to CNNs but also align more closely with SNNs in their ability to handle sequential data.

The reconstruction error and compression ratio of the network can be tuned by adjusting the size of the latent vector. After training the teacher autoencoder network, the latent vector and decoder weights are fixed to train the student network. The visual representation of this process is illustrated in Fig. 4. The subsequent will elaborate on the rationale behind choosing an SNN for the student network.

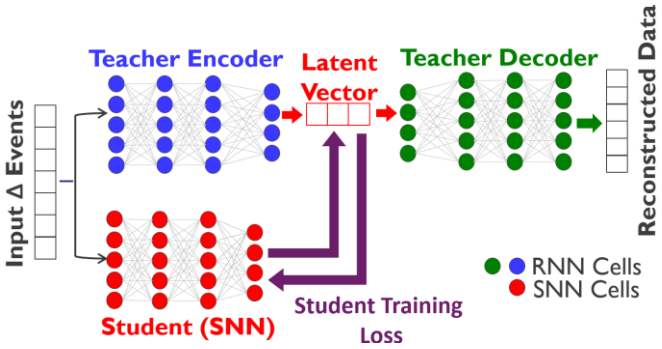



**Figure 4:** Knowledge distillation process in teacher – student architecture.

#### *3.2.3 Spiking Neural Network (SNN)*

The final component that completes our system is the implementation of the student network as a Spiking Neural Network (SNN). This choice leverages the unique operational characteristics of SNNs, which closely mimic the behavior of biological neurons, thereby achieving high power efficiency. Unlike traditional neural networks that use continuous values for layer-to-layer communication, SNNs utilize binary values, known as spikes (0 or 1). This transition simplifies the traditional Multiply-And-Accumulate (MAC) operation to a mere accumulate operation, facilitating more efficient event-driven hardware operations [29]. Due to the sparsely distributed nature of inputs, SNNs can significantly reduce both power consumption and hardware overhead. Fig. 6

illustrates the activity of a single neuron in an SNN for combined spiking inputs (x1,x2,x3,x4).

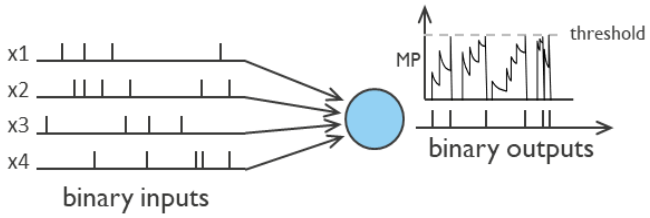



**Figure 5:** Illustration of a binary input system processing signals through a neuron model. The membrane potential (MP) is compared against a threshold to generate binary outputs.

In an SNN cell, the neuronal state is represented by an intrinsic state called the membrane potential (MP). The MP changes in response to incoming inputs and, upon reaching a specific threshold, the neuron generates a spike to the subsequent layer while resetting the MP to zero (hard reset). This temporal storage mechanism allows the SNN to capture the dynamic temporal characteristics of input data, reacting only to events. Since the output of the Δ-ADC is also obtained through event-based sampling, this alignment enhances the overall efficiency of the system.

Additionally, the decision to choose SNNs was influenced by their event-based nature, particularly due to the integration-and-fire operation. As the output from the Δ-ADC (referenced from [17]) is sampled in an event-based fashion, spikes are generated in the SNN only when an event occurs. Once the spiking neuron accumulates these events to a certain threshold, it fires. This event-based sampling nature of the Δ-ADC and the subsequent accumulation operation inherently introduces a degree of compression even without dimensionality reduction.

Despite the inherent hardware efficiency and biological plausibility of SNNs, their learning capabilities can be limited by the temporal precision required for event-based communication. In our system, some information loss may occur due to the use of a lower frequency clock, which may not fully capture the temporal dynamics encoded in the spikes [13]. To address this issue, we utilize knowledge transfer from a larger and more complex network, as discussed in section 3.3.2 (and illustrated in Fig. 4). This approach aims to recover a substantial amount of accuracy, thereby making SNNs suitable for implementation on edge devices.

### *3.3 Training the Spiking Neural Network*

Our system utilizes two primary innovative strategies to train the teacher-student autoencoder structure. These strategies are designed to ensure that the spiking student network optimally compresses the events from the Δ-ADC. Detailed explanations of these strategies are provided in the subsequent subsections.

#### *3.3.1 2 Stage Student Encoder-Decoder Training*

After training the teacher network, we fix its weights and proceed with the training of the student network in two distinct stages. This staged approach allows the student network to effectively capture and replicate the essential features learned by the teacher network, ensuring that the critical aspects of the data compression process are preserved. This method is inspired by layer-wise training techniques, which have been shown to improve the stability and performance of deep networks by isolating the learning process of different components[28][30]. By fixing the encoder weights after initial training, we ensure that the student network benefits from a robust feature extraction process before fine-tuning the decoder.

In the first stage, we train only the encoder of the student network while utilizing the teacher's decoder with its weights fixed. The loss function used during this stage is the mean squared error (MSE) between the latent spaces of the student and teacher networks. This strategy ensures that the encoder of the student network closely mimics the behavior of the teacher network, effectively performing a layer-by-layer training. This process is illustrated in Fig. 6(a).

In the second stage, we focus on training the student decoder, which is initialized with the pre-trained weights of the teacher decoder. During this phase, we fix the weights of the student encoder, allowing the decoder to adapt to the latentspace learned in the first stage. This step fine-tunes the student decoder to its corresponding latent space, ensuring cohesive and efficient compression. The detailed process of this stage is depicted in Fig. 6(b).

By adopting this two-stage training method, we ensure that the student network effectively learns to compress data by leveraging the knowledge distilled from the teacher network, leading to significant improvements in performance and efficiency. Through simulations, we demonstrate that this approach achieves a remarkable 9.4 × compression ($CR_{SNN}$) in the final spiking layer of the encoder, highlighting the substantial benefits of the two-stage training process. These results underscore the effectiveness of our method in optimizing data compression while maintaining high levels of accuracy, making it an ideal solution for resource-constrained environments

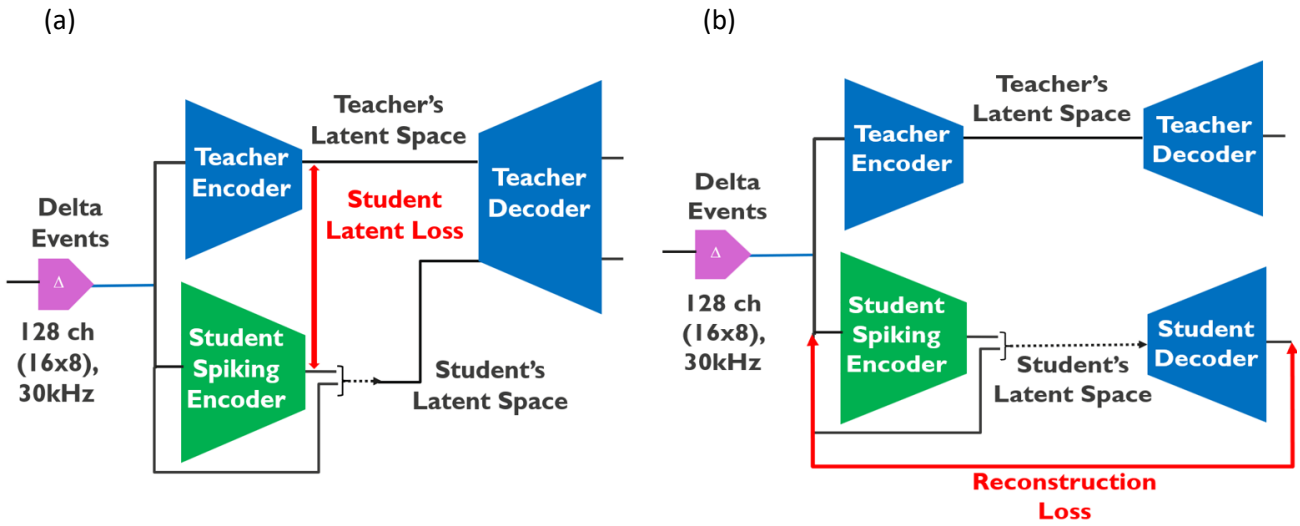



**Figure 6:** (a) Stage 1: student encoder training using the teacher's latent space. (b) Stage 2: student decoder training using the reconstruction loss

