University of Sheffield

Designing Spiking Neural Controllers for Neuroprosthetic Systems

Junho Park

Supervisor: Dr Luca Manneschi

A report submitted in fulfilment of the requirements for the degree of MSc in Cybersecurity & AI

in the

Department of Computer Science

September 7, 2025

Declaration

Name: Junho Park

Signature: Junho Park

Date: 07/09/2025

Abstract

This dissertation examines spiking-based neural controllers for sEMG-driven neuroprosthetic systems, benchmarking five architectures: LSTM, TCN-only, SNN-only, SpikingTCN, and Hybrid TCN–SNN. Experiments were conducted on the NinaPro DB6 dataset using rate, latency, and delta encoding schemes, while varying spike simulation steps to analyze the effect of temporal resolution. Results show that TCN achieved consistently high accuracy (85%), while SNN-only demonstrated energy-efficient firing rates (5–20%) but limited performance (62% macro-F1). SpikingTCN offered a biologically plausible compromise, achieving 76.6% with rate encoding. The Hybrid TCN–SNN consistently delivered the best performance, reaching 87.8% macro-F1 with delta encoding while reducing spike activity by more than an order of magnitude compared to SNN-only.

Analysis of firing rates (1–26%) further highlighted the trade-off between accuracy and efficiency, with Hybrid achieving the most favorable balance. Overall, Hybrid TCN–SNN emerges as the most practical candidate for real-time, low-power neuroprosthetic systems, while SpikingTCN demonstrates strong potential as a lightweight and neuromorphic-friendly alternative for future research.

Contents

1 Introduction **1**

1.1 Background and Motivation . . . 1

1.2 Aims and Objectives . . . 3

1.2.1 Experimental Setup . . . 3

1.2.2 Research Objectives . . . 3

2 Literature Survey **4**

2.1 Neuroprosthetic Systems and sEMG-Based Control . . . 4

2.1.1 Components of Neuroprosthetic Systems . . . 4

2.1.2 Physiological Basis of sEMG . . . 6

2.1.3 Conventional Feature-Based Pipelines vs End-to-End Learning . . . 7

2.1.4 Public sEMG Datasets . . . 8

2.2 Deep Learning Approaches for Time-Series Classification . . . 9

2.2.1 Success of Deep Learning in Modern AI . . . 9

2.2.2 CNN-Based Approaches for Time-Series . . . 10

2.2.3 RNN and LSTM-Based Approaches . . . 10

2.2.4 Temporal Convolutional Networks (TCNs) . . . 12

2.2.5 The Energy Barrier of Deep Learning Models . . . 14

2.2.6 Neuromorphic Computing as a Benchmark . . . 14

2.3 Spiking Neural Networks (SNNs) . . . 14

2.4 Spike Encoding Techniques . . . 16

2.4.1 Rate encoding . . . 16

2.4.2 Latency (temporal) encoding . . . 17

2.4.3 Delta (event) encoding . . . 17

2.5 Hybrid Neural Architectures . . . 18

2.5.1 Existing Hybrid Models . . . 18

2.5.2 Structural Approaches . . . 19

2.5.3 Hardware and Energy Considerations . . . 19

2.5.4 Background of TCN–SNN Integration . . . 20

3 Analysis and Planning 21
3.1 Requirements 21
3.2 Desirable Additions 21
3.3 Risk Analysis 22

4 Methodology and Design 24
4.1 Data Preparation 25
4.1.1 Dataset Description 25
4.1.2 Data Pre-processing 26
4.1.3 Batching the Data 28
4.2 Encoding Schemes 28
4.2.1 Rate Encoding 28
4.2.2 Delta Encoding 28
4.2.3 Latency Encoding 29
4.2.4 Hyperparameters 29
4.2.5 Remarks 29
4.3 Model Development 30
4.3.1 Libraries Utilised 30
4.3.2 Long Short-Term Memory Network (LSTM) 30
4.3.3 Temporal Convolutional Network (TCN) 31
4.3.4 Spiking Neural Network (SNN) 32
4.3.5 Hybrid Fusion Model 33
4.3.6 SpikingTCN 34
4.4 Training & Reporting 36

5 Experiments and Results 38
5.1 Comparison of Feature Combinations with in Models 38
5.1.1 LSTM 38
5.1.2 TCN-only Model 39
5.1.3 SNN-only 40
5.1.4 Hybrid TCN-SNN 42
5.1.5 SpikingTCN 43
5.2 Comparison Across Models 44
5.2.1 Training Dynamics 44
5.2.2 Neuron firing-rate comparison 45
5.2.3 Model Comparisons 46
5.2.4 Accuracy–energy efficiency trade-off 47
5.2.5 Discussion 48

6 Conclusion 50

Appendices 59

A Appendix: Additional Figures by Encoding Method 60

List of Figures

1.1 Classical sEMG-based classification pipeline. The raw EMG signals undergo feature extraction and dimension reduction to generate a meaningful representation, which is then classified into discrete gesture labels. [1]. 2

2.1 General signal flow in a neuroprosthetic system. Bioelectrical signals (e.g., sEMG/EEG/ECoG) are acquired, preprocessed (notch/band-pass filtering, normalisation, segmentation), and passed to a feature extraction/encoding module (time, frequency, time–frequency, or raw-to-DL). A pattern recognition/controller module produces control commands that are converted into physical movements by the actuation stage. 5

2.2 Architectural elements in a TCN. (a) **Dilated causal convolution** with dilation factors $d = 1, 2, 4$ and a filter size of $k = 3$, where the receptive field encompasses the entire input sequence. (b) **TCN residual block**, in which a 1×1 convolution is employed when the dimensions of the residual input and the output differ. (c) In the context of a TCN, the blue lines denote filters within the residual function, while the green lines indicate identity mappings, illustrating an example of a residual connection [2]. 13

2.3 LIF neuron 15

2.4 Rate encoding 17

2.5 Latency (temporal) encoding 17

2.6 Delta (event) encoding 18

4.1 Overall workflow of the proposed sEMG-based gesture recognition framework: NinaPro dataset → preprocessing → spike encoding (rate/delta/latency) → model families (LSTM, TCN, SNN, SpikingTCN, Hybrid) → training → evaluation (accuracy/F1/confusion matrix + spike-based measures). 25

4.2 Sliding-window segmentation (W=200, overlap=100) and window-level majority voting for label assignment. 27

4.3 Comparison of spike encoding methods (Rate, Delta, Latency). 30

4.4 Architecture of the TCN. 32

4.5 Architecture of the SNN. 33

4.6 Hybrid TCN–SNN fusion: parallel feature extraction for classification 34

4.7 Architecture of the SpikingTCN. 36

5.1 Training and validation curves for all models under latency encoding. TCN and Hybrid exhibit rapid and stable convergence, while SNN-only achieves moderate accuracy. LSTM suffers from overfitting, and SpikingTCN shows higher variance due to dense spiking dynamics. 45
5.2 Accuracy–firing rate trade-off. 48

A.1 Ts = 20, Encoding = rate 60
A.2 Ts = 20, Encoding = rate 61
A.3 Ts = 20, Encoding = latency 62
A.4 Ts = 20, Encoding = latency 63
A.5 Ts = 20, Encoding = delta 64
A.6 Ts = 20, Encoding = delta 65

List of Tables

2.1 Comparison of representative hybrid ANN-SNN models 19

3.1 Overview of project steps for the hybrid TCN-SNN sEMG study 23
3.2 Risk analysis of project steps for the hybrid TCN-SNN sEMG study 23

4.1 Sample records extracted from the NinaPro DB6 (S1_D1_T1). EMG shows the first four channels; ACC shows the first three accelerometer values (x-z). . . . 26

5.1 LSTM classifier performance across three evaluation runs on NinaPro DB6. . 39
5.2 TCN classifier performance across three evaluation runs on NinaPro DB6. . . 40
5.3 Hybrid TCN-SNN performance by spike encoding at T_s=20 (Macro-F1, %). . 42
5.4 Hybrid TCN-SNN Top-1 Accuracy (%) across spike simulation steps T_s. . . . 42
5.5 SpikingTCN performance by spike encoding at T_s=20 (Accuracy / Macro-F1, %). 44
5.6 SpikingTCN Top-1 Accuracy (%) across spike simulation steps T_s. 44
5.7 Spike statistics across models and encodings ($T = 20$, $N = 256$). Spikes are averaged per sample; firing rate is mean per neuron. 46
5.8 Comparison of best models across different architectures. 46

Chapter 1

Introduction

1.1 Background and Motivation

In recent years, healthcare and assistive technologies have seen neuroprosthetics emerge as a revolutionary frontier. Neuroprosthetic systems serve as a direct link between the human nervous system and external devices, restoring motor functions, facilitating intuitive human-machine interaction (HCI), and opening up new opportunities in rehabilitation and wearable assistive systems [3]. These systems usually depend on precisely understanding an user's motor intentions because of bioelectrical signals made by the human body.

Among various biosignals, surface electromyography (sEMG) measures the electrical activity of muscle fibres in response to electrochemical or neurological stimuli, providing quantitative information on muscle activation, tone, and fatigue, as well as motor unit recruitment and synchronisation patterns.[4, 5] Among various bio-signals, surface electromyography (sEMG) is widely adopted across multiple domains due to its high informational value and versatility [4, 6, 7].

