Spiking Neural Networks for Nanopore Basecalling: A Sustainable Approach Leveraging BONITO and Neuromorphic Computing

Luca Faieta
Politecnico of Turin
s323770@studenti.polito.it

Stefano Barcio
Politecnico of Turin
s320174@studenti.polito.it

Abstract

Spiking Neural Networks (SNNs) have gained attention as an energy-efficient and biologically inspired alternative to traditional artificial neural networks. Unlike conventional networks (ANN), SNNs process information through discrete, event-driven spikes, enabling sparse computation and reducing power consumption, especially in neuromorphic hardware implementations. This makes SNNs particularly appealing for applications requiring real-time processing or resource-constrained environments. In this work, we explore the application of SNNs to the problem of nanopore sequencing basecalling—a computationally intensive task central to translating raw nanopore signals into DNA sequences. Building on Bonito, the current basecalling state of the art ANN architecture created by Oxford Nanopore Technologies, we propose a novel SNN-based architecture designed to leverage the temporal dynamics of spiking neurons. Our model improves performance by 7% against baselines in reconstructing the squiggle signal: this demonstrates the viability of SNNs for nanopore sequencing but also highlights their potential to redefine the computational landscape of bioinformatics by combining performance with energy efficiency, opening new possibilites for scalable and low-consuming Neural Networks

1. Introduction

Classical neural network (NN) models, including deep learning architectures, have achieved remarkable results across various applications, from image recognition to natural language processing. However, their computational intensity and reliance on large-scale matrix operations result in significant energy demands. When deployed on cloud servers, these models require substantial power not only for computation but also for cooling, driving up energy costs. This issue becomes even more pronounced when deploying these models on mobile, wearable, and edge devices, where resources such as battery life, memory, and processing power are inherently limited. The high energy consumption of traditional NNs not only impacts device longevity but also hampers their viability in real-time or resource-constrained environments. These challenges highlight the need for alternative computational paradigms that balance performance with energy efficiency. Brain-inspired approaches, such as spiking neural networks (SNNs) and neuromorphic hardware, offer a promising solution by emulating biological neural systems to process information asynchronously, sparsely, and with much lower energy consumption. These emerging paradigms aim to reduce computational overhead while maintaining high performance, making them ideal for next-generation edge intelligence and low-power applications.

Spiking Neural Networks (SNNs) represent a paradigm shift in artificial intelligence by offering a biologically inspired approach that mimics how information is processed in the human brain. Unlike traditional artificial neural networks (ANNs), which rely on continuous-valued activations, SNNs use discrete, event-driven spikes to encode and transmit information. This asynchronous and sparse communication mechanism inherently makes SNNs more energy-efficient. These models are particularly effective when deployed on neuromorphic hardware—computing systems inspired by the structure and function of biological neural networks. Neuromorphic hardware is optimized to work with the spiking behavior of neurons and synapses, enabling energy-efficient, event-driven computation. This combination of SNNs and neuromorphic hardware represents a powerful tool for exploiting the computational paradigm offered by SNNs.

By leveraging the temporal dynamics of spiking neurons, SNNs enable efficient processing of time-varying data, such as audio, sensor signals, or video streams. In this work, we explore the application of SNNs to basecalling, a biological and medical task aimed at converting raw signal data from DNA sequencing technologies into nucleotide sequences. The following subsections will provide further context and detail on the components and methods discussed here.

1.1. DNA Sequencing and Basecalling

DNA sequencing is the process of determining the exact order of nucleotides (A, T, C, G) in a DNA molecule. Nanopore sequencing, in particular, is a technique where a DNA molecule is passed through a small protein or synthetic nanopore, and the resulting changes in electrical current are used to decode the genetic sequence. This method has revolutionized sequencing by providing real-time data, enabling faster, cost-effective solutions to sequencing challenges. Accurate DNA base sequencing is crucial for addressing a range of microbiological and medical tasks, such as detecting genetic mutations (especially for cancer diagnosis and treatment) and rapidly identifying pathogens for disease diagnosis, making it a critical advancement in medical development.