### *3.3.2 Hardware Aware Optimizations*

To enhance the system's efficiency for deployment across various hardware platforms, it was essential to optimize the number of weights in the student Spiking Neural Network (SNN) designed for the implant. This optimization process involved both pruning and quantization of the weights. Pruning effectively streamlined the network by eliminating weights that were deemed less important. Subsequently, the remaining weights were quantized, given that PyTorch, by default, utilizes the float32 datatype. By converting these weights to int6, we improved memory efficiency, achieving nearly 1.5× reduction in bits per channel compared to the state-of-the-art deep learning model used for compression (will see in section 5.6). These optimizations resulted in a more efficient and compact model, making it highly suitable for hardware implementation.

Leveraging the principle that neurons primarily communicate with their nearest neighbors, we focused on pruning weights connected to more distant neurons. This neuromorphic approach mirrors biological neural networks, where synaptic inputs from nearby neurons are prioritized over those from farther ones. By emulating this natural efficiency, we significantly reduced the computational burden of our SNN. Pruning distant connections helped eliminate redundant or less influential synapses, which in turn reduced the complexity and energy consumption of the network. This biologically inspired strategy ensured that the core functionality and performance of the SNN were preserved while minimizing the hardware resources required for its implementation [31], [32].

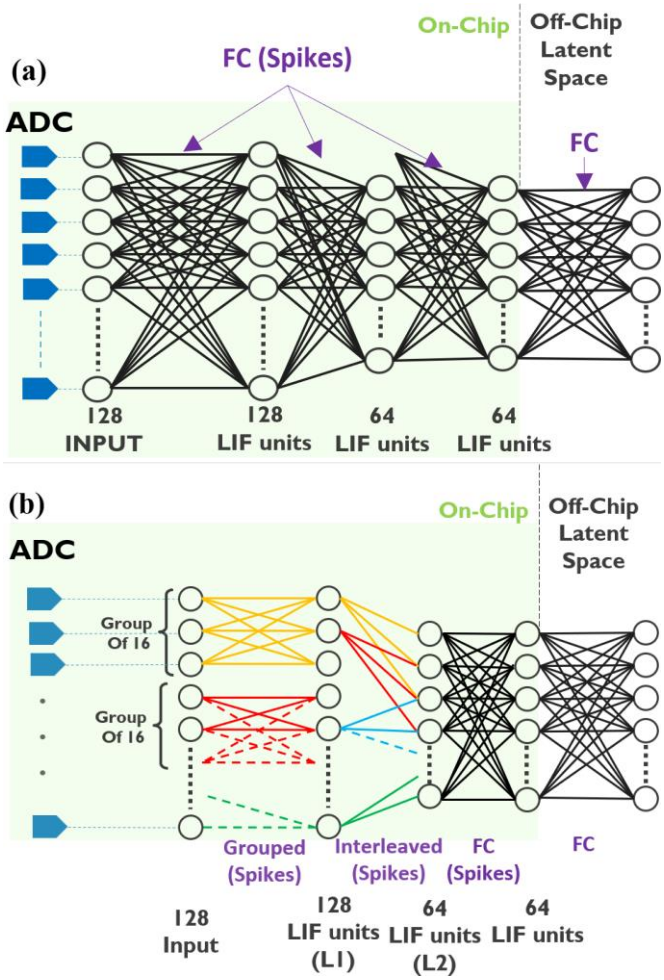
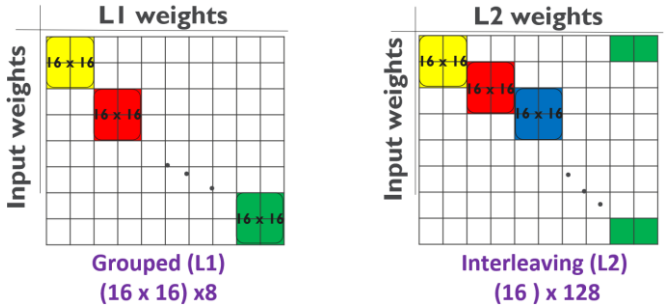



**Figure 7:** (a) Fully Connected Network (b) Pruned Network

As illustrated in Fig. 7, we have employed various strategies for pruning the network. The Δ-ADC inputs are directly mapped to the input layer, with the weights

connecting to the first hidden layer (L1) pruned in a manner that mirrors the Δ-ADC's information processing. This process is organized into eight groups of 16 electrodes each. Within each group, the 16 electrodes are closely correlated due to their spatial proximity, as depicted in Fig. 8(a), which shows the grouping from the input layer to the first hidden layer (L1).




(a) (b)

**Figure 8:** (a) Weight matrix input L1 (b) Weight matrix L2

From the second layer onward, we employ a strategic interleaving of connections to enhance the network's ability to identify patterns across different electrodes. Specifically, each neuron in the second layer is connected to the 16 nearest neurons in the subsequent layer. For instance, the first neuron in the second layer is connected to neurons 1 through 16 in the following layer, while the second neuron is connected to neurons 8 through 24, and so forth. Fig. 8 illustrates this interleaving process, showing how the synaptic weights are organized in a 2×2 matrix, clearly demonstrating the interleaving pattern. This method enhances the network's ability to discern meaningful relationships and improves overall performance by ensuring a more comprehensive coverage of input patterns.

## Metrics and Performance Evaluation

To thoroughly evaluate the performance of the proposed neuromorphic network architecture, we employ six key metrics: Compression Ratio (CR), Signal-to-Noise and Distortion Ratio (SNDR), Root Mean Square Error (RMSE), Memory usage per Channel (bits/ch), Multiply and Accumulate Operations (MAC), and Memory Access. These metrics provide a comprehensive quantitative basis for assessing both the computational efficiency and the signal reconstruction accuracy of the network. It is important to note, however, that absolute power consumption and computation latency are platform-dependent, making it challenging to directly compare these aspects with previous works. Therefore, our focus remains on metrics that offer a fair and consistent evaluation across different platforms.

### *4.1 Compression Ratio (CR):*

Since neuromorphic systems process data with sparse events, the Compression Ratio (CR) in our work is clearly defined here by considering the difference between conventional Nyquist sampling and event-based sampling. It is defined in two stages: first, the compression ratio from the original input data (which is equivalent to be sampled by the conventional Nyquist ADC without delta modulation) to the output of the Δ-ADC, and second, the compression ratio from the Δ-ADC output to the final compressed data output from the SNN encoder.