Nonetheless, sEMG signals often introduce noise, exhibit high dimensionality, coupled with substantial variance amid individuals along time [8, 9]. The standard sEMG classification mechanism (Figure 1.1) characteristically includes extracting features from unprocessed signals. Subsequently, dimensionality reduction and classification do ensue upon this extraction. This multi-tiered framework renders the complete system detailed and impedes end-to-end enhancement, since each tier functions autonomously.

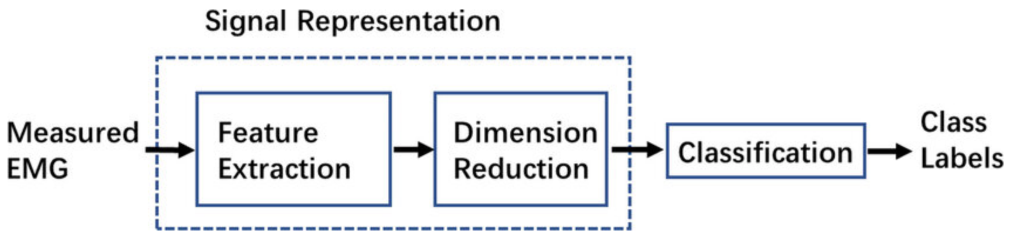


Figure 1.1: Classical sEMG-based classification pipeline. The raw EMG signals undergo feature extraction and dimension reduction to generate a meaningful representation, which is then classified into discrete gesture labels. [1].

To address these limitations, end-to-end deep learning models such as CNNs and RNNs have recently been widely applied to sEMG analysis. These models take raw or minimally preprocessed signals as input and perform feature extraction and classification in an integrated manner, thereby streamlining the pipeline and improving performance as the amount of data increases.

Established deep learning models such as CNNs, RNNs as well as LSTMs have demonstrated excellent performance upon processing time-series data. Nevertheless, they expend a significant amount computationally, utilise wide-ranging quantities of memory, and employ critical power, which renders them inappropriate for energy-restricted platforms [1, 10, 11]. Neural architectures drawing from biology, mediating exactitude, velocity and energetic economy, have kindled fascination in this divide linking notional skill and practical application [1].

From amongst these, Spiking Neural Networks (SNNs) have materialized as an approach that seems hopeful. The third iteration of neural networks, closely emulating information processing mechanisms of biological neurons, is regarded as SNNs [12]. SNNs interact by way of individual electrical impulses, in contrast to typical artificial neural networks (ANNs), which handle compact, uninterrupted information during each increment of time. They function through an event-driven and temporally sparse fashion, which substantially diminishes power usage while remaining highly congruous to neuroprosthetic hardware platforms [13, 14, 15].

The Temporal Convolutional Network (TCN) represents another established methodology in modelling temporal data, augmenting typical sequential neural network architectures. TCNs employ dilated causal convolutions for the effective capture of long-range temporal dependencies. This methodology furnishes parallel training stability and strict temporal causality since it guarantees that no future information gets utilised for present predictions [2, 16]. TCNs are therefore especially suitable. These attributes are helpful to instantaneous applications.

This study puts forward a compound schema that combines the biological efficacy of SNNs and the time-based modelling proficiencies of TCNs, for the purpose of developing a novel group of neural controllers apt for sEMG-based neuroprosthetic control. The suggested model is created for being precise, strong, and energy-sparing, and that renders it virtually employable inside wearable and edge computing milieus.

1.2 Aims and Objectives

sEMG signals offer a reliable interface for interpreting motor intentions, though their irregular and high-dimensional nature presents modeling challenges. However, the high-dimensional and irregular nature of sEMG signals presents significant challenges to model development, particularly when aiming to achieve both energy efficiency and real-time responsiveness. While deep learning models such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory networks (LSTMs) have demonstrated high classification accuracy, their computational complexity and power consumption restrict their practicality in wearable and embedded environments.

To address these challenges, this research proposes a novel hybrid architecture that combines the advantages of Temporal Convolutional Networks (TCNs) and Spiking Neural Networks (SNNs). The proposed TCN–SNN model is designed to perform temporally precise and energy-efficient hand gesture recognition from sEMG signals. The primary objective of this study is to evaluate the effectiveness and efficiency of the proposed hybrid approach and to compare its performance against alternative model architectures across multiple dimensions.

1.2.1 Experimental Setup

For a fair and thorough evaluation, this study considers five model configurations. To benchmark sequential classification performance, a conventional RNN based on LSTM is employed. To examine energy-efficient inference under sparse firing conditions, a pure Spiking Neural Network (SNN-only) is introduced. In contrast, a Temporal Convolutional Network (TCN-only) is evaluated for its ability to capture long-range temporal dependencies. The proposed Hybrid TCN–SNN model combines TCN-based feature extraction with SNN firing-rate fusion, with the aim of achieving a balance between accuracy and efficiency.

Finally, the SpikingTCN model, which embeds Leaky Integrate-and-Fire (LIF) neurons directly within convolutional blocks and applies temporal average pooling, is explored as a biologically plausible alternative.

1.2.2 Research Objectives

The primary objective of this study is to investigate the trade-offs among energy consumption, predictive accuracy, and temporal expressiveness in sEMG-based gesture recognition.

Specifically, the research compares LSTM, TCN, SNN, SpikingTCN, and Hybrid TCN–SNN architectures, highlighting both their strengths and limitations alongside classification accuracy. Another aim is to analyse the impact of different spike encoding methods including rate, delta, and latency on classification outcomes as well as spiking activity. Computational efficiency is assessed by quantifying spike counts, which serve as a practical proxy for energy consumption.

Finally, the study examines how hybrid and spiking-based architectures perform in real-time control scenarios, thereby illustrating the fundamental balance between accuracy and energy efficiency in neuroprosthetic applications.

Chapter 2

Literature Survey

2.1 Neuroprosthetic Systems and sEMG-Based Control

In this section, we explore the basic ideas of neuroprosthetic systems as well as current advancements, especially focusing on how surface electromyography (sEMG) facilitates the effective control of these technologies.

2.1.1 Components of Neuroprosthetic Systems

A neuroprosthetic system creates a bidirectional interface from the human nervous system to external assistive devices because it restores or improves motor functions lost due to neurological damage or musculoskeletal disorders [17, 18]. These systems can decode neural signals in real-time as well as reflect the user's intentions [19, 20]. In this respect, they can be distingushed from simple mechanical assistive devices.

Bioelectrical signals such as sEMG, EEG, and ECoG are acquired from the user's muscle activity and then preprocessed to remove noise and normalise the data. In the subsequent feature extraction/encoding stage, time domain, frequency domain and time-frequency domain methods are applied, or alternatively, deep learning-based encoding can be used using minimally preprocessed signals. The pattern recognition/controller module then classifies the user's intent or generates continuous control commands. Finally, the actuation system converts these commands into physical movements. Neuroprosthetic Systems, as shown in Figure 2.1, are composed of several components, each of which will be examined in detail.

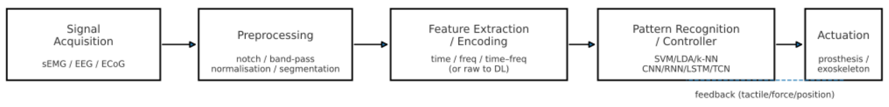


Figure 2.1: General signal flow in a neuroprosthetic system. Bioelectrical signals (e.g., sEMG/EEG/ECoG) are acquired, preprocessed (notch/band-pass filtering, normalisation, segmentation), and passed to a feature extraction/encoding module (time, frequency, time–frequency, or raw-to-DL). A pattern recognition/controller module produces control commands that are converted into physical movements by the actuation stage.

Signal Acquisition Module

The Signal Acquisition Module functions as a mechanism because it documents bioelectrical signals which reflect the user's motor intentions. There exists a range of sensor types. Surface electromyography (sEMG) electrodes [9], electroencephalography (EEG), together with electrocorticography (ECoG) constitute the sensors principally utilised in this stage.

Preprocessing Unit

Preprocessing is defined as the process of noise reduction and baseline normalisation to prepare signals for learning. Numerous studies have shown that whenever preprocessing is of a high quality, feature extraction as well as classification are significantly more accurate [9, 21].

Feature Extraction / Encoding Module

This stage entails the rapid abstraction and structured representation of information from preprocessed signals to facilitate motion classification. The established methodology integrates time-domain features (RMS, MAV, ZC, SSC), frequency-domain features (mean/median frequency), and time–frequency-domain features (Wavelet, STFT), followed by dimensionality reduction prior to classifier input [21, 22]. This approach offers high interpretability and relatively low computational cost, making it well suited for deployment in embedded environments. In contrast, deep learning–based methods autonomously learn hierarchical feature representations by processing raw or minimally preprocessed signals. Architectures such as CNNs, TCNs, and Transformers have demonstrated sustained performance improvements over conventional feature-based approaches through the use of this paradigm [23, 24].

Pattern Recognition / Neural Controller

In this stage, the derived features are utilised to generate continuous control signals or to classify the user's intent. Conventional classifiers such as SVM, LDA, and k-NN possess low computational complexity, making them suitable for embedded environments where real-time performance and energy efficiency are critical, although they remain limited in modelling complex temporal dependencies [22]. In contrast, deep learning–based approaches such as CNNs, RNNs, LSTMs, and TCNs are capable of effectively learning nonlinear patterns as well as long-term temporal dependencies. From a practical perspective, controllers must balance inference latency comprising both model execution against classification accuracy, energy constraints, and memory budgets [2].