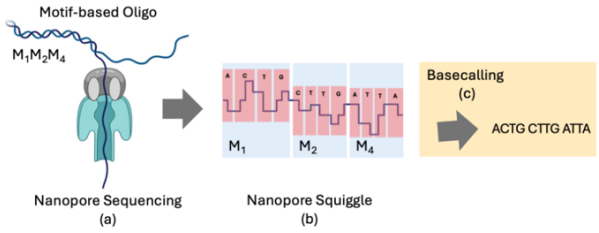

Figure 1. Nanopore readings and Basecalling.

Oxford Nanopore Technologies (ONT) has played a pivotal role in basecalling, offering tools and algorithms that have advanced sequencing through their portability, long read lengths, and real-time data acquisition capabilities. However, the raw signal data produced by nanopore sequencers is often complex and noisy, posing a significant challenge for basecalling. To address this, deep learning-based models, such as BONITO, have emerged as powerful solutions. BONITO utilizes convolutional neural networks (CNNs) to decode the raw, noisy signals into nucleotide sequences with high accuracy. Despite its success, traditional basecalling methods like BONITO remain computationally intensive, which makes them less suited for resource-constrained environments. This limitation motivates the exploration of alternative models, such as Spiking Neural Networks (SNNs), which offer high performance and energy efficiency, particularly when deployed on neuromorphic hardware.

1.2. Spiking Neural Networks

Spiking Neural Networks (SNNs) are a biologically inspired computational paradigm that reimagines artificial intelligence by closely mimicking the temporal and spiking dynamics of biological neural networks. Unlike traditional artificial neural networks (ANNs), which rely on continuous-valued activations and parallel processing, SNNs use discrete, event-driven spikes to encode and transmit information. These spikes occur when a neuron's membrane potential surpasses a predefined threshold, leading to an all-or-nothing "firing" event that carries information in the form of precisely timed events.A crucial feature of SNNs is their synaptic dynamics, which are governed by spike-timing-dependent plasticity (STDP). STDP is a biologically inspired learning rule that strengthens or weakens synaptic connections based on the precise timing between pre-synaptic and post-synaptic spikes. This enables SNNs to learn from dynamic input patterns and adapt their synaptic weights over time, allowing the network to model complex, time-sensitive information. By representing and integrating diverse dimensions of information, including time, frequency, and phase, SNNs are well-suited for tasks that require processing of complex temporal patterns, such as our study case of basecalling.

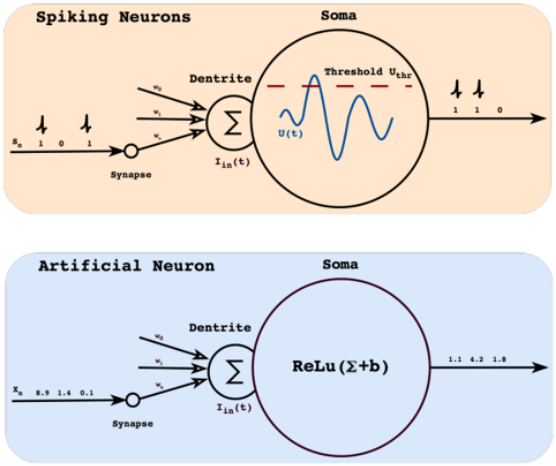

Figure 2. Functional schema of ANN vs SNN.

While traditional ANNs have been successful in numerous applications, they struggle to capture the temporal dynamics inherent in biological processes. SNNs, however, excel at handling dynamic input and integrating multiple information channels, making them highly effective for complex computational tasks. One key advantage of SNNs is their energy efficiency. Due to their sparse, event-driven nature, they consume significantly less power compared to traditional neural networks, especially in large-scale computations.

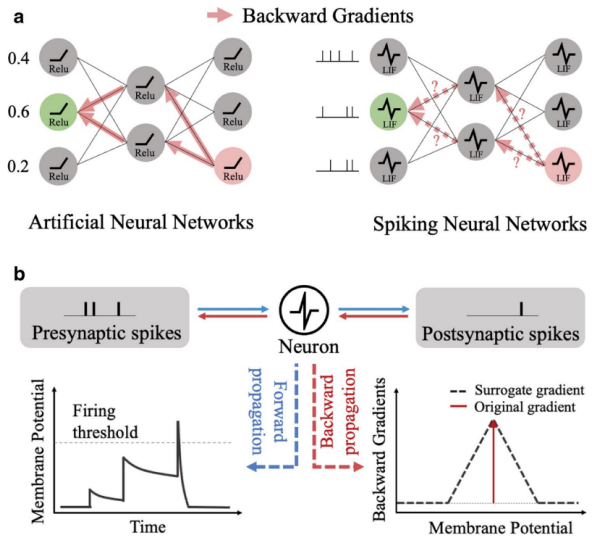

Figure 3. Gradient backpropagation in ANN vs SNN.