**CR from Δ-ADC ($CR_{\Delta ADC}$):** This is the ratio of the original data rate, represented by the bits generated based on conventional Nyquist sampling without delta modulation, to the event rate after it has been converted by the Δ-ADC. This step includes the level-crossing event generation and the encoding of address event representation (AER). Even with the overhead from the AER protocol, the original data volume (assuming from Nyquest ADC) can still be effectively reduced by focusing only the changes (i.e., delta) in the signal [17].

$$CR_{\Delta ADC} = \frac{Input\ Data\ Rate\ (Nyquist\ ADC)}{Output\ Data\ Rate\ (\Delta \mathrm{ADC})} \quad (6)$$

**CR from SNN encoder ($CR_{SNN}$):** This ratio quantifies the compression achieved by the SNN encoder itself, defined as the ratio of the event rate after the Δ-ADC (which serves as the input to the SNN encoder) to the final compressed data rate produced by the SNN encoder, i.e., autoencoder's latent space.

$$CR_{SNN} = \frac{Input\ Data\ Rate\ (\Delta \mathrm{ADC})}{Output\ Data\ Rate\ (SNN)} \quad (7)$$

The overall Compression Ratio (CR) for the entire system can be expressed as the product of these two ratios as given by Equation. (8).

$$CR = CR_{\Delta ADC} \times CR_{SNN} \quad (8)$$

4.2 Signal-to-Noise and Distortion Ratio (SNDR):

Signal-to-Noise and Distortion Ratio (SNDR) is a critical metric used to evaluate the accuracy of signal reconstruction within neural network architectures, particularly in the context of neuromorphic systems. SNDR quantifies how closely the reconstructed signal matches the original input signal by comparing the strength of the original signal to the magnitude of the reconstruction error. This ratio provides a robust measure of signal fidelity, indicating how well the network preserves the essential features of the input data during the compression and reconstruction processes.

The higher the SNDR value, the more accurately the network has reconstructed the original signal, with lower levels of noise and distortion. It is defined as:

$$SNDR = 20 \times \log_{10}\left(\frac{||X||_2}{||X - \hat{X}||_2}\right) \quad (9)$$

where:

- $X$ is the original input signal,
- $\hat{X}$ is the reconstructed output signal,
- $||.||_2$ denotes the L2 norm, calculated as $||X||_2 = \sqrt{\sum_{i=1}^{n} {X_i}^2}$ for the original signal and $||X - \hat{X}||_2 = \sqrt{\sum_{i=1}^{n} ({X_i}^2 - {\hat{X}_i}^2)^2}$ for the error signal.

In our specific implementation, SNDR measures the similarity between the cumulative delta events obtained from the input and output of the decoder, providing a meaningful indication of signal fidelity. Unlike other works where the comparison is made directly on the waveform, our approach processes delta events as inputs, making direct waveform comparison impractical. Therefore, we consider SNDR calculated from the cumulative events to be a fair and appropriate metric for evaluating the signal reconstruction accuracy in our system. This approach ensures that we maintain a consistent and reliable measure of how well our network performs in preserving the integrity of the original signal throughout the compression and reconstruction stages.

### *4.3 Root Mean Squared Error (RMSE):*

Root Mean Square Error (RMSE) is a widely used metric for evaluating the accuracy of predictions made by neural networks, particularly during the training phase. RMSE quantifies the difference between the values predicted by the model and the actual values from the dataset, providing a clear measure of the model's prediction accuracy. Specifically, RMSE calculates the square root of the average squared differences between the predicted and actual values, thereby penalizing larger errors more heavily. This makes RMSE a sensitive and effective metric for identifying how well the model is learning during training. It is mathematically is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} \quad (10)$$

where:

- $n$ is the total number of delta modulated events,
- $y_i$ represents the actual value at the iii-th data point, and
- $\hat{y_i}$ represents the predicted or reconstructed value at the *i-th* data point.

In our work, RMSE was primarily utilized during the evaluation of the model in the training phase. This metric provided a fair and reliable measure of model performance as we iteratively refined the network's parameters. By focusing on RMSE during training, we were able to build a strong foundation for the model, ensuring that it learned to accurately reconstruct the input data before moving on to more advanced evaluation metrics.

However, it is important to note that while RMSE is a valuable tool during training, it is not commonly reported in final performance evaluations across different studies. Consequently, we have chosen not to include RMSE in our state-of-the-art comparison Table. 3, as it may not provide a consistent basis for comparison with other works. Instead, we rely on other metrics more commonly used in the literature to benchmark our model's performance against the state of the art.

### *4.4 Memory per Channel (bits/ch):*

Memory per Channel (bits/ch) refers to the number of bits allocated to each channel in the network. This metric is crucial for assessing the memory efficiency of the architecture, as it reflects the amount of data storage required for synapses, membrane potentials, decay parameters, and other channel-specific components to maintain the desired performance.

This metric allows for a fair comparison with other state-of-the-art methods, such as those by Biedermann[21] and Wu[15]. It provides an indication of the buffering required for specific data processing steps, such as spike alignment or detection. Additionally, normalizing the total bits per channel offers a more universal metric, enabling comparison across works with varying numbers of channels.

It is important to note that memory access significantly influences overall power consumption, while the memory footprint predominantly determines the total chip area [33]. These factors make Memory per Channel a critical consideration in the design of neuromorphic systems, especially for applications where power efficiency and chip area are of paramount importance.

### *4.5 Multiply and Accumulate Operations (MAC):*

In the context of Spiking Neural Networks (SNNs), the estimation of computational efficiency is crucial for understanding the network's performance, especially when

compared to traditional artificial neural networks (ANNs). One key metric for evaluating this efficiency is the number of effective Multiply-Accumulate (MAC) operations. The effective MACs in an SNN can be described as the sum of the operations required for both inter-layer connections and intra-neuron (decay parameter) dynamics.

The first component of the effective MAC metric quantifies the operations required to propagate spikes between layers. In a network with $L$ layers, where the spiking rate of layer $l$ is denoted as $r_l$ , the number of neurons in the *lth* layer as $d_l$, and the number of neurons in the preceding layer as $d_{l-1}$, the inter-layer MACs are given by:

$$MAC_{inter-layer} = \sum_{l=1}^{L} (r_l \times d_{l-1} \times d_l) \quad (11)$$

Secondly, each neuron in an SNN typically has additional operations related to its internal dynamics, such as membrane potential decay and bias accumulation. These operations can be quantified as:

$$MAC_{intra-neuron} = \sum_{l=1}^{L} ((decay + bias_l) \times d_l) \quad (12)$$

In this context, *decay* represents the computational cost of maintaining the membrane potential dynamics, and $bias_l$ represents the accumulation of bias terms in the neurons of layer *l*. Therefore now the total effective MAC operations for the SNN can thus be expressed as:

$$Total\ MACs = MACs_{inter-layer} + MACs_{intra-neuron} \quad (13)$$

This metric inspired from [34] provides a comprehensive measure of the computational cost associated with the network, taking into account the sparsity of spikes, the presence of recurrent connections, and the intrinsic dynamics of spiking neurons. However, as we utilize a spiking neural network in our work, a more accurate term would be "Accumulate" (ACC) operations instead of Multiply-Accumulate (MAC) operations, reflecting the event-driven nature of our SNN. In general state-of-the-art comparisons, the metric remains a MAC, but in the context of our specific implementation, it is better referred to as ACC. This ensures that the metric remains general enough to benchmark against other state-of-the-art methods while accurately representing the computational efficiency in our neuromorphic system.

### *4.6 Memory Access:*

Memory Access (MA) is a critical metric that quantifies the rate at which memory is accessed during the operation of the network. This metric is essential for a comprehensive estimation of the network's energy consumption, as memory read-and-write operations typically require one to two orders of magnitude more energy than arithmetic operations performed by linear units [31][32].

Given that most of the energy consumption in neural architectures stems from these memory operations, simply reporting the effective arithmetic operations without including memory access would provide an incomplete assessment of the network's efficienc. For instance, it has been reported in [34] that during inference, there can be approximately 10 times more memory reads than effective operations, emphasizing the significant energy impact of memory access.