Actuation System

The actuation system converts predicted control commands into physical movements and can be implemented in various forms, including robotic hands, prosthetic arms and legs, exoskeletons, and wheelchairs. The responsiveness and stability of the controller are directly influenced by friction, backlash, and the dynamic characteristics of actuation mechanisms such as motors, gear reducers, and hydraulic systems. Recent research has aimed to enhance user operability and system stability through closed-loop control, integrating tactile, force, and positional feedback. Furthermore, several clinical studies have reported sensory feedback using epidural electrical stimulation (EES) or peripheral nerve stimulation [19, 20]. However, due to constraints in cost, complexity, and portability [17], commercial wearable devices often implement only limited feedback capabilities.

2.1.2 Physiological Basis of sEMG

Surface electromyography (sEMG) is a technique that is used to measure the electrical activity within skeletal muscles when they are contracting voluntarily or involuntarily. α-motor neurons fire so as to generate motor unit action potentials (MUAPs), representing this electrical activity. The action potentials that result will then propagate along within muscle fibres [9, 25]. Factors such as the number of active motor units and their firing rates, in addition to the relative position between the electrodes and the muscle fibres do influence the recorded signal [7]. sEMG is non-intrusive, provides a high temporal resolution on the millisecond scale, and gives little discomfort to the user. Its suitability to repeated measurements and wearable applications [21] arises because it requires relatively simple instrumentation.

One key advantage for sEMG is its predictive capability. This proves particularly useful in neuroprosthetic control. sEMG signals can occur from 50 to 150 ms before visible muscle contraction begins, so motor intentions are decoded prior to actual movement [26, 27, 28]. This property reduces perceived latency advantageously and improves responsiveness when controlling real-time neuroprosthetics.

Nonetheless, various sources of variability as well as noise intrinsically affect sEMG signals. Sweat, as well as skin moisture, in addition to electrode displacement changes electrode–skin impedance, and inter-individual anatomical differences along with muscle fatigue, including motor unit recruitment patterns, affect physiology, and all of these degrade signal quality

[9, 29, 30]. Due to these factors, stable decoding is challenging, so optimised electrode placement, signal preprocessing, as well as model adaptation strategies are necessary over time.

2.1.3 Conventional Feature-Based Pipelines vs End-to-End Learning

Customary sEMG-based gesture recognition generally adheres to a multi-stage pipeline, with each stage concentrating upon extracting meaningful information from raw signals for a particular task. This pipeline includes the subsequent phases.

Preprocessing elevates signal fidelity as it includes noise attenuation, standardises baselines, and partitions signals.

Feature Extraction obtains diverse attributes, such as time-domain attributes, including RMS and MAV. Frequency-domain attributes such as Fourier coefficients get extracted, coupled with time-frequency domain attributes, for example, wavelet transforms.

Dimensionality reduction employs approaches to diminish the feature space. Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) are instances of techniques employed in retaining important information.

Categorisation: Typical classifiers like k-Nearest Neighbours (k-NN), Artificial Neural Networks (ANN), and Support Vector Machines (SVM) are employed to categorise gestures [31].

This typical conduit construes data nicely as it profits people when datasets are diminutive. Seeing as each individual stage has the capability for optimisation separately, it is particularly well suited to efficient real-time applications. However, because each stage operates independently, end-to-end optimisation is difficult to perform, as it leads to suboptimal performance with real-time efficiency diminishing.

In contrast to alternative methodologies, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Temporal Convolutional Networks (TCN), which constitute holistic deep learning strategies, acquire attributes directly from unprocessed or scarcely manipulated sEMG signals [32]. Such methodologies permit the network to autonomously ascertain layered characteristic depictions, which obviates any requirement for hand-operated characteristic derivation. These techniques attain improved exactitude compared with typical ones. Nevertheless, these methodologies require substantially greater computational power and memory, which could restrict low-power and real-time neuroprosthetic applications [11, 33].

Recent investigations have demonstrated that deep learning models, specifically TCNs, exhibit superior performance when set against typical feature-based methods regarding classification precision. TCNs employ dilated convolutions for modelling long-term temporal dependencies, thus affording them a distinctive advantage. Consequently, they are apt at sequence data processing like sEMG [2]. Nonetheless, employing such models may prove arduous within embedded systems or real-time applications because they remain computationally demanding.

2.1.4 Public sEMG Datasets

The development and evaluation of sEMG-based neuroprosthetic systems depend heavily on high-quality datasets for effective machine learning model training and testing. Publicly available sEMG datasets have become essential resources, enabling the creation of accurate and generalisable models for decoding motor intentions from surface electromyography (sEMG) signals.

NinaPro Dataset

The NinaPro dataset is a key resource for sEMG-based gesture recognition, providing extensive recordings of diverse hand and finger movements. Its large participant pool and wide range of gestures offer high variability, which is critical for developing robust and generalisable models. NinaPro is widely used as a benchmark for both traditional feature-based and deep learning-based gesture classification methods [34, 35].

PhysioNet Dataset

PhysioNet is an open-access archive containing well-characterised digital recordings of physiological signals and related data. It includes electrocardiograms, heart rate time series, and, in some datasets, sEMG recordings acquired during motor tasks. PhysioNet enables investigations into the effects of muscle fatigue on signal interpretation and supports the development of models that perform reliably under noisy or highly variable conditions [36, 37].

CapgMyo Dataset

The CapgMyo datasets consist of high-density (HD) sEMG recordings collected from 23 participants using an 8×16 electrode array during hand gesture tasks. They are widely used for intra-session and inter-session gesture recognition research, as well as for evaluating domain adaptation algorithms. Their capability for combined spatial–temporal analysis makes them valuable for next-generation muscle–computer interface (MCI) research, often in conjunction with NinaPro and CSL-HDEMG [35].

CSL-HDEMG Dataset

The CSL-HDEMG dataset is another representative HD-sEMG dataset, comprising 27 finger and hand movement classes, including dynamic motions. Recorded using a 192 electrodes(8×24) grid, it captures both spatial and temporal variations in gestures, making it particularly suitable for evaluating spatio-temporal deep learning models such as 3D CNNs [38]. It is frequently employed in fine finger control studies and comparative analyses of time-series modelling approaches.

Collectively, these datasets are fundamental to advancing sEMG-based neuroprosthetic systems. The diversity and scale of the data support the development of machine learning algorithms capable of delivering accurate, real-time, and robust control in practical applications.

2.2 Deep Learning Approaches for Time-Series Classification

2.2.1 Success of Deep Learning in Modern AI

Over the past decade, the field of Artificial Intelligence (AI) has undergone remarkable expansion, primarily driven by the advent and maturation of deep learning. This paradigm shift has enabled human-level, and in some cases superhuman, performance across diverse application domains, including computer vision, natural language processing, and speech recognition, firmly establishing AI as a foundational technology in both industry and academia [39].

Contemporary deep learning fundamentally diverges from preceding AI methodologies. The confluence of large-scale dataset availability, high-performance computing resources such as GPUs and TPUs, advances in optimisation techniques, and the introduction of novel neural architectures has led to unprecedented performance gains [40, 41]. The success of modern AI is exemplified by several landmark developments.

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs), inspired by the organisation of the visual cortex, leverage local receptive fields and weight-sharing mechanisms to extract spatially localised patterns from images [42]. A pivotal moment occurred with the introduction of AlexNet, which achieved a groundbreaking performance improvement at the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), catalysing the widespread adoption of deep learning in computer vision [43]. Subsequent architectures such as ResNet and EfficientNet further advanced the state of the art by improving both accuracy and computational efficiency [44, 45].

Transformer architecture

The Transformer architecture, introduced by Vaswani et al., employs an attention mechanism to assign varying importance to different elements of the input sequence while enabling parallel computation [46]. This architecture has largely supplanted recurrent models for sequential data processing in applications such as machine translation, large-scale language modelling, and speech recognition, and it forms the foundation for advanced models including BERT, the GPT family, and multimodal learning frameworks [47].

AlphaGo

DeepMind's AlphaGo, integrating Monte Carlo Tree Search (MCTS) with deep neural networks, surpassed world-class professional players in the game of Go, a domain characterised by combinatorial complexity and strategic uncertainty [48]. This demonstrated the feasibility and competitiveness of deep learning–based decision-making in highly uncertain environments [49].

Collectively, these examples illustrate the evolution of deep learning from a specialised pattern recognition tool to a versatile framework capable of addressing a broad spectrum of complex tasks. The following section examines representative deep learning architectures commonly employed in time-series processing, including CNNs, RNNs, LSTMs, and TCNs.

2.2.2 CNN-Based Approaches for Time-Series

Convolutional Neural Networks (CNNs), originally developed for image processing, have in recent years been effectively applied to the analysis of time-series biosignals such as surface electromyography (sEMG) [42, 50]. By employing local receptive fields with weight-sharing mechanisms, CNNs can efficiently extract local patterns from input data. Their parallel computation structure further enables faster training and higher hardware efficiency. Owing to these characteristics, CNNs are particularly effective for detecting local patterns that occur over short time intervals [23].

For time-series data $\mathbf{x} \in \mathbb{R}^{T \times C}$, where T denotes the sequence length and C is the number of channels, the one-dimensional convolution operation can be expressed as:

$$y_t^{(k)} = \sigma\left(\sum_{c=1}^{C}\sum_{i=0}^{K-1} w_{i,c}^{(k)} \cdot x_{t+i-p,c} + b^{(k)}\right) \tag{2.1}$$

Here, K is the kernel size, p is the padding size, $w_{i,c}^{(k)}$ represents the weights of the k-th filter, $b^{(k)}$ is the bias term, and $\sigma(\cdot)$ denotes a nonlinear activation function (e.g., ReLU, tanh). This operation learns local patterns along the temporal axis, and applying multiple filters in parallel enables the capture of multi-scale spatio-temporal features [50].