1.2.1 Neuromorphic hardware.

Neuromorphic hardware represent a specific class of devices ideally suited for running Spiking Neural Networks (SNNs). These systems operate on an event-driven paradigm, responding only to significant changes or events in the input, reflecting the sparse activation patterns inherent in SNNs. Neuromorphic hardware is optimized for parallel processing, enabling simultaneous processing of multiple spikes across neurons, which boosts computational efficiency. This parallelism, shared by both the brain and SNNs, allows large-scale computations to be executed with reduced energy consumption. Platforms such as IBM's TrueNorth, SpiNNaker, and Intel's Loihi are engineered to support SNNs, offering extreme energy efficiency, parallel processing, and real-time adaptability. The integration of SNNs with neuromorphic hardware offers substantial potential for advancing artificial intelligence, computational neuroscience, and real-time processing, tackling complex computational problems while aligning with the principles of biological neural processing.

1.2.2 The Gradient problem.

A key distinction between traditional Artificial Neural Networks (ANNs) and Spiking Neural Networks (SNNs) lies in how gradients are computed during the backpropagation phase of learning. While ANNs use continuous-valued activation functions that are easily differentiable, SNNs rely on discrete neuron firing, resulting in non-differentiable behavior as shown in Fig 3. This makes applying backpropagation and gradient descent more challenging in SNNs. However, surrogate gradient methods offer a solution by approximating the spike activation function with a differentiable surrogate. This enables the use of conventional machine learning techniques while preserving the benefits of spiking dynamics, with minimal loss in gradient information.

2. Related Works

Since its inception in the 1980s, when nanopore sequencing was first proposed, numerous data analysis methods have been applied to the challenging basecalling task [8]. Over the years, the increasing need for efficient and accurate basecalling techniques has driven the development of a wide range of solutions, particularly with the rise of machine learning methods. These techniques have led to the creation of various network architectures and software tools aimed at improving basecalling accuracy, processing speed, and overall performance [11]. Many of these approaches focused on enhancing model performance by incorporating consensus read strategies, which combine multiple readings to create a more accurate sequence, similar to the ensemble learning concept in machine learning [7]. Other innovative strategies have involved applying breakthroughs from other fields, such as natural language processing (NLP), to further enhance basecalling capabilities [3]. For example, sequence-to-sequence models, commonly used in NLP, have been leveraged to improve the prediction of nucleotide sequences from raw signal data.

In addition to improving accuracy, several works have aimed to optimize the speed of basecalling algorithms, enabling them to operate efficiently in real-time applications. This has led to the development of new components and techniques, such as hardware acceleration, as well as the application of Neural Architecture Search (NAS) methods to optimize network design for faster inference and real-time performance [10]. These optimizations have made basecalling more practical for high-throughput sequencing technologies, where quick processing is crucial for meeting the demands of real-time data analysis [1]. Moreover, with the growing interest in edge computing, more recent research has focused on developing models that can run efficiently on specialized hardware accelerators, such as the Coral accelerator [6], which offer significant advantages in terms of power consumption and processing speed.

Simultaneously, the rise of Spiking Neural Networks (SNNs) has shown promise in addressing the growing challenges of energy consumption and memory footprint in computational models. The combination of SNNs and neuromorphic hardware has proven to be effective in reducing both the computational load and memory requirements, making it a valuable tool for real-time, resource-constrained applications in fields such as nanopore sequencing [2]. These developments point to a future where more energy-efficient, edge-based basecalling systems could be deployed in a wide range of scientific and industrial appli-

cations, benefiting from the combined strengths of machine learning, neuromorphic computing, and hardware acceleration.

3. Methods

The first key implementation of this work stems from the findings of Bonito [5] and [9] and aims to extend the conversion from a standard ANN model to a fully spiking architecture. The other key feature its the implementation of a stateful memory based encoder based on a novel spiking Long Short Term Memory cell, proposed in [4].

3.1. Network Architecture

Our proposed models resemble the structure of a typical artificial neural network (ANN) architecture, comprising a fully spiking convolutional backbone, a feature encoder based on the L2MU memory cell, and a CRF decoder for sequence modeling and error correction.