In line with previous studies [34], we conservatively estimate the number of memory accesses by assuming that each MAC operation involves three memory loads and one store (this includes the memory read for bias since we don't use one in our student network) and each Accumulate (ACC) involves two memory loads and one store (since in our netwokr we don't have a bias). This estimate is further adjusted to account for the sparseness of network activity, providing a realistic measure of memory access in the system. The total number of memory accesses can be expressed as:

$$MA = (3\ Loads + 1\ Store)\ MACs + (2\ Loads + 1\ Store)\ ACCs \quad (14)$$

This equation ensures that the evaluation of the network's energy efficiency is both accurate and comprehensive by accounting for the energy impact of memory operations relative to computational operations. In our work, we exclusively utilize ACC operations, resulting in a total of three memory operations. In contrast, other state-of-the-art methods that rely on MAC operations require four memory operations in total. This difference highlights the inherent efficiency of our approach in minimizing memory access, which is a significant contributor to overall energy consumption.

## Results and Experiments

In this section, we present the main results of our Autoencoder System, followed by detailed ablation studies that explore the influence of different teacher networks, network sizes, and configurations.

| **Reference** | He [17] | Biederman [21] | Zhang [16] | Wu [15] | **This Work** |
|---|---|---|---|---|---|
| **Publication** | TBioCAS'24 | JSSC'15 | IOP'15 | IOP'18 | - |
| **No. of channels** | 128 | 64 | 4 | 256 | 128 |
| **Encoder Type** | Delta mod. | Epoch | Compresssed Sensing | CNN | Delta mod. + SNN |
| **Compression Ratio (CR)** | 10 × | 8 × | 8 × - 16 × | 20 × | **94 ×** |
| **Peak SNDR** | 36 dB | >50 dB | ~13 dB | 10 – 13 dB | **22 dB** |
| **Memory (bit /ch)** | 27 | 1500 | - | 576[A] | **384** |
| **Computations (MACs/spike)** | - | - | - | 79.25K | 18.07K |
| | | | | 309[B] | **141[B]** |
| **Memory Access per spike** | - | - | - | 317K | 54.21K |
| | | | | 1238[B] | **423[B]** |

[A]This number excludes the additional data buffer required for spike detection and alignment, which requires extra ~1000b/channel of memory for 3ms of spike data.
[B] This number has been normalized to number of channel.

**Table 1:** Summarized performace and comparsion to State-of-the-Art (SoTA) neural compression systems

These experiments were conducted using multi-electrode recordings from two datasets, each comprising 30 minutes of data from different mice. The first dataset, containing hippocampal recordings, was obtained from University College London (UCL) [37], and the second, consisting of cerebellar recordings, was sourced from Erasmus Medical Center [38]. To ensure robustness and generalizability across various conditions and subjects, we refer to these datasets as hippo_dat1 and cere_dat2, respectively.

### *5.1 Performance Summary and State-of-the-Art (SoTA) Comparison*

We compared various state-of-the-art compression techniques, as outlined in Section 2, against the metrics defined in Section 4 to obtain a comprehensive assessment of our system's efficiency.

As shown in Table. 1, the proposed compression system achieves superior compression, 4.7× better than the closest existing work [15], while demonstrating 9-12 dB better reconstruction quality in SNDR. This significant improvement is attributed to the combination of the delta modulation, the event-based neuromorphic encoder, and the proposed two-stage teacher-student network training. It also requires only approximately 5× less memory usage compared to [21], which is attributed to buffer-less real-time processing of neuromorphic encoders and meticulous pruning techniques implemented in this work.

Additionally, the use of the neuromorphic encoder enabled us to minimize multiplication operations, reducing the effective MAC operations by nearly 4× compared to the closest neural network-based compression systems. This reduction also led to a corresponding decrease in the number of memory read and write operations by the same factor, further enhancing the overall efficiency of the proposed compression system. Fig. 9 illustrates a detailed visual comparison between the cumulative waveforms of an original input signal from the Δ-ADC and its corresponding reconstructed signal our Neuromorphic Autoencoder for the *hippo_dat1* and *cere_dat2* datasets. The original waveform (blue) and the reconstructed waveform (orange) closely align, indicating the decoder's high fidelity in preserving the overall signal trend.

It is important to note that the figures presented in Table. 1 were derived from the *hippo_dat1* dataset [37], which is publicly available and can be reproduced if needed, ensuring the robustness and reproducibility of the presented results.

### *5.2 Influence of Teacher Network Architectures on Performance*

To optimize the training of our spiking-based student network, we conducted an ablation study on various teacher network architectures using the *hippo_dat1* dataset. We evaluated Spiking Neural Networks (SNN), Convolutional Neural Networks (CNN), Spiking Convolutional Neural Networks (SCNN), Long Short-Term Memory networks (LSTM), Spiking Long Short-Term Memory networks

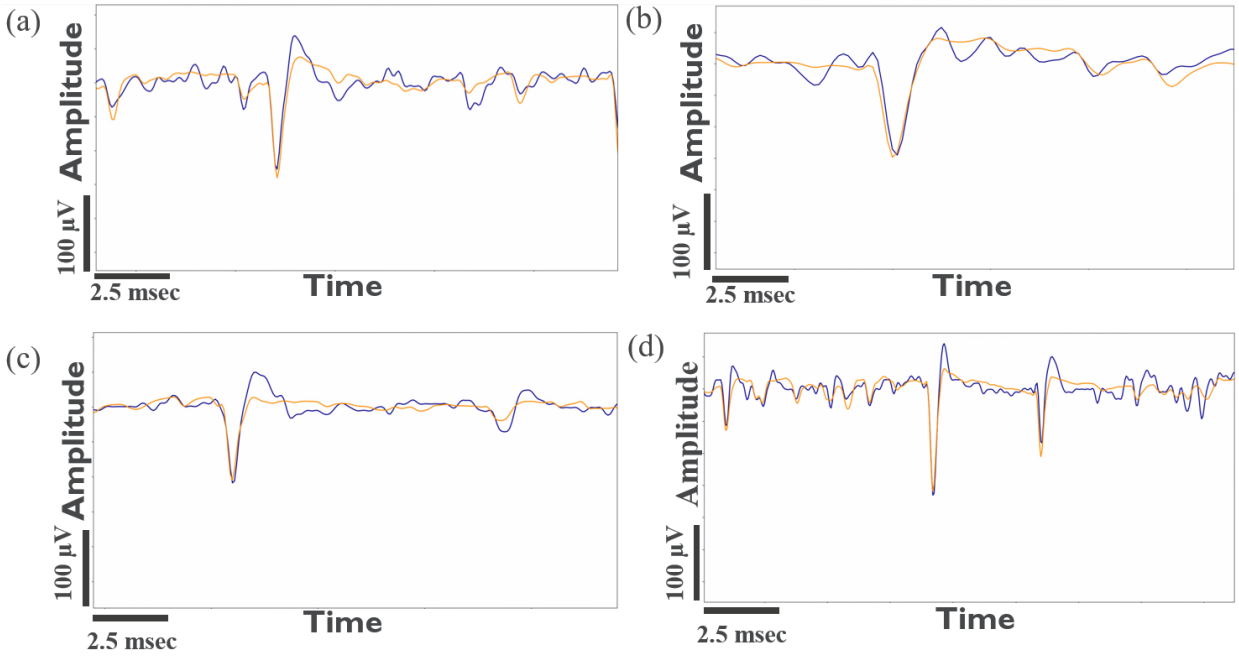



**Figure 9:** Comparison of Original (blue) and Reconstructed Waveforms (orange) for different spikes in the *hippo_dat1* ((a) & (b)) and *cere_dat2* ((c) & (d)) from our Neuromorphic Autoencoder Compression System.

(SLSTM), Recurrent Neural Networks (RNN), and Spiking Recurrent Neural Networks (SRNN).