Nevertheless, CNNs have limitations in modeling long-term dependencies. Due to the fixed receptive field of the convolutional kernel, capturing long-range temporal patterns requires either very deep networks or large kernel sizes. The effective receptive field R of a CNN layer depends on the kernel size K and the number of layers L as follows:

$$R = 1 + (K-1) \cdot L \tag{2.2}$$

2.2.3 RNN and LSTM-Based Approaches

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) differ from traditional feedforward neural networks in that they are designed to retain both short-term and long-term memory. This architecture is well-suited for handling sequential or time-series data, as it can capture temporal dependencies, making RNNs particularly effective in tasks such as speech recognition, language modeling, and time-series forecasting.

In a simple RNN, the hidden state at each time step t is computed as [51]:

$$h^{(t)} = \phi(W_{hx}x^{(t)} + W_{hh}h^{(t-1)} + b_h) \tag{2.3}$$

The output is obtained using a softmax activation:

$$y^{(t)} = \text{softmax}(W_y h^{(t)} + b_y) \tag{2.4}$$

where:

- $x^{(t)}$: input at time t
- $h^{(t)}$: hidden state at time t
- W_{hx}: weights from input to hidden layer
- W_{hh}: recurrent weights between hidden states
- W_y: weights from hidden layer to output
- b_h, b_y: bias terms
- ϕ: activation function (e.g., tanh or ReLU)

Although RNNs are capable of modeling temporal patterns, they suffer from training issues such as the *vanishing* and *exploding gradient* problems [51, 52].

Long Short-Term Memory (LSTM)

To overcome these limitations, **Long Short-Term Memory (LSTM)** networks were proposed by Hochreiter and Schmidhuber [53]. Unlike standard RNNs, LSTMs incorporate a memory cell and gate mechanisms to maintain long-term dependencies.

The key components of LSTM include:

- Input gate $i^{(t)}$
- Forget gate $f^{(t)}$
- Output gate $o^{(t)}$
- Input modulation (candidate) gate $g^{(t)}$
- Cell state $s^{(t)}$

The following equations describe a modern LSTM:

$$f^{(t)} = \sigma(W_f x^{(t)} + U_f h^{(t-1)} + b_f) \tag{2.5}$$

$$i^{(t)} = \sigma(W_i x^{(t)} + U_i h^{(t-1)} + b_i) \tag{2.6}$$

$$g^{(t)} = \tanh(W_g x^{(t)} + U_g h^{(t-1)} + b_g) \tag{2.7}$$

$$s^{(t)} = f^{(t)} \odot s^{(t-1)} + i^{(t)} \odot g^{(t)} \tag{2.8}$$

$$o^{(t)} = \sigma(W_o x^{(t)} + U_o h^{(t-1)} + b_o) \tag{2.9}$$

$$h^{(t)} = o^{(t)} \odot \tanh(s^{(t)}) \tag{2.10}$$

Here, $\odot$ denotes element-wise multiplication, and σ is the sigmoid activation function.

While LSTMs effectively capture long-range dependencies, they are computationally intensive due to their complex gating structure. As the depth or size of the hidden layers increases, the training time and memory usage grow substantially [54, 55]. Furthermore, the inherently sequential nature of LSTMs makes parallelization difficult, which can hinder training efficiency [51, 55].

2.2.4 Temporal Convolutional Networks (TCNs)

Temporal Convolutional Networks (TCNs) are convolutional architectures specifically designed for time-series and sequential data modelling. In contrast to recurrent neural networks (RNNs) or long short-term memory networks (LSTMs), TCNs process the entire input sequence in parallel while preserving temporal order through the use of a **causal convolutional structure** [2]. They have been proposed as an effective alternative to overcome the inherent limitations of recurrent architectures by using dilated causal convolutions that capture long-range dependencies with parallel training and stable gradients while enforcing strict causality [2, 16, 56].

To capture long-range dependencies, TCNs integrate **dilated convolutions** [56] with **residual blocks** [57], enabling the modelling of temporal relationships over extended time spans without excessive network depth or parameter growth.

The TCN framework is based on two key principles:

- The input and output sequences must have the same length [2].
- Future information must not influence past predictions (causality).

The first condition is satisfied using one-dimensional **fully convolutional networks (FCNs)** [58] with appropriate zero-padding (of size kernel size $-$ 1) to ensure the output length matches the input length. The second condition is achieved through **causal convolutions**, where the output at time step t depends only on inputs from time steps $\leq t$, thus maintaining temporal causality [2].

While causal convolutions are effective for enforcing temporal order, they require either deep networks or large kernel sizes to capture long-term dependencies. TCNs address this challenge using dilated convolutions, which introduce gaps between kernel elements to expand the receptive field without increasing the number of parameters [56]. A one-dimensional dilated convolution is formally defined as:

$$F(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i} \tag{2.11}$$

where:

- d is the dilation factor,
- k is the filter size,
- x is the input sequence,
- $f(i)$ is the filter weight at position i.

This formulation allows for efficient computation while achieving a receptive field that grows exponentially with depth.

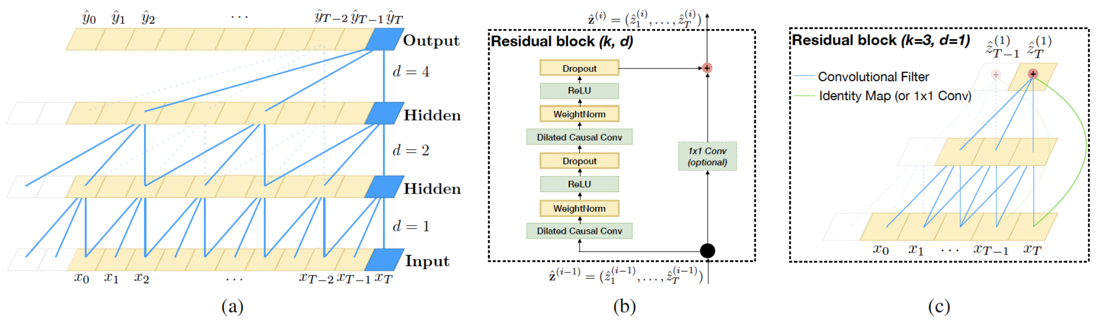

Output
Hidden
Hidden
Input
$d = 4$
$d = 2$
$d = 1$
$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ ... $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
x_0 x_1 x_2 ... x_{T-2} x_{T-1} x_T
(a)
Residual block (k, d)
$\hat{\mathbf{z}}^{(i)} = (\hat{z}_1^{(i)}, \ldots, \hat{z}_T^{(i)})$
Dropout
ReLU
WeightNorm
Dilated Causal Conv
Dropout
ReLU
WeightNorm
Dilated Causal Conv
1x1 Conv (optional)
$\hat{\mathbf{z}}^{(i-1)} = (\hat{z}_1^{(i-1)}, \ldots, \hat{z}_T^{(i-1)})$
(b)
Residual block (k=3, d=1)
$\hat{z}_{T-1}^{(1)}$ $\hat{z}_T^{(1)}$
Convolutional Filter
Identity Map (or 1x1 Conv)
x_0 x_1 ... x_{T-1} x_T
(c)

Figure 2.2: Architectural elements in a TCN. (a) **Dilated causal convolution** with dilation factors $d = 1, 2, 4$ and a filter size of $k = 3$, where the receptive field encompasses the entire input sequence. (b) **TCN residual block**, in which a 1×1 convolution is employed when the dimensions of the residual input and the output differ. (c) In the context of a TCN, the blue lines denote filters within the residual function, while the green lines indicate identity mappings, illustrating an example of a residual connection [2].

As network depth increases, training becomes more challenging due to vanishing gradients. To alleviate this, TCNs employ **residual connections** [57], which preserve gradient flow and facilitate the training of deeper architectures. A residual block is expressed as:

$$o = \text{Activation}(F(x) + x) \tag{2.12}$$

where $F(x)$ denotes a sequence of transformations applied to the input x.

Compared to RNN-based models, TCNs offer several advantages:

- Parallelizable training due to their convolutional structure [2],

- Flexible receptive field control via dilation [56],
- Stable gradient flow through residual connections [57],
- Ability to handle variable-length inputs and reduced memory usage during training.

However, TCNs also present certain limitations:

- The entire input sequence must be available for inference [2].
- Performance may be sensitive to receptive field size depending on the application domain.

Despite these limitations, TCNs remain a powerful choice for sequence modelling tasks requiring both long-term dependency handling and computational efficiency.

2.2.5 The Energy Barrier of Deep Learning Models

As powerful deep learning models such as CNNs, LSTMs, and TCNs have emerged, and as the influence of the AI industry continues to grow and its applications in automation and prediction become more widespread, energy consumption has also increased significantly [10, 59]. In large-scale enterprises, this often results in a substantial rise in computational costs for both training and inference, leading not only to increased financial expenses but also to growing environmental concerns [59].

2.2.6 Neuromorphic Computing as a Benchmark

Alongside the success of AI, the limitations of available resources, rising computational costs, and increasing environmental concerns are steering the field in a new direction [15, 59]. In response, next-generation paradigms such as Green AI [59] and neuromorphic computing have been proposed, aiming to enhance sustainability and energy efficiency. In the following chapter, a Spiking Neural Network (SNN) model is proposed as one of these neuromorphic alternatives.