3.1.1 Fully Spiking Backbone

Our architecture consists of a series of one 1D classical convolutional layer followed by a layer of spiking neurons, repeated three times to form a hierarchical feature extraction pipeline. The classical convolutional layer extracts spatial features from the input data, while the spiking neuron layer encodes these features into spike trains, leveraging the temporal dynamics of SNNs. Each spiking neuron operates based on a Leaky Integrate-and-Fire (LIF) mechanism, where the membrane potential integrates incoming spikes and generates output spikes when a threshold is exceeded. This combination allows the network to capture both spatial patterns and temporal dependencies in the data. By stacking this structure three times, the network progressively extracts higher-level features, enabling robust and efficient processing of complex spatio-temporal signals.

3.1.2 L2MU Encoder

The L2MU (Layered Spiking Legendre Memory Unit) is a neuromorphic adaptation of the original LMU, designed to process spiking signals in alignment with spiking neural networks. Unlike the LMU, which operates on continuous signals, the L2MU transforms each component—memory, hidden layer, and encoding module—into populations of neurons. These neurons communicate via synaptic currents, enabling spike-based information flow.

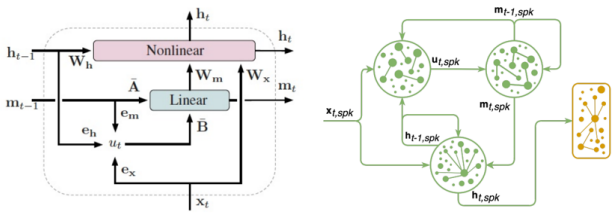

Figure 4. LMU vs L2MU.

Key equations governing the LMU are reformulated to handle spiking dynamics. The synaptic current $u_{t,\text{curr}}$ is computed as:

$$u_{t,\text{curr}} = e_x^T x_{t,\text{spk}} + e_y^T h_{t-1,\text{spk}} + e_m^T m_{t-1,\text{spk}},$$

where $x_{t,\text{spk}}$, $h_{t-1,\text{spk}}$, and $m_{t-1,\text{spk}}$ represent spike emissions from input, hidden, and memory states, respectively. Spikes are generated when membrane potentials exceed threshold values:

$$u_{t,\text{spk}} = \begin{cases} 1, & \text{if } u_{t,\text{mem}} > u_{\text{thr}} \\ 0, & \text{otherwise.} \end{cases}$$

The memory state m_t evolves through discretized matrices:

$$m_t = Am_{t-1} + Bu_t,$$

where A and B are defined as:

$$A = (\Delta t/\theta)A + I, \quad B = (\Delta t/\theta)B.$$

Its spiking counterpart $m_{t,\text{spk}}$ is determined by:

$$m_{t,\text{spk}} = \begin{cases} 1, & \text{if } m_{t,\text{mem}} > m_{\text{thr}} \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, the hidden state h_t is modeled as a neuron population:

$$h_{t,\text{curr}} = W_x x_{t,\text{spk}} + W_h h_{t-1,\text{spk}} + W_m m_{t,\text{spk}},$$

with spikes emitted when membrane potentials surpass thresholds:

$$h_{t,\text{spk}} = \begin{cases} 1, & \text{if } h_{t,\text{mem}} > h_{\text{thr}} \\ 0, & \text{otherwise.} \end{cases}$$

For the Leaky Integrate-and-Fire (LIF) neuron model, the membrane potential v_t is updated as:

$$v_t = \alpha v_{t-1} + \sum_i w_i x_{i,t},$$

where α is the leak factor ($0 < \alpha < 1$), v_{t-1} is the previous membrane potential, w_i are synaptic weights, and $x_{i,t}$ are input spikes at time t. A spike is generated when the membrane potential exceeds a threshold v_{thr}:

$$\text{Spike}_t = \begin{cases} 1, & \text{if } v_t \geq v_{\text{thr}} \\ 0, & \text{otherwise.} \end{cases}$$

After a spike, the membrane potential is reset to a resting potential v_{reset}:

$$v_t = v_{\text{reset}} \quad \text{if Spike}_t = 1.$$

A notable distinction from the original LMU is the absence of an explicit activation function in the L2MU, as the discrete nature of spiking neurons inherently serves this purpose. This adaptation enables the L2MU to efficiently process spiking signals, making it suitable for neuromorphic computing applications.