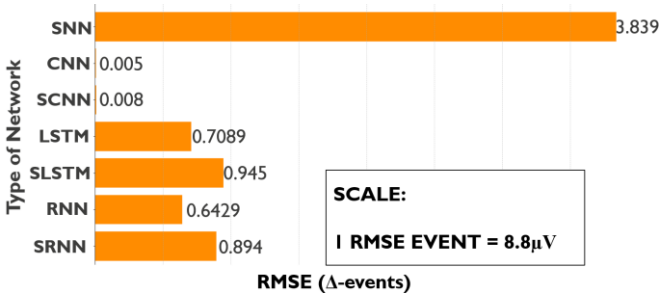



**Figure 10:** Comparative performance of teacher architectures

As illustrated in Fig. 10, other network architectures outperformed the SNN, evidenced by lower Root Mean Squared Error (RMSE) loss during training. This observation underscores the critical importance of knowledge distillation in training spiking-based student networks effectively.

Our findings show that non-spiking architectures generally outperform spiking ones, with CNNs achieving the best performance in terms of training loss. However, when training the student network using the two-stage encoder-decoder method described earlier in Section 3.3.1, RNN and LSTM models more effectively preserved waveform fidelity, despite the CNN's favorable compression ratios (as illustrated in Fig. 11). The spiking variants performed significantly worse than the architectures shown in Fig. 10, primarily due to inherent limitations such as lower frequency clocks, which may not fully capture the temporal dynamics encoded in spikes[13].

These results indicate that LSTM and RNN architectures are more effective as teacher models for training spiking neural networks, given their ability to handle temporal sequences that align better with the spiking network's requirements. For the remainder of this paper, we work with RNN as the teacher network.

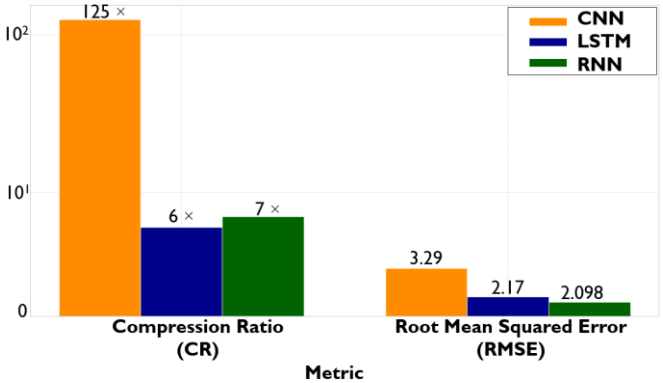



**Figure 11:** Comparison of student network compression ratios and accuracy

### *5.3 Impact of Teacher Network Depth*

We conducted an ablation study to determine the optimal number of layers for the RNN-based teacher network, aiming to balance model complexity and accuracy. As illustrated in

Fig. 12, we evaluated various layer configurations across the *hippo_dat1* and *cere_dat2* datasets.

The results reveal a clear trend: increasing the number of layers beyond three leads to a significant increase in Root Mean Square Error (RMSE) for both datasets. This observation suggests that adding more layers increases the risk of overfitting, where the model becomes too tailored to the training data and loses its ability to generalize effectively to new data. Specifically, the RMSE sharply rises with five or more layers, indicating that excessive complexity hampers the network's performance rather than enhancing it, consistent with findings in related studies [39].

Conversely, a configuration with three or four layers consistently delivers robust performance with minimal RMSE, striking the optimal balance between complexity and accuracy. These configurations provide sufficient depth to capture the underlying patterns effectively without the drawbacks of overfitting.

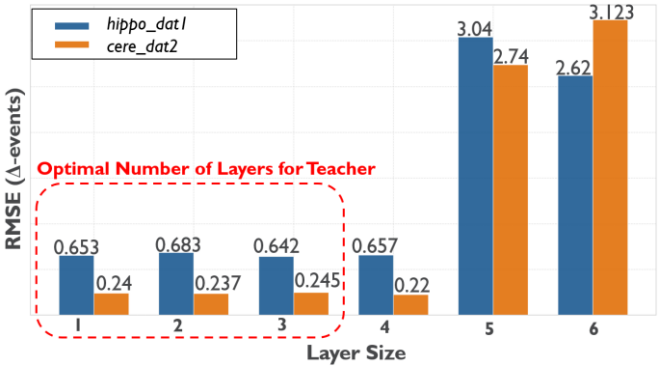



**Figure 12:** Comparison of teacher RNN network with different layer sizes (1 RMSE Event = 8.8µV)

However, reducing the layer count below three may introduce the risk of the spiking student model's ability to generalize. As shown in the lower RMSE values for the two-layer configuration compared to a single layer, the model's capacity to capture complex temporal dynamics improves with additional layers, which is crucial for the effective training of recurrent models[40]. This suggests that a three- or four-layer teacher network offers a solid foundation for training, effectively balancing generalization and performance. While these findings provide valuable guidance, further analysis is necessary to conclusively determine the impact of reducing layers on the student network's performance as will be elaborated in subsequent sections.

### *5.4 Optimization of Latent Space Dimensionality*

To further enhance the performance of our Recurrent Neural Networks (RNNs) as a teacher model, we conducted an analysis of the impact of different latent space dimensions on the network's ability to capture and generalize from the underlying data. This step is crucial as it informs the selection of an appropriate latent space dimension, which directly affects the model's capacity to represent complex patterns within the datasets.

To ensure consistent and accurate loss calculation during training the student, particularly when computing the latent loss during training (as explained in *section 3.4.1*), an additional layer of 64 units (off chip layer as shown in Fig. 7) is included as the final layer. However, this last layer is not considered part of the spiking layers; instead, it functions as an RNN cell. This design choice is critical for maintaining a seamless alignment between the latent representations of the student and teacher networks, thereby facilitating effective knowledge transfer through knowledge distillation.

The initial spiking layers, with their 128 and 64 units respectively (as shown in Fig. 7) , provide sufficient depth and capacity to encode complex features while maintaining computational efficiency. The inclusion of the RNN cell at the end serves to refine the output, ensuring that the latent space is accurately captured and utilized in the loss calculation. This structure not only enhances the model's performance but also ensures that the training process remains robust and aligned with the objectives of efficient and accurate data compression within the autoencoder framework.

As illustrated in Fig. 13, we evaluated the RMSE performance across various latent space dimensions—specifically, 8, 16, 32, 64 and 128—using both the *hippo_dat1* and *cere_dat2* datasets. The results demonstrate a clear downward trend in RMSE as the latent dimension increases, indicating that a larger latent space allows the model to better encode the intricate relationships present in the data.

For the *hippo_dat1* Dataset, the RMSE decreases sharply from around 3.0 with an 8-dimensional latent space to 0.65 with a 64-dimensional latent space. Similarly, the *cere_dat2* Dataset exhibits a reduction in RMSE from 2.45 to around 0.22 across the same range. This significant improvement underscores the importance of a higher-dimensional latent space in providing the necessary capacity for the RNN to accurately model the complexities of the datasets.

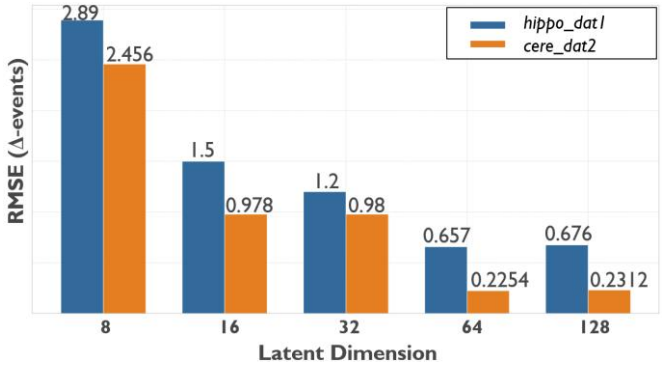



**Figure 13:** Comparison of teacher RNN network with different latent dimensions (1 RMSE Event = 8.8µV)

Notably, while the RMSE values for both datasets converge at the higher latent dimensions, the *cere_dat2* dataset consistently achieves lower RMSE values, particularly at the larger latent space sizes. This difference suggests that the *cere_dat2* dataset may exhibit more straightforward patterns, which can be efficiently captured even as model complexity increases. In contrast, the *hippo_dat1* dataset appears to benefit more significantly from an expanded latent space, likely due to its more complex signal characteristics.