2.3 Spiking Neural Networks (SNNs)

Unlike conventional artificial neural networks (ANNs), spiking neural networks (SNNs) more closely emulate the information-processing mechanisms of biological nervous systems [12]. In SNNs, neurons communicate via discrete electrical pulses ("spikes"); unlike the continuous-valued activations used in other networks, the precise timing of these spikes carries critical information. Owing to these properties, SNNs are regarded as the "third generation" of neural networks that more faithfully model biological information processing. They support time-based coding strategies such as rate coding and latency (timing) coding [13, 14, 60].

Spiking Neural Networks (SNNs) exploit *event-driven computation* and *temporal sparsity*, whereby computations are performed only when spikes occur, thereby reducing overall energy consumption. This property makes SNNs particularly well aligned with neuromorphic computing, and recent reviews on algorithm-hardware co-design have emphasized their advantages in this regard [15].

Representative neuron models

Among the most widely used neuron models are the *Integrate-and-Fire (IF)* and the *Leaky Integrate-and-Fire (LIF)* models. In the LIF formulation, the membrane potential evolves according to

$$\tau_m \frac{du(t)}{dt} = -u(t) + RI(t), \quad I(t) = \sum_j w_j \sum_f \epsilon\big(t - t_j^{(f)}\big). \tag{2.13}$$

Here, τ_m represents the *membrane time constant*, R the *membrane resistance*, w_j the *synaptic weight*, $\epsilon(\cdot)$ the *synaptic kernel*, and $t_j^{(f)}$ the f-th spike time of the j-th presynaptic neuron. Conceptually, the model describes the membrane potential as a leaky integrator of input currents, which resets once a firing threshold is reached [12].

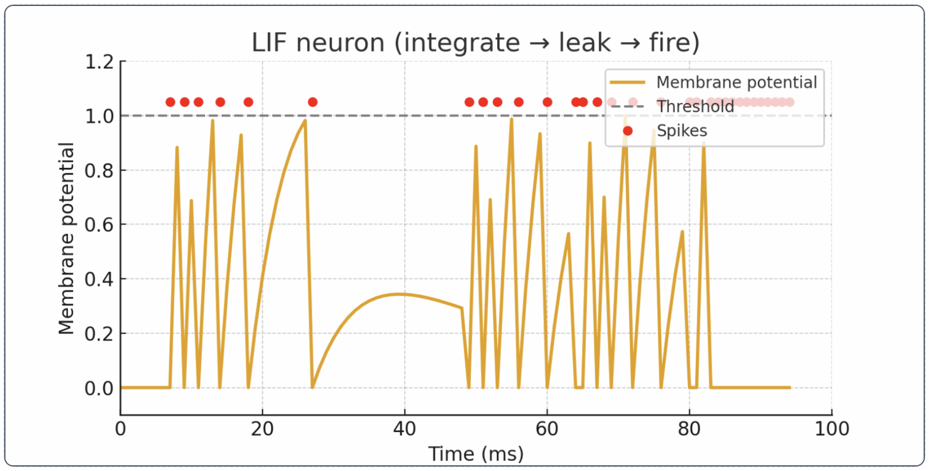


Figure 2.3: LIF neuron

Learning in Spiking Neural Networks

Spikes are inherently non-differentiable events, which precludes the direct application of standard backpropagation to spiking neural networks (SNNs). Among the various learning strategies proposed, the *surrogate gradient* method where the spike derivative is approximated by a smooth, differentiable function has emerged as a particularly effective solution. This approach enables gradient-based optimization in SNNs, thereby bridging the gap between biological plausibility and practical trainability [61, 60]. Recent comprehensive reviews provide detailed accounts of these advances [62], with both supervised and unsupervised learning paradigms being actively investigated.

Representative examples of surrogate gradient functions include:

$$\text{Fast Sigmoid:} \quad \frac{1}{(1+\alpha|u-\vartheta|)^2} \tag{2.14}$$

$$\text{ATan:} \quad \frac{\alpha}{1+(\alpha(u-\vartheta))^2} \tag{2.15}$$

where α is a slope parameter that controls the sharpness of the surrogate gradient. Commonly adopted loss functions include:

- **Cross-Entropy Loss:** spike counts or firing rates of output neurons are used as logits.
- **Mean Squared Error (MSE):** minimizes the difference between target signals and the observed firing rates.

In Summary, SNNs not only more faithfully emulate biological neural systems but also provide remarkable energy efficiency through event-driven computation [14], distinguishing them from conventional ANNs and DNNs. While they do not consistently surpass DNNs in raw performance [62], SNNs are increasingly recognised as a promising candidate for next-generation neuromorphic computing [60, 61]. A critical factor for their effective utilization lies in the manner by which continuous input signals are transformed into spike trains. The next section therefore examines spike encoding techniques, with particular emphasis on rate-based encoding approaches [12, 63, 60].

2.4 Spike Encoding Techniques

Spike encoding converts a continuous signal (e.g., sEMG) into a spike train or event stream so that an SNN can process it, thereby determining the balance among expressive power, energy efficiency (sparsity), and biological plausibility. During training, it is particularly important that the encoder output couple naturally with *surrogate-gradient (SG)* optimization [60, 61]. Because sEMG exhibits fine spatiotemporal variations, especially with high-density electrodes, this section focuses on **Rate**, **Latency (Temporal)**, and **Delta (Event)** coding [14, 15, 62].

2.4.1 Rate encoding

Normalized input magnitude is interpreted as a firing probability over time, and spikes are sampled accordingly; within a chosen observation window, the average spike frequency represents stimulus intensity. Both Poisson and Bernoulli interpretations are common, and channel-wise normalization with probability mapping (e.g., sigmoid or linear clipping) yields a simple implementation that is relatively robust to noise [14, 13]. Averaging can dilute fine temporal structure, and extending the window to reach higher accuracy increases the total spike count, which can reduce the energy advantage [62]. Owing to its stability, rate coding is frequently used as a baseline in SG training pipelines [60, 61].

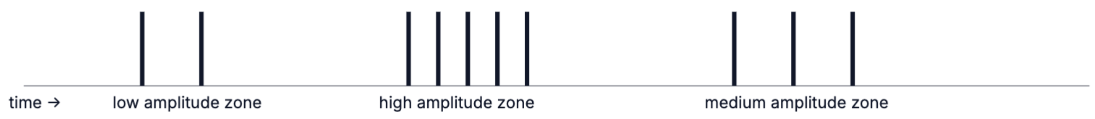


Figure 2.4: Rate encoding

2.4.2 Latency (temporal) encoding

Input magnitude is encoded by the first-spike time or precise spike timing under the assumption that stronger stimuli fire earlier. Decisions can then be made from a few early spikes, which shortens inference latency and reduces spike counts while preserving temporal structure. Sensitivity to timing jitter and synchronization errors makes stable normalization, careful choice of the reference time range, and differentiable timing approximations for SG training especially important [14, 60, 61, 64]. This approach is particularly relevant for high-density sEMG, where channel alignment and fine temporal resolution are required.

time →
strong → fast

Figure 2.5: Latency (temporal) encoding

2.4.3 Delta (event) encoding

Events are produced only when the signal increment exceeds a threshold; rising and falling events can be represented separately or as a signed single stream. In segments with little change, few or no events are generated, which naturally induces sparsity and reduces data, computation, and power. Performance depends on design choices such as threshold selection, drift compensation, and leaky-integration-based reconstruction. In practice, training often maintains an SG path by feeding continuous (analog) inputs and applying an event-based representation at inference (including STE), or by using adaptive thresholds [15, 60, 61, 62].

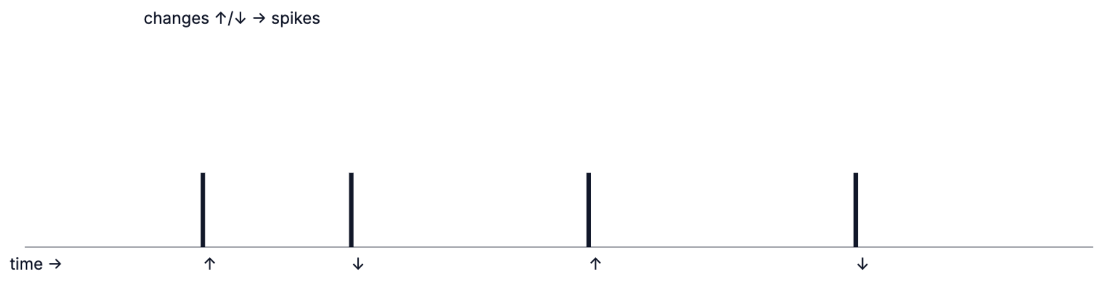


Figure 2.6: Delta (event) encoding

2.5 Hybrid Neural Architectures

With the growing demand for real-time and energy-efficient biosignal classification in wearable and implantable devices, hybrid neural architectures have gained significant attention. Conventional deep learning models such as CNNs, LSTMs, and Transformers provide high accuracy but are computationally expensive and power-hungry, limiting their deployment in resource-constrained environments. In contrast, spiking neural networks (SNNs) achieve intrinsic energy efficiency through event-driven sparse computation. However, they often struggle to capture long-range dependencies in complex temporal data and typically fail to reach the accuracy levels of conventional ANNs [61, 60, 65, 66]. Hybrid neural architectures have been proposed as a solution to overcome these complementary limitations.