3.1.3 CRF Decoder

A Conditional Random Field (CRF) decoder is a probabilistic model widely used for structured prediction tasks, such as sequence labeling in natural language processing (e.g., part-of-speech tagging, named entity recognition) and computer vision (e.g., image segmentation). Unlike traditional classifiers that predict labels independently, CRFs model the dependencies between labels in a sequence, leveraging contextual information to improve accuracy. The CRF decoder computes the most likely sequence of labels given an input sequence by maximizing the conditional probability $P(\mathbf{y} \mid \mathbf{x})$, where $\mathbf{x}$ represents the input features and $\mathbf{y}$ represents the output labels. This is achieved through energy-based modeling, where the probability distribution is defined as:

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_i \psi(y_i, \mathbf{x}) + \sum_{i<j} \phi(y_i, y_j, \mathbf{x})\right),$$

where $\psi(y_i, \mathbf{x})$ represents the node potential for label y_i, $\phi(y_i, y_j, \mathbf{x})$ represents the edge potential capturing dependencies between labels y_i and y_j, and $Z(\mathbf{x})$ is the partition function for normalization. Inference in CRFs is typically performed using dynamic programming algorithms like the Viterbi algorithm, which efficiently finds the optimal label sequence. CRF decoders are particularly effective in scenarios where label dependencies are strong, making them a powerful tool for structured prediction tasks.

3.1.4 snnTorch

Our architecture is implemented in python making use of `snnTorch` library. snnTorch is a Python-based framework designed for simulating and training spiking neural networks (SNNs). Built on top of PyTorch, it leverages its automatic differentiation and GPU acceleration capabilities to enable efficient and scalable SNN development. snnTorch provides a wide range of neuron models, learning rules, and synaptic dynamics, making it suitable for both research and practical applications in neuromorphic computing. It simulates the behaviour of Neuromorphic Hardware on a typical Van Neumann architecture, allowing us to train our model and assess its performance all using a more conventional and accessible hardware.

3.2. Proposed Architectures

We propose our architecture based on the models described above. The model, that we call BonitoSCNN in the following experiments, consists of three layers of fully spiking convolutional cells, followed by three layers of L2MU cells. The decision to shorten the encoder was primarily driven by technical constraints, particularly considering the increased depth of the backbone introduced by the convolutional neuron layers. T

In the subsequent section, we present the results of the hyperparameter search for the, followed by an evaluation of its basecalling performance.

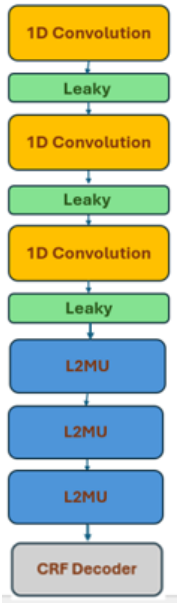

Figure 5. BonitoSCNN Architecture.

4. Experiments

4.1. Dataset

Our dataset comprises genome nanopore readings from various microbiological species, with a focus on bacteria from the Klebsiella, Acinobacter, and other families.

These readings were made publicly available by Monash University in Australia and can be accessed via links provided in our GitHub repository. The dataset is part of a larger collection that also includes human and other complex species' DNA readings. We specifically selected bacterial data due to its simpler genomic structure and shorter sequence lengths, which allowed us to train our model on a broader range of samples.

To ensure compatibility with our architecture, all genome readings in the dataset undergo preprocessing. We adopted the preprocessing pipeline from [9]: the main step in this process involves the use of Tombo, a Python library developed by Oxford University, which performs re-squiggle operations on electric signal-like data. In nanopore sequencing, the raw electric current signal is referred to as a squiggle, which often introduces errors compared to a reference sequence of nucleotids. The Tombo re-squiggle algorithm addresses this by reassigning the squiggle data to the reference sequence, creating a re-squiggle. This algorithm takes as input a read file (in FAST5 format) containing raw signals and their corresponding base calls. The base calls are mapped to a reference genome or transcriptome, and the raw signal is reassigned to the reference sequence based on an expected current level model. This step ensures higher accuracy and consistency in the data, which is crucial for downstream analysis and model training.