Moreover, the superior performance of the *cere_dat2* dataset compared to the *hippo_dat1* dataset at this stage might be attributed to differences in data collection methods. Recordings from the cerebellum are characterized by a higher firing rate compared to those from the hippocampus, providing more frequent data points from the cerebellum [41]. Additionally, hippocampal recordings typically have larger signal amplitudes, as the electrodes are placed deeper within the brain. This depth results in more significant RMSE errors due to inherent amplitude reductions in the reconstructed waveforms[42].

Based on these findings, a 64-dimensional latent space is justified as the optimal choice for the teacher network within the context of our study. This dimension provides a robust capacity for the model to learn and generalize effectively across both datasets, minimizing RMSE and thereby improving the accuracy of the model's predictions. However, it is worth noting that while a 64-dimensional latent space offers the best performance in this scenario, it also introduces additional computational complexity.

### *5.5 Evaluation of Student Network Depth*

Building upon the insights from our previous findings, we further examined the impact of varying the number of layers (hidden layers) in the student network, focusing specifically on configurations ranging from one to three layers. As shown in Fig. 14, the RMSE values decrease progressively as the number of layers increases from one to three across both the *hippo_dat1* and *cere_dat2* datasets for the student spiking network.

For the *hippo_dat1* Dataset, the RMSE decreases from 2.69 with a single layer to 2.01 with three layers, while for the *cere_dat2* Dataset, the RMSE reduces from 1.90 to 1.543 over the same range. This reduction in RMSE highlights the importance of deeper spiking network architectures, at least within this tested range, in effectively capturing the intricate patterns present in the data.

Notably, while both datasets exhibit a similar downward trend, the RMSE values for the *cere_dat2* Dataset consistently remain lower than those for the *hippo_dat1* Dataset across all tested layer sizes. This difference likely reflects the varying characteristics and complexities inherent in the two datasets. The *cere_dat2* Dataset appears to be "cleaner," with patterns that are more readily captured by the RNN, even with fewer layers, whereas the *hippo_dat1* Dataset may require a more complex model to achieve comparable accuracy.

These findings reinforce the conclusion that a three-layer configuration (hidden layers) for the student network is a reasonable choice for our task, striking a balance between reconstruction fidelity and model complexity. This depth provides enough capacity to effectively capture the nuances of the training data without introducing unnecessary complexity.

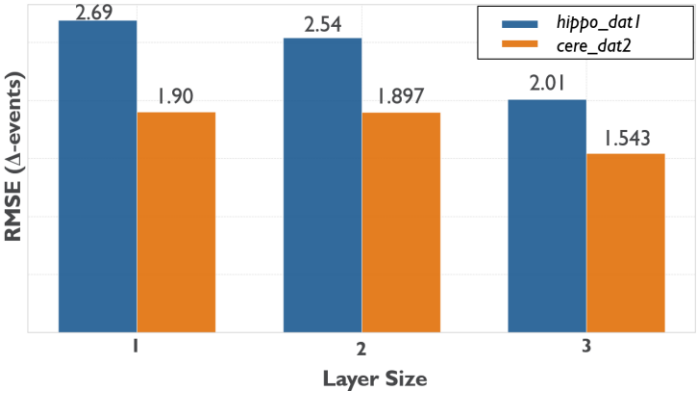



**Figure 14:** Comparison of student spiking network with different layer sizes (1 RMSE Event = 8.8µV)

However, it is not the only viable option; depending on specific application constraints, a one- or two-layer structure might be preferred, trading off a slight increase in RMSE for reduced model complexity and lower computational overhead.

While a three-layer (hidden) structure appears to deliver fair reconstruction performance, the choice of network depth

should ultimately align with the broader goals and limitations of the particular application.

### *5.6 Pruned Student Model Configuration*

Following extensive experimentation, we identified a three hidden layer student spiking neural network with a 64-dimensional latent space as the most suitable configuration. This model, detailed in Table 2, balances memory efficiency and accuracy effectively.

As discussed in *Section 3.4.2*, we employed input grouping and interleaving techniques to reduce hardware complexity, gradually pruning the network layer by layer, starting from the fully connected configuration as shown in Table 2.

| Connection Type | | | Network No. |
|---|---|---|---|
| **Input Layer** | **2nd Hidden Layer** | **3rd Hidden Layer** | |
| Fully Connected | Fully Connected | Fully Connected | 1 |
| Grouped | Fully Connected | Fully Connected | 2 |
| Grouped | Interleaved | Fully Connected | 3 |
| Grouped | Interleaved | Interleaved | 4 |

**Table 2:** Types of student network connection explored for hardware aware optimization

This approach resulted in a nearly 7× reduction in memory footprint per channel from the fully connected model to the final pruned model, as seen in Table 3. However, this reduction came with a trade-off in RMSE values, which gradually increased due to information loss during pruning. A similar effect was observed in the decreasing SNDR values.

.

| Data | Network No. | Memory (kbit/ch) | $CR_{SNN}$ | RMSE | SNDR (dB) |
|---|---|---|---|---|---|
| *hippo_dat1* | 1 | 2.55 | 5.48 | 2.01 | 24.47 |
| | 2 | 1.64 | 7.63 | 2.137 | 23.94 |
| | 3 | 0.384 | 9.4 | 2.185 | 23.75 |
| | 4 | 0.24 | 19.7 | 6.217 | 10.67 |
| *cere_dat2* | 1 | 2.55 | 6.12 | 1.543 | 24.19 |
| | 2 | 1.64 | 7.98 | 1.78 | 22.95 |
| | 3 | 0.384 | 10.2 | 1.83 | 22.7 |
| | 4 | 0.24 | 12.9 | 5.667 | 10.94 |

**Table 3:** Hardware aware student analysis - bits used vs compression ratio vs reconstruction fidelity on different datasets.

To strike the best balance between memory efficiency and accuracy, we identified Network No. 3 as the optimal model. In this configuration, pruning was applied only up to the penultimate spiking layer, while the final spiking layer remained fully connected. This approach minimized the trade-off between memory usage and accuracy.

### *5.7 Quantization*

In the previous section, we identified Network No. 3 as the optimal model due to its balanced trade-off between memory efficiency and accuracy. Building on this, we proceeded to further optimize this model through quantization. In the process of optimizing our neuromorphic encoder model, we employed symmetric quantization to reduce the precision of weights and activations from floating-point 32 (FP32) to integer formats (INT8, INT6, and INT4).

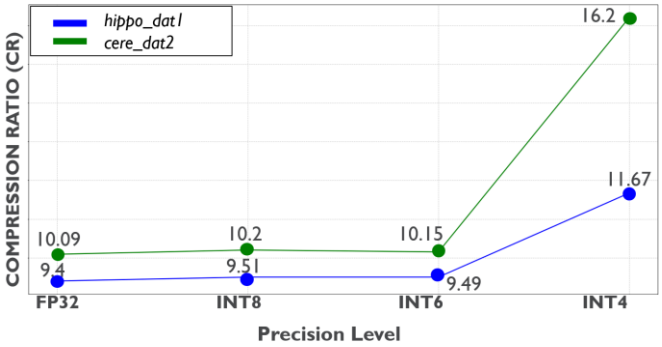



**Figure 15:** Compression Ratio ($CR_{SNN}$) Across Different Precision Levels for the hippo_dat1 and cere_dat2 datasets.