2.5.1 Existing Hybrid Models

Recent studies have introduced a range of hybrid models that integrate ANNs and SNNs to leverage the strengths of both paradigms. For instance, Spike-TCN integrates dilated convolutions with spiking neurons to enhance sequence modeling while maintaining energy efficiency [63]. Similarly, iSpikformer embeds spiking dynamics into a Transformer backbone, demonstrating efficiency in complex sequential tasks such as video classification [67, 68]. In biomedical applications, PCSN-Net combines CNNs and SNNs in a parallel architecture, achieving both high accuracy and energy efficiency in skin lesion analysis [69]. More recently, Song et al. proposed the Multiscale Fusion-enhanced SNN (MFSNN), which incorporates TCNs with channel attention mechanisms, improving both accuracy and efficiency in BCI signal decoding [70].

Collectively, these studies highlight that hybrid models not only improve accuracy but also enhance robustness, generalizability, and energy efficiency.

Table 2.1: Comparison of representative hybrid ANN-SNN models

Model	**Backbone**	**Key Feature / Domain**
Spike-TCN [63]	TCN + Spiking	Energy-efficient sequence modeling
iSpikformer [68]	Transformer + Spiking	Long-range modeling, video tasks
PCSN-Net [69]	CNN + SNN (parallel)	Biomedical, high accuracy + efficiency
MFSNN [70]	TCN + Attention + SNN	BCI decoding, accuracy + efficiency
SpikeSTAG [71]	ST-Graph + SNN	Spatio-temporal forecasting

2.5.2 Structural Approaches

Hybrid architectures are typically implemented through four main strategies [15, 72, 73]:

- **Parallel Branching** – ANN and SNN modules process the same input independently, and their extracted features are fused via concatenation, attention-based fusion, or similar methods. This approach maximizes representational diversity and robustness.
- **Serial Injection** – Features extracted by an ANN are converted into spikes and passed into an SNN for temporal refinement, enhancing biological plausibility.
- **Knowledge Distillation** – The ANN functions as a teacher model, transferring knowledge to the SNN and enabling efficient standalone inference.
- **Block-level Hybridization** – Spiking neurons are directly embedded into ANN computation blocks, enabling continuous feature extraction and event-driven spiking within a single block. This method naturally combines the strong feature learning capability of ANNs with the energy efficiency of SNNs, and it operates effectively in end-to-end training structures through surrogate gradient optimisation.

2.5.3 Hardware and Energy Considerations

Another key advantage of hybrid architectures lies in their compatibility with emerging hardware platforms. Neuromorphic chips such as Intel Loihi and IBM TrueNorth execute SNN computations efficiently through event-driven processing, while ANN computations are well supported on GPUs and TPUs. Prior research has shown that hybrid ANN–SNN systems can achieve higher accuracy than pure SNNs while consuming less power than pure ANNs [72]. This balance makes them particularly suitable for wearable medical applications, where battery lifetime and latency are critical factors. For example, biosignal analysis tasks involving ECG, EEG, and EMG have increasingly leveraged the low-power characteristics of SNNs [65, 66], while studies on high-density electrode array-based sEMG [38] provide practical motivation for adopting hybrid models.

2.5.4 Background of TCN–SNN Integration

In this context, the integration of Temporal Convolutional Networks (TCNs) with SNNs represents a particularly promising direction. TCNs, through causal and dilated convolutions, effectively capture long-range dependencies in biosignals such as sEMG. Meanwhile, SNNs excel at energy-efficient inference and sparse coding. Combining the two offers three key advantages:

1. The strong sequence modeling capacity of TCNs,
2. The event-driven efficiency of SNNs,
3. A balance between accuracy and power consumption.

Although parallel fusion structures may introduce additional computational overhead, they enhance feature complementarity and robustness, forming a foundation for future optimizations via serial integration or knowledge distillation.

Chapter 3

Analysis and Planning

3.1 Requirements

The primary goal of this study is to design and evaluate a combined architecture leveraging the advantages of **Spiking Neural Networks (SNNs)** and **Temporal Convolutional Networks (TCNs)** to achieve both high accuracy and energy efficiency in sEMG-based neuroprosthetic control. For this purpose, the following major stages were defined.

Literature Review Recent research on SNNs, TCNs, and sEMG-based control systems was reviewed in depth. Particular attention was given to the energy-saving characteristics of biologically inspired SNNs, the long-term dependency modeling capability of TCNs, and the performance variations among different spike encoding schemes.

Baseline Implementation and Testing Baseline models for LSTM, SNN-only, TCN-only, HybridTCN-SNN and SpikingTCN architectures were implemented using the NinaPro sEMG dataset. Initial classification performance and resource consumption were measured to establish benchmark results prior to introducing the hybrid architecture.

Hybrid Architecture Design A hybrid TCN-SNN, SpikingTCN architectures were created by integrating TCNs and SNNs. Three spike encoding schemes: rate coding, delta coding, and latency coding, were applied individually to produce model variants.

Evaluation Metrics and Analysis The evaluation plan encompassed accuracy measurement and estimation of computational load based on spike counts, used as an energy proxy.

3.2 Desirable Additions

In addition to the required experiments, the following extended studies are planned to enhance the model's practical applicability and academic contribution:

- **Real-Time Simulation Evaluation:** Test the trained model in a real-time streaming sEMG data environment to verify response speed and system stability.
- **Channel Reduction Experiments:** Gradually reduce the number of sEMG channels to measure performance changes and propose optimal sensor configurations for wearable devices.
- **Edge Deployment Testing:** Deploy the model on neuroprosthetic hardware simulators or embedded GPU/FPGA platforms to measure power consumption and processing latency.
- **Real-Time Simulation and Hardware Testing:** Highest risk, due to potential hardware limitations and challenges in meeting real-time processing requirements. However, optimization insights gained from this stage are expected to be of substantial academic and practical value.

3.3 Risk Analysis

The potential risks associated with each stage are as follows:

- **Literature Review and Baseline Implementation:** Low risk, as these are based on well-established methods.
- **Hybrid Architecture Design:** Medium risk, given the limited prior work combining SNNs and TCNs, and potential training difficulty variations depending on the spike encoding method used.
- **Spike Encoding Comparison Experiments:** Risk of suboptimal performance or training instability with certain encoding methods, which could affect the interpretation of overall results.

Title of Steps	**Description**	**Requirement Status**	**Time Required**
Literature Review	Comprehensive review of sEMG-based neuroprosthetic systems, deep learning models (TCN), SNNs, HybridTCNSNN, SpikingTCN and spike encoding schemes. Identification of gaps in energy-efficient real-time control.	Required	4 weeks
Baseline Implementation and Testing	Implement SNN-only and TCN-only using NinaPro sEMG dataset to establish performance and energy benchmarks.	Required	6 weeks
Hybrid Architecture Design	Design hybrid TCN–SNN models incorporating rate, delta, and latency coding for spike representation. Optimise architecture for temporal precision and low power consumption.	Required	6 weeks

Table 3.1: Overview of project steps for the hybrid TCN-SNN sEMG study

Title of Steps	**Risk Level**
Literature Review	Low
Baseline Implementation and Testing	Low
Hybrid Architecture Design	Medium
Hybrid Model Evaluation	Medium

Table 3.2: Risk analysis of project steps for the hybrid TCN-SNN sEMG study

Chapter 4

Methodology and Design

This study focuses on balancing accuracy and efficiency with an emphasis on sEMG-based gesture recognition. Figure 4.1 summarizes the overall workflow. A single-session file (`S1_D1_T1.mat`) from the NinaPro dataset is used following preprocessing. Three encoding schemes: rate, delta, and latency, are applied to generate spike trains within the same window, allowing for the analysis of the trade-off between encoding strategy and classification performance.

Subsequently, three model families are trained under unified timesteps: (i) an SNN-only LIF network, (ii) the proposed SpikingTCN incorporating dilated temporal convolutions with LIF neurons, and (iii) the proposed Hybrid TCN-SNN.

All models are optimised end-to-end using the Adam optimiser with early stopping. Evaluation metrics include classification accuracy, macro- and weighted-F1 scores, and confusion matrices. These are complemented by spike-based indicators such as total, mean, and maximum spike counts, firing-rate distributions, and raster plots. This combined evaluation framework enables a joint comparison of predictive performance and sparsity, the latter serving as a proxy for energy efficiency.