4.2. Training and Hyperparameter Optimization

The models we evaluated contain approximately two million parameters, making hyperparameter optimization a complex and computationally demanding task. A comprehensive search over such a vast parameter space requires substantial computational resources, including specialized software, hardware, and time. Given the constraints of this study, it is essential to adopt a rigorous yet efficient workflow that balances complexity reduction with the preservation of generality and model performance.

From these considerations stems the decision to reduce the hyperparameters search to the L2MU encoder of both networks, since it represent the core of this analysis and the part of more interest of which to assess performance. Each encoder layer has the following parameters:

All betas and thresholds refer to the degrade rate and activation threshold of the three neuron populations in the encoder. We decide to keep the L and M values from [9] and to limit our search to the space

$$\mathcal{S} = \{P, \theta, \beta, thresholds\}$$

As a first step in the direction of optimizing the hyperparameter search, we conduct a qualitative analysis of the available dataset to identify more precise and constrained parameter ranges. This targeted approach allows us to refine the search space, thereby improving the efficiency of subsequent experiments.

Table 1. L2MU Model Parameters

Parameter	Description
P	Polynomial approximation grade
θ	Samples duration (s)
L	Hidden Layer size
M	Memory size
β_u	beta_spk_u
th_u	threshold_spk_u
β_h	beta_spk_h
th_h	threshold_spk_h
β_m	beta_spk_m
th_m	threshold_spk_m
β_o	beta_spk_output
th_o	threshold_spk_output

4.2.1 Dataset Analysis

Analyzing the dataset distribution gives us useful information to hypothesize a good parameter range. We plot in Fig.6 and 7 the distribution of the electric signals in input to our dataset. Fig 6 shows the input distribution of the network, and Fig 7. contains the distribution of values before the encoder input, at the end of the backbone after the last convolutional layer. We can see that at the input of the encoder the distribution is well contained in the range (-1,1), thus suggesting similar values for activation levels:

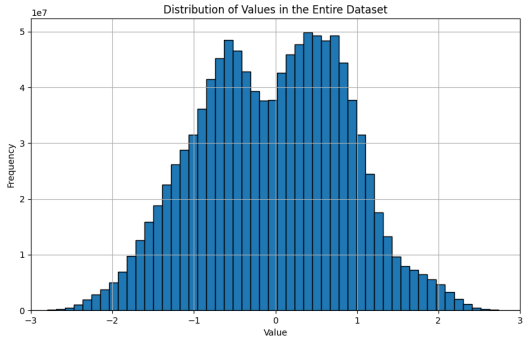

Figure 6. Signal distribution at network input.

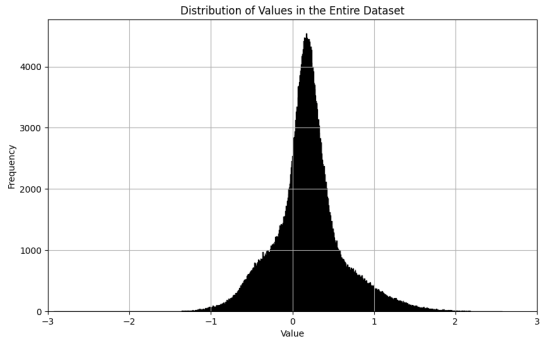

Figure 7. Signal distribution after last convolutional layer.

Another analysis has been conducted on the grade of the signal reconstruction signal. We tested different polynomials against a large number of signal batches, and searched for a value that suggests a good reconstruction while not being prone to overfitting. Results of these analysis are shown in Fig. 8:

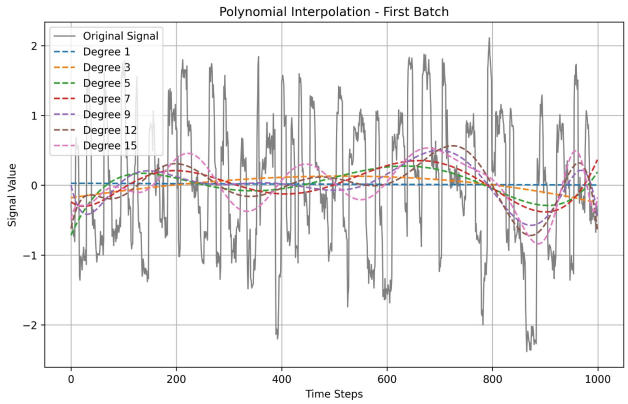

Figure 8. Example signal batch reconstructed with various grades polynomials.