Symmetric quantization was chosen for its ability to maintain a balanced range of positive and negative values around zero, which is particularly effective for the types of data encountered in our model [43]. This approach led similar compression ratio as in FP32, as highlighted in Fig. 15. However, as depicted in Fig. 16, a reduction in precision increased RMSE, particularly at the INT4 level, indicating a trade-off between compression and signal reconstruction accuracy

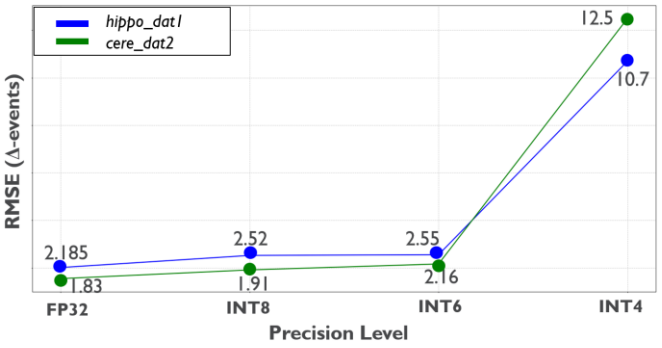



**Figure 16:** RMSE Across Different Precision Levels for the hippo_dat1 and cere_dat2 datasets. (1 RMSE Event = 8.8µV)

To further enhance the efficiency of our quantization process, we integrated a technique known as BatchNorm folding [43]. BatchNorm layers, which are typically used during training to stabilize and accelerate convergence, were folded into the preceding convolutional or fully connected layers during the inference phase. This folding process

involves incorporating the BatchNorm parameters (mean and variance) into the weights and biases of the corresponding layers, effectively reducing the computational overhead and memory requirements during inference. By doing so, we were able to maintain the benefits of BatchNorm during training while simplifying the model for deployment on resource-constrained hardware.

This combination of symmetric quantization and BatchNorm folding not only improved the overall memory efficiency of our model giving almost a 1.5 × fold reduction in memory (bit/ch) but also ensured that it remained compact and suitable for real-time applications in neuromorphic systems [44].

## Conclusion

This paper presents a significant advancement in neuromorphic systems, particularly for intra-cortical Brain-Computer Interfaces (iBCIs), by introducing a novel compression architecture that integrates delta modulation with Spiking Neural Networks (SNNs). The proposed system achieves a noteworthy compression ratio of >94×, far surpassing existing state-of-the-art methods, and effectively addresses the critical challenges of data transmission and storage in high-density neural recording environments.

Our experimental results, validated on hippocampal and cerebellar datasets [37], [38], demonstrate the superiority of our approach in terms of compression efficiency, Signal-to-Noise and Distortion Ratio (SNDR), and memory usage compared to existing state-of-the-art techniques[15][20][25]. Notably, our system not only achieves a peak SNDR of 24 dB but also significantly reduces memory usage to 384 bits per channel, with computations per spike minimized to 18.07K MACs and memory access per spike reduced to 54.21K—highlighting its efficiency for resource-constrained implantable devices.

The ability to drastically reduce data throughput while preserving the essential characteristics of neural signals underscores the potential of SNNs in creating highly efficient and scalable neuromorphic systems. Moreover, the low power consumption and minimal heat generation of the system make it particularly well-suited for implantable devices, where resource constraints are a primary concern.

The effectiveness of this work lies in its innovative application of neuromorphic principles, leveraging the event-driven nature of SNNs to achieve both high compression and energy efficiency. Moreover the training method used The hardware-aware optimization techniques, including pruning and quantization, further enhance the system's applicability in real-world, resource-limited settings. These techniques enabled the reduction of synaptic weights and the associated memory requirements, contributing to the overall compactness and efficiency of the system.

## Discussions

Our system addresses several key bottlenecks encountered by existing state-of-the-art methods. Zhang's et.al approach [16] faces challenges with the complexity of optimizing signal reconstruction, as evidenced by its low SNDR value, which is nearly 2.5 times lower than that achieved by our method. Furthermore, Zhang's method requires additional buffer memory to store the sampling and transformation matrices, which complicates its implementation. Similarly, the methods proposed by Biedermann et.al [21] and Wu et.al [15] also necessitate extra buffer memory, as they rely on spike detection and alignment, requiring nearly 1500b per channel of memory for 3 ms of spike data. Wu's study is the closest to our work, given its use of neural networks; however, it is burdened by the large number of parameters inherent in CNNs. Our approach, which employs an event-based neuromorphic design, significantly reduces memory and computation costs by nearly twofold, while also achieving a signal fidelity that is approximately 10 dB higher than Wu's work.

However, the Fig. 9(d) we can observe minor amplitude shrinkages in the reconstructed waveform, which may occur due to the use of a lower frequency clock, potentially failing to fully capture the temporal dynamics encoded in the spikes [13]. These discrepancies are minimal and occur at certain critical points, as shown in the boxed areas. Despite these small variations, which are further quantified through the RMSE analysis earlier, the decoder effectively replicates the signal with high accuracy. The identification and visualization of these discrepancies underscore the robust performance of the decoder while also acknowledging the areas where slight deviations occur, thereby providing a comprehensive evaluation of the signal reconstruction process.

Despite these advancements, there remain opportunities for further research. Future studies could explore the real-time implementation of our system and assess its performance in dynamic environments. Additionally, applying this compression architecture to other domains within neuromorphic computing may yield further insights and enhancements.

The neuromorphic compression system presented in this paper establishes a new benchmark for neural signal processing in intra-cortical Brain-Computer Interfaces (iBCIs). It offers a scalable and energy-efficient solution that not only meets the current demands of high-density neural data processing but also paves the way for future

advancements in neural prosthetics and neuroengineering. The findings and methods developed here represent a critical step toward the practical realization of advanced Brain-Computer Interface technologies, making a substantial contribution to the field.

## References


[1] H. Hosseini-Nejad, A. Jannesari, and A. M. Sodagar, "Data Compression in Brain-Machine/Computer Interfaces Based on the Walsh–Hadamard Transform," *Biomed. Circuits Syst. IEEE Trans. On*, vol. 8, pp. 129–137, Feb. 2014, doi: 10.1109/TBCAS.2013.2258669.

[2] "Brain Computer Interfaces in Rehabilitation Medicine - Bockbrader - 2018 - PM&R - Wiley Online Library." Accessed: Aug. 07, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1016/j.pmrj.2018.05.028

[3] I. H. Stevenson and K. P. Kording, "How advances in neural recording affect data analysis," *Nat. Neurosci.*, vol. 14, no. 2, pp. 139–142, Feb. 2011, doi: 10.1038/nn.2731.

[4] J. Navajas, D. Y. Barsakcioglu, A. Eftekhar, A. Jackson, T. G. Constandinou, and R. Q. Quiroga, "Minimum Requirements for Accurate and Efficient Real-Time On-Chip Spike Sorting," *J. Neurosci. Methods*, vol. 230, pp. 51–64, Jun. 2014, doi: 10.1016/j.jneumeth.2014.04.018.

[5] N. Even-Chen *et al.*, "Power-saving design opportunities for wireless intracortical brain computer interfaces," *Nat. Biomed. Eng.*, vol. 4, no. 10, pp. 984–996, Oct. 2020, doi: 10.1038/s41551-020-0595-9.

[6] "Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings | Science." Accessed: Aug. 19, 2024. [Online]. Available: https://www.science.org/doi/10.1126/science.abf4588

[7] G. Indiveri and S.-C. Liu, "Memory and Information Processing in Neuromorphic Systems," *Proc. IEEE*, vol. 103, no. 8, pp. 1379–1397, Aug. 2015, doi: 10.1109/JPROC.2015.2444094.