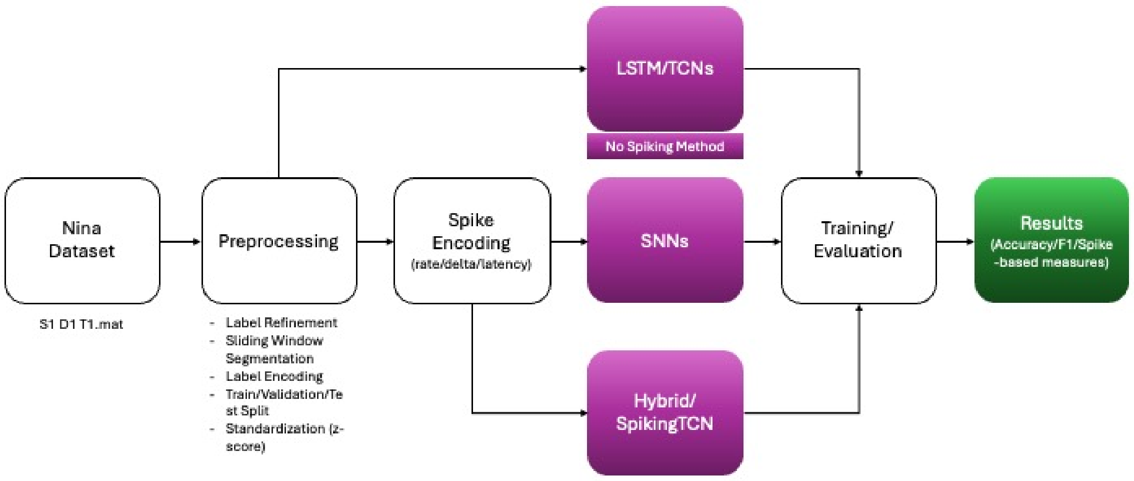

LSTM/TCNs
No Spiking Method
Nina Dataset
S1 D1 T1.mat
Preprocessing
- Label Refinement
- Sliding Window Segmentation
- Label Encoding
- Train/Validation/Test Split
- Standardization (z-score)
Spike Encoding (rate/delta/latency)
SNNs
Hybrid/ SpikingTCN
Training/ Evaluation
Results (Accuracy/F1/Spike -based measures)

Figure 4.1: Overall workflow of the proposed sEMG-based gesture recognition framework: NinaPro dataset → preprocessing → spike encoding (rate/delta/latency) → model families (LSTM, TCN, SNN, SpikingTCN, Hybrid) → training → evaluation (accuracy/F1/confusion matrix + spike-based measures).

4.1 Data Preparation

4.1.1 Dataset Description

This study employs the publicly available NinaPro DB6 corpus of surface electromyography (sEMG) recordings. Seven distinct grasp gestures were performed 12 times by ten subjects. The gestures were conducted twice daily (morning and afternoon) over five consecutive days. Recordings were obtained at 2 kHz using 14-channel Delsys Trigno double-differential electrodes uniformly placed on the forearm. These recordings were synchronized with additional modalities including accelerometry and eye-tracking.

The dataset includes the following variables:

- **subj**: subject identifier
- **emg (16 columns)**: sEMG signals from 14 channels (with 2 empty columns)
- **acc (48 columns)**: tri-axial accelerometer data from 12 electrodes (some columns empty)
- **stimulus / restimulus**: instructed gesture labels / post-processed refined gesture labels

- **object / reobject**: object used / re-labeled object
- **repetition / rerepetition**: repetition index (original vs. refined labeling)
- **repetition object**: object repetition index
- **daytesting**: acquisition day (1–5)
- **time**: acquisition session (1 = morning, 2 = afternoon)

Representative examples of these variables are summarized in Table 4.1. For analysis, the ground truth labels are taken from the post-processed *restimulus*, as they more accurately reflect the executed gestures. The dataset is partitioned into 70% for training, 15% for validation, and 15% for testing of the predictive models.

Index	**Gesture**	**Refined**	**Object**	**Rep**	**Day**	**EMG_CH1-4**	**ACC_x-z**
122793	1	1	4	8	1	(0.000, 0.000, 0.000, 0.000)	(0.081, 1.034, -0.418)
1105770	0	0	-1	0	1	(-0.000, 0.000, -0.000, -0.000)	(-0.135, 1.079, -0.274)
127515	1	1	4	8	1	(0.000, -0.000, -0.000, -0.000)	(0.097, 1.070, -0.414)
935201	0	0	-1	0	1	(-0.000, -0.000, -0.000, -0.000)	(-0.220, 1.103, -0.250)
627036	0	6	-1	0	1	(-0.000, 0.000, 0.000, 0.000)	(0.239, 1.018, -0.549)

Table 4.1: Sample records extracted from the NinaPro DB6 (S1_D1_T1). EMG shows the first four channels; ACC shows the first three accelerometer values (x-z).

The dataset comprises several annotated variables that guide both preprocessing and subsequent analysis. The Gesture field denotes the instructed gesture label (stimulus), with the value 0 corresponding to the resting state. The Refined field provides a more accurate version of these labels by correcting and adjusting annotations after post-processing. The Object field records the identifier of the object used during the task, where a value of minus 1 indicates the absence of annotation. For each gesture instance, the Rep field specifies the repetition index, while the Day field indicates the acquisition day, ranging from one to five. In addition to these labels, the dataset also contains biosignals: EMG CH1–4 represent example values from the first four sEMG channels, and ACC x–z capture accelerometer measurements along the x, y, and z axes. Collectively, these variables enable reliable alignment of gesture annotations with multimodal sensor data.

4.1.2 Data Pre-processing

The preprocessing pipeline in this study proceeds through five distinct steps:

1. **Data Selection and Label Refinement**
   Only active gesture segments are retained, and rest labels (label 0) are excluded. This mitigates class imbalance during training and reduces bias in spike statistics introduced by rest intervals.

2. **Sliding Window Segmentation**
With the sampling frequency of 2 kHz, a fixed window of length $W = 200$ (≈ 100 ms) and overlap $O = 100$ (50 ms stride) is applied to construct sequences. For each window $X \in \mathbb{R}^{W \times C}$, the representative label $\hat{y}$ is assigned using majority voting among the labels within the window:

$$\hat{y} = \arg\max_{k} \#\{t \in [1, W] \mid y_t = k\}.$$

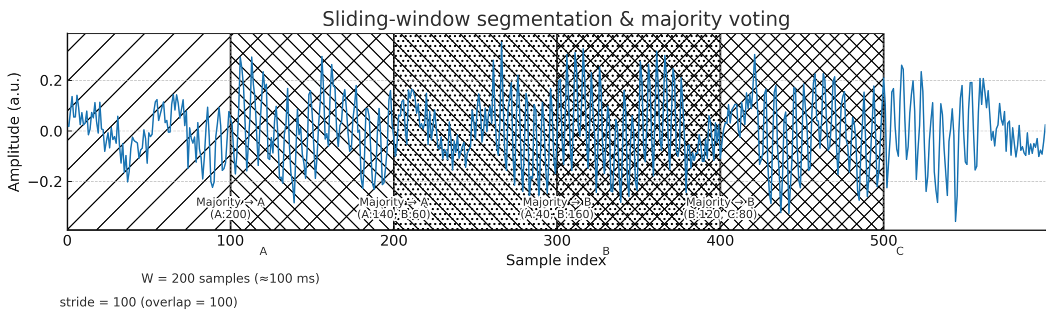


Figure 4.2: Sliding-window segmentation (W=200, overlap=100) and window-level majority voting for label assignment.

3. **Label Encoding**
Since gesture labels may not be continuous, all non-rest classes are mapped to consecutive indices $\{0, \ldots, K-1\}$. This enables a consistent definition of cross-entropy loss and one-hot encoding dimensions, ensuring comparability of performance across models.

4. **Train/Validation/Test Split**
A stratified split is performed at the window level. First, 15% of the dataset is held out as a test set. From the remaining data, 17.6% is allocated to validation, resulting in an approximate split of (train/val/test) $\approx$ (70/15/15). Random seeds are fixed to guarantee reproducibility.

5. **Standardization (z-score)**
Channel amplitudes vary due to electrode and skin conditions, which may bias training toward high-amplitude channels. To address this, per-channel mean μ_c and standard deviation σ_c are computed from the training set only, and the same parameters are applied to the validation and test sets. Data are reshaped into $(N_{\text{win}} \cdot W, C)$ for normalization and then restored to (N_{win}, W, C):

$$x'_{t,c} = \frac{x_{t,c} - \mu_c}{\sigma_c + \varepsilon}.$$

This preprocessing avoids handcrafted feature extraction or filtering, and instead ensures (i) fixed-length sequence construction, (ii) leakage-free per-channel normalization, and (iii) consistent labeling. These steps allow TCN, SNN, and SpikingTCN models to be trained end-to-end under identical input conditions. The window length and stride were chosen with real-time constraints in mind (≈100 ms observation, 50 ms update).

4.1.3 Batching the Data

sEMG sequences segmented by sliding windows ($W = 200$, overlap $= 100$) are fed into training as mini-batches. The training DataLoader uses a batch size of 32 with random shuffling, ensuring that each parameter update mixes windows across gestures, repetitions, and object conditions. This reduces overfitting to specific sequences and lowers gradient variance.

The batch input shape is $(B = 32, T = W = 200, C)$, where T is the temporal length of the window and C is the number of channels (electrodes).

For SNN, Hybrid SNN branch, and SpikingTCN models, each window is converted into spike trains with temporal length $T_s = 20$, followed by training using surrogate-gradient-based BPTT. The choice of $T_s = 20$ reflects a trade-off between accuracy and efficiency, since computational and memory costs scale.

By contrast, TCN processes the raw window of length W in parallel using dilated causal convolutions.

This batching–encoding–model pipeline exposes each batch to diverse conditions, thereby improving generalization while maintaining feasible time and computation budgets for stable optimization.

4.2 Encoding Schemes

We transform windowed sEMG sequences into spike trains using a unified `SpikeEncoder`.

4.2.1 Rate Encoding

Continuous inputs are mapped to spike probabilities through a sigmoid nonlinearity. Bernoulli sampling at each step yields spike trains:

$$\mathbf{P} = \sigma(\mathbf{X}), \quad \mathbf{S}^{(s)} \sim \text{Bernoulli}(\mathbf{P}), \quad s = 1, \ldots, S.$$

The result $\mathbf{S} \in \{0,1\}^{S \times B \times T \times C}$ preserves amplitude information as firing probability and is well suited for stochastic temporal coding.