Observing the plot and computing Mean Squared Error aggregated on all dataset batches for various polynomial grades, we decide to test our models choosing P = 9, which provided a MSE of 0.24, suggesting enough generalization capability while retaining a good accuracy.

Finally, to choose the θ parameter we refer to [4] work, and adapt it to Oxford Technologies Nanopore documentation, which declares a sampling reference of 3000 Hz. For this reason, we select the value of one second for theta.

4.2.2 Hyperparameter Search

From these values, we constructed a set of 16 possible parameter combinations and trained the model until loss convergence was achieved. The rapid convergence observed can be attributed to the computationally intensive nature of each epoch; given the substantial volume of data processed per iteration, fewer epochs were required for the model to reach stability.

Table 2. Parameter Values for the Search Space

Parameter	**Values**			
β_u	0.2	0.4	0.6	0.8
θ_u	0.15	0.30	0.45	0.60
β_h	0.1	0.2	0.3	0.4
θ_h	0.35	0.50	0.65	0.80
β_m	0.35	0.55	0.75	0.95
θ_m	0.60	0.75	0.85	0.95
β_o	0.4	0.55	0.7	0.95
θ_o	0.35	0.55	0.75	0.95

Among the tested configurations, three combinations exhibited particularly promising performance, as reported in Table 3.

For the subsequent evaluation phase, we selected models corresponding to the β/θ values presented in Table 4. These selected models underwent an extended training phase, after which we identified the highlighted configuration as the optimal one. The final model outperformed the previous baseline [9] by 8.7% in signal reconstruction accuracy, reaching 64.5%.

These results are particularly encouraging, considering the constraints on training, the reduced model depth—featuring only three encoding layers compared to the five-layer baseline—and the limited number of training epochs due to computational restrictions. A more exhaustive hyperparameter search in the vicinity of the selected values, further extending to the convolutional backbone and incorporating all model parameters, could potentially yield even greater improvements. This suggests that the proposed approach holds significant promise and has the potential to be competitive with more traditional architectures.

4.2.3 Basecalling

To rigorously validate the model's accuracy, we performed basecalling on a restricted test set of bacterial genomes. The dataset consists of 1.5 Gb of sequencing data, comprising 6,000 reads from multiple bacterial species. In this section, we also compare the performance of the second proposed model against baselines provided by [9].

Our results show that Model NN achieves comparable performance to the baseline, whereas Model SNN struggles to reliably assign bases to test squiggles. Unfortunately, due to the unavailability of exact baseline values, we cannot quantitatively confirm the improvement trend previously observed in signal reconstruction. Moreover, to the best of our knowledge, no directly comparable architectures

Table 3. Parameter Values for β and θ

	$(\beta_u$ - β_h - β_m - $\beta_o)$			
$(\theta_u$ - θ_h - θ_m - $\theta_o)$	**0.2 - 0.1 - 0.35 - 0.4**	**0.4 - 0.2 - 0.55 - 0.55**	**0.6 - 0.3 - 0.75 - 0.7**	**0.8 - 0.4 - 0.95 - 0.95**
0.15 - 0.35 - 0.6 - 0.35	0.542	0.573	0.514	0.604
0.30 - 0.50 - 0.75 - 0.55	0.514	0.576	0.577	0.574
0.45 - 0.65 - 0.90 - 0.75	0.628	0.474	0.539	0.615
0.60 - 0.80 - 0.95 - 0.95	0.457	0.597	0.516	0.503

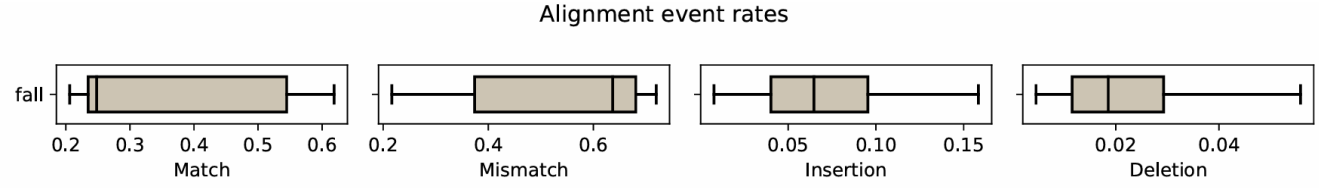

Figure 9. Values for insertion, mismatch, match and deletion events in our experiments.

have been reported in the literature.