[8] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990, doi: 10.1109/5.58356.

[9] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128\times 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008, doi: 10.1109/JSSC.2007.914337.

[10] P. D. Wolf, "Thermal Considerations for the Design of an Implanted Cortical Brain–Machine Interface (BMI)," in *Indwelling Neural Implants: Strategies for Contending with the In Vivo Environment*, W. M. Reichert, Ed., in Frontiers in Neuroengineering. , Boca Raton (FL): CRC Press/Taylor & Francis, 2008. Accessed: Aug. 23, 2024. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK3932/

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[12] P. Hueber *et al.*, *Benchmarking of Hardware-efficient Real-time Neural Decoding in Brain-computer Interfaces*. 2023. doi: 10.36227/techrxiv.24523291.v2.

[13] M. Pfeiffer and T. Pfeil, "Deep Learning With Spiking Neurons: Opportunities and Challenges," *Front. Neurosci.*, vol. 12, Oct. 2018, doi: 10.3389/fnins.2018.00774.

[14] M. Davies *et al.*, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018, doi: 10.1109/MM.2018.112130359.

[15] T. Wu, W. Zhao, E. Keefer, and Z. Yang, "Deep compressive autoencoder for action potential compression in large-scale neural recording," *J. Neural Eng.*, vol. 15, no. 6, p. 066019, Dec. 2018, doi: 10.1088/1741-2552/aae18d.

[16] J. Zhang *et al.*, "A closed-loop compressive-sensing-based neural recording system," *J. Neural Eng.*, vol. 12, no. 3, Art. no. 3, Apr. 2015, Accessed: Aug. 08, 2024. [Online]. Available: https://eprints.gla.ac.uk/117586/

[17] Y. He *et al.*, "An Event-based Neural Compressive Telemetry with >11× Loss-less Data Reduction for High-bandwidth Intracortical Brain Computer Interfaces," *IEEE Trans. Biomed. Circuits Syst.*, vol. PP, Mar. 2024, doi: 10.1109/TBCAS.2024.3378973.

[18] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, Dec. 1997, doi: 10.1016/S0893-6080(97)00011-7.

[19] G. Buzsáki, *Rhythms of the Brain*. Oxford University Press, 2006. doi: 10.1093/acprof:oso/9780195301069.001.0001.

[20] S. Makeig, S. Debener, J. Onton, and A. Delorme, "Mining event-related brain dynamics," *Trends Cogn. Sci.*, vol. 8, pp. 204–10, Jun. 2004, doi: 10.1016/j.tics.2004.03.008.

[21] W. Biederman *et al.*, "A 4.78 mm 2 Fully-Integrated Neuromodulation SoC Combining 64 Acquisition Channels With Digital Compression and Simultaneous Dual Stimulation," *IEEE J. Solid-State Circuits*, vol. 50, no. 4, pp. 1038–1047, Apr. 2015, doi: 10.1109/JSSC.2014.2384736.

[22] D. Valencia, P. P. Mercier, and A. Alimohammad, "Efficient in Vivo Neural Signal Compression Using an Autoencoder-Based Neural Network," *IEEE Trans. Biomed. Circuits Syst.*, vol. 18, no. 3, pp. 691–701, Jun. 2024, doi: 10.1109/TBCAS.2024.3359994.

[23] X. Ran, W. Chen, B. Yvert, and S. Zhang, "A hybrid autoencoder framework of dimensionality reduction for brain-computer interface decoding," *Comput. Biol. Med.*, vol. 148, p. 105871, Sep. 2022, doi: 10.1016/j.compbiomed.2022.105871.

[24] C. Tan, F. Sun, B. Fang, T. Kong, and W. Zhang, "Autoencoder-based transfer learning in brain–computer interface for rehabilitation robot," *Int. J. Adv. Robot.*

*Syst.*, vol. 16, p. 172988141984086, Apr. 2019, doi: 10.1177/1729881419840860.
[25] M. R. Cohen and A. Kohn, "Measuring and interpreting neuronal correlations," *Nat. Neurosci.*, vol. 14, no. 7, pp. 811–819, Jun. 2011, doi: 10.1038/nn.2842.
[26] "Dynamic Power Reduction in Scalable Neural Recording Interface Using Spatiotemporal Correlation and Temporal Sparsity of Neural Signals | IEEE Journals & Magazine | IEEE Xplore." Accessed: Aug. 08, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/8265022
[27] S. Zanos, T. P. Zanos, V. Z. Marmarelis, G. A. Ojemann, and E. E. Fetz, "Relationships between spike-free local field potentials and spike timing in human temporal cortex," *J. Neurophysiol.*, vol. 107, no. 7, pp. 1808–1821, Apr. 2012, doi: 10.1152/jn.00663.2011.
[28] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," Mar. 09, 2015, *arXiv*: arXiv:1503.02531. doi: 10.48550/arXiv.1503.02531.
[29] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going Deeper in Spiking Neural Networks: VGG and Residual Architectures," *Front. Neurosci.*, vol. 13, Mar. 2019, doi: 10.3389/fnins.2019.00095.
[30] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, in ICML '08. New York, NY, USA: Association for Computing Machinery, Jul. 2008, pp. 1096–1103. doi: 10.1145/1390156.1390294.
[31] S. Lall, "How do neurons communicate (so quickly)?," MIT McGovern Institute. Accessed: Aug. 08, 2024. [Online]. Available: https://mcgovern.mit.edu/2019/02/28/ask-the-brain-how-do-neurons-communicate/
[32] "How do neurons work?" Accessed: Aug. 08, 2024. [Online]. Available: https://qbi.uq.edu.au/brain-basics/brain/brain-physiology/how-do-neurons-work
[33] J. Stuijt, M. Sifalakis, A. Yousefzadeh, and F. Corradi, "μBrain: An Event-Driven and Fully Synthesizable Architecture for Spiking Neural Networks," *Front. Neurosci.*, vol. 15, May 2021, doi: 10.3389/fnins.2021.664208.
[34] J. 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)*, Jun. 2022, pp. 134–137. doi: 10.1109/AICAS54282.2022.9869846.
[35] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, *Efficient Processing of Deep Neural Networks*. in Synthesis Lectures on Computer Architecture. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-031-01766-7.
[36] M. 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)*, Feb. 2014, pp. 10–14. doi: 10.1109/ISSCC.2014.6757323.
[37] "Recording with a Neuropixels probe." University College London, Feb. 16, 2024. doi: 10.5522/04/25232962.v2.
[38] "Contact - Department of Neuroscience." Accessed: Aug. 21, 2024. [Online]. Available: https://neuro.nl/contact
[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.
[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.
[41] D. A. McCormick and R. F. Thompson, "Neuronal responses of the rabbit cerebellum during acquisition and performance of a classically conditioned nictitating membrane-eyelid response," *J. Neurosci. Off. J. Soc. Neurosci.*, vol. 4, no. 11, pp. 2811–2822, Nov. 1984, doi: 10.1523/JNEUROSCI.04-11-02811.1984.
[42] G. Buzsáki, "Large-scale recording of neuronal ensembles," *Nat. Neurosci.*, vol. 7, no. 5, pp. 446–451, May 2004, doi: 10.1038/nn1233.
[43] X. Zhao, Y. Wang, X. Cai, C. Liu, and L. Zhang, "Linear Symmetric Quantization of Neural Networks for Low-precision Integer Hardware," presented at the International Conference on Learning Representations, Sep. 2019. Accessed: Aug. 16, 2024. [Online]. Available: https://openreview.net/forum?id=H1lBj2VFPS
[44] S. Liu and Y. Yi, "Quantized Neural Networks and Neuromorphic Computing for Embedded Systems," 2020. doi: 10.5772/intechopen.91835.