However, our findings highlight substantial room for model improvement, as discussed in earlier sections, which would inevitably translate into better performance in the basecalling task.

In Figure 8, we present the results of our basecalling experiments. We report:

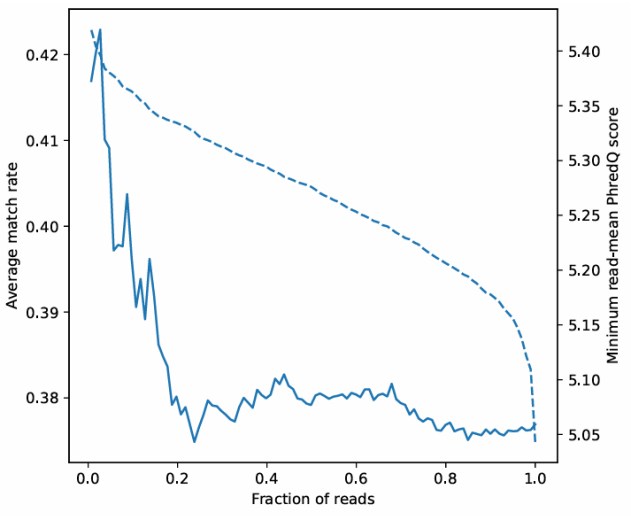

Figure 10. AUC curve of BonitoSCNN.

5. Conclusions

This work, although limited to the context of an academic course, demonstrates the potential of Spiking Neural Network (SNN)-based deep learning models in the biomedical domain, specifically for basecalling. Our proposed model outperforms the previous baseline by 7 percentage points in accuracy during the signal reconstruction phase. However, it has yet to achieve reliable performance in basecalling tasks.

Future research directions are numerous and promising. These include conducting a more extensive and refined hyperparameter search, optimizing a fully-convolutional spiking backbone tailored specifically to the basecalling task, and implementing more robust, comprehensive testing. With these improvements, we anticipate that the model's performance could be further enhanced, ultimately making it more competitive with traditional approaches in the field.

References

[1] Z. X. et al. Fast -bonito: A faster deep learning based basecaller for nanopore sequencing,. Artificial Intelligence in the Life Sciences, vol. 1,, 2021. 3

[2] V. Fra, E. Forno, R. Pignari, T. Stewart, E. Macii, and G. Urgese. "human activity recognition: suitability of a neuromorphic approach for on -edge aiot applications. Neuromorphic Computing and Engineering , vol. 2,, 2022. 3

[3] N. Huang, F. Nie, P. Ni, F. Luo, and J. Wang. "sacall: A neural network basecaller for oxford nanopore sequencing data based on self-attention mechanism. IEEE/ACM Transactions on Computational Biology and Bioinformatics , vol. 19, p. 614 –623, 2022. 3

[4] Benedetto Leto. Lif-based legendre memory unit: neuromorphic redesign of a recurrent architecture and its application to human activity recognition., 2024. 4, 7

[5] Available Online. 4

[6] ì P. Peres˘´ını, V.Boz˘a, and B. Brejov´and T.Vina˘r. "nanopore base calling on the edge,". Bioinformatics, vol. 37, no. 24, pp. 4661 –4667, 2021. 3

[7] J. S. R. and I. Holmes. Sacall: A neural network basecaller for oxford nanopore sequencing data based on self-attention mechanism. Genome Biology, 2021. 3

[8] F. J. Rang, W. P. Kloosterman, and J. de Ridder. From squiggle to basepair: computational approaches for improv-

ing nanopore sequencing read accuracy. ” Genome Biol , vol. 19, p. 90, 2018. 3

[9] Salvatore Tilocca, Carlucci Francesco, and Serra Matteo. 4, 6, 7

[10] V.Boz˘a, P. Peres˘ ´ını, B. Brejova´, and T.Vinar˘. Dynamic pooling improves nanopore base calling accuracy. arXiv, 2021. 3

[11] Y. K Wan, C. Hendra, P. N. Pratanwanich, and J.Goke. Beyond sequencing: machine learning algorithms extract biology hidden in nanopore signal data,. ” Trends in Genetics, vol. 38, p. 246 –257, 2012. 3