4.2.2 Delta Encoding

Delta modulation emits a spike whenever the absolute discrete derivative exceeds a threshold θ_Δ:

$$\mathbf{D}_t = \mathbf{X}_t - \mathbf{X}_{t-1}, \quad \mathbf{S}_t = \mathbb{1}\{|\mathbf{D}_t| \geq \theta_\Delta\}, \quad t = 2, \ldots, T.$$

With $\mathbf{S}_1 = \mathbf{0}$, this produces $\mathbf{S} \in \{0,1\}^{B\times T\times C}$. If needed, the result is duplicated along the simulation dimension to match (S, B, T, C). Delta encoding highlights abrupt transitions (onsets/offsets) and results in sparse, event-driven spike trains.

4.2.3 Latency Encoding

Larger input magnitudes are mapped to earlier spike times within S discrete steps. To stabilise across channels and windows, per-channel min–max normalisation is applied:

$$\tilde{\mathbf{X}}_{b,:,c} = \frac{\mathbf{X}_{b,:,c} - \min_t \mathbf{X}_{b,t,c}}{\max_t \mathbf{X}_{b,t,c} - \min_t \mathbf{X}_{b,t,c} + \varepsilon}.$$

Optionally, amplitude thresholding ($\mathbf{X} \leftarrow \mathbb{1}\{\mathbf{X} \geq \tau\} \odot \mathbf{X}$ with $\tau =$ `latency_threshold`) suppresses noise. The latency mapper then generates $\mathbf{S} \in \{0,1\}^{S\times B\times T\times C}$, where stronger values tend to spike earlier. The parameter `latency_linear` determines whether mapping is linear or nonlinear.

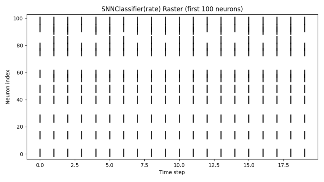
4.2.4 Hyperparameters

- `encoding_type` $\in$ {`rate`, `delta`, `latency`} specifies the encoding scheme.
- `num_steps` (S): number of simulation steps (temporal resolution).
- `latency_linear` (bool): linear vs. nonlinear latency mapping.
- `latency_threshold` (τ): optional amplitude gate applied prior to normalisation.
- `per_channel_norm` (bool): per-channel min–max normalisation for scale invariance.

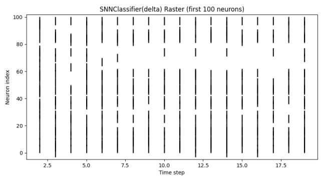
4.2.5 Remarks

- **Unified shape:** (S, B, T, C) across schemes simplifies batching and downstream SNN/TCN integration.
- **Sparsity control:** `num_steps`, thresholds (θ_Δ, τ), and normalisation determine spike density and are tuned to balance accuracy and efficiency.
- **Latency stability:** Per-channel min–max normalisation improves robustness to inter-session and electrode gain variations. Global contrast can be used if disabled (`per_channel_norm=false`).
- **Reproducibility:** Rate encoding is stochastic; thus, multiple seeds should be reported along with spike statistics such as firing-rate histograms and raster plots.

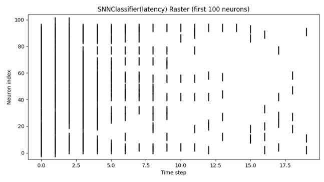
Implementation uses `snntorch.spikegen.delta`, `.rate`, and `.latency`, with outputs normalized to (T_s, B, T, C).



(a) Rate encoding



(b) Delta encoding



(c) Latency encoding

Figure 4.3: Comparison of spike encoding methods (Rate, Delta, Latency).

4.3 Model Development

This study develops and compares five models for classifying hand gestures from surface electromyography (sEMG) sequences: the Temporal Convolutional Network (TCN), the Spiking Neural Network (SNN), Long short-term memory(LSTM), Hybrid TCN-SNN and SpikingTCN. All models share the same preprocessing pipeline (§4.1.2), follow an identical data partitioning scheme (train/validation/test), and are trained under consistent loss functions and optimization settings.

4.3.1 Libraries Utilised

- **numpy, pandas**: numerical computation and dataset construction
- **matplotlib, seaborn**: visualization of training curves, confusion matrices, and spike statistics
- **scikit-learn**: `StandardScaler`, `LabelEncoder`, `train_test_split`, F1-score
- **PyTorch**: model definition, training/inference, and `DataLoader`
- **snnTorch**: LIF neurons, surrogate gradients, and spike encoding
- **scikit-learn.metrics / stats**: accuracy, macro-/weighted-F1 scores

4.3.2 Long Short-Term Memory Network (LSTM)

Architecture

The LSTM classifier is designed to capture sequential dependencies in sEMG signals using stacked recurrent layers:

- **LSTM Encoder**
  - Input size: number of sEMG channels (C)
  - Hidden size: $H = 128$
  - Number of layers: $L = 2$

  - Bidirectional LSTM with dropout ($p = 0.2$) applied across layers
  - Output hidden states h_n have dimensions $(L \times \text{dir}, B, H)$, where $\text{dir} = 2$ for bidirectional
  - For bidirectional mode, the final forward and backward hidden states are concatenated, resulting in a feature vector of size $2H$

- **Classifier Head**
  - Linear($2H$, 128) $\rightarrow$ ReLU $\rightarrow$ Dropout(0.2) $\rightarrow$ BatchNorm(128)
  - Linear(128, 64) $\rightarrow$ ReLU $\rightarrow$ Dropout(0.1)
  - Linear(64, K) for gesture classification into K classes

4.3.3 Temporal Convolutional Network (TCN)

Architecture

The TCN employs stacked dilated 1D convolutions to capture both short- and long-term temporal dependencies:

- **TemporalBlock (2 $\times$ Conv1d layers)**
  - Dilation rate $d = 2^i$, kernel size $k = 3$, stride $= 1$
  - Apply `Conv1d(padding =` $(k-1) \cdot d$`)` to preserve the sequence length, and then use `Chomp1d(chomp_size =` $(k-1) \cdot d$`)` to cut off the last $(k-1) \cdot d$ time steps in order to remove future information leakage.
  - Conv1d $\rightarrow$ BatchNorm1d $\rightarrow$ ReLU $\rightarrow$ Dropout(0.2)
  - Residual connections adjusted with 1×1 Conv when channel dimensions differ
- **Stack Configuration**: channel sizes [64, 128, 256] corresponding to dilations $d = \{1, 2, 4\}$
- **Self-Attention**: Multi-Head Attention (8 heads) applied to the highest-level sequence features
- **Classifier Head**: Linear(128) $\rightarrow$ ReLU $\rightarrow$ Dropout(0.2) $\rightarrow$ BatchNorm $\rightarrow$ Linear(64) $\rightarrow$ ReLU $\rightarrow$ Dropout(0.2) $\rightarrow$ Linear(K)

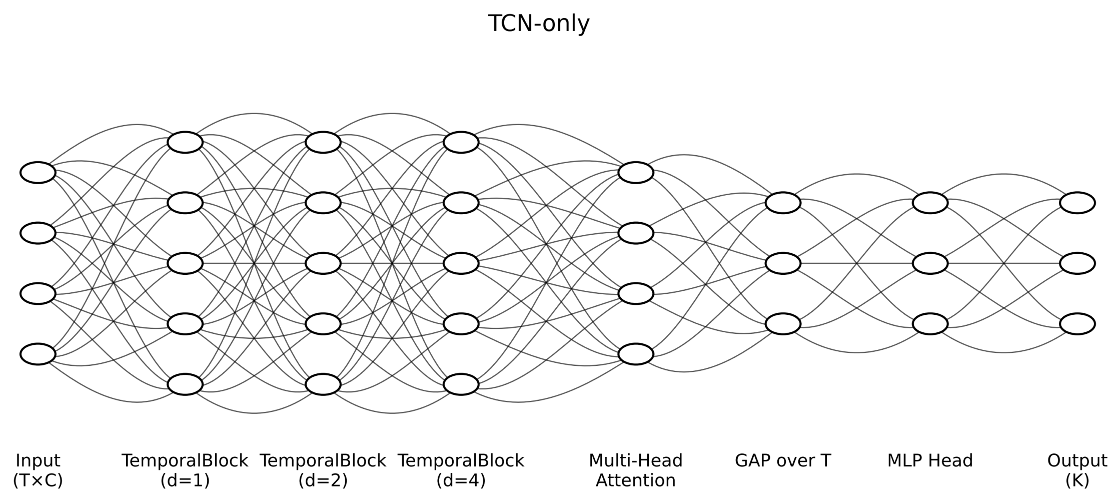


Figure 4.4: Architecture of the TCN.

4.3.4 Spiking Neural Network (SNN)

Spike Encoding

Continuous EMG signals are transformed into spike trains to emulate biological temporal coding.

- The original sequence length $T = 200$ is uniformly partitioned into $T_s = 20$ frames, with events allocated per frame.
- The output tensor has shape (T_s, B, T, C), followed by temporal averaging (**time pooling**) into (T_s, B, C)

Neuron Model & Layers

- Leaky Integrate-and-Fire (LIF) neurons (snnTorch) with $\beta = 0.9$ and $v_{\text{th}} = 1.0$.
- The non-differentiability of spike generation is approximated using a **fast-sigmoid surrogate gradient** with slope = 25.
- Multi-layer SNN: LIF blocks stacked with hidden sizes [256, 128, 64], followed by a fully-connected readout layer.
- Readout: the temporal mean of spike rates $\bar{s} = \frac{1}{T_s}\sum_{t=1}^{T_s} s_t$, mapped to K gesture classes via a linear layer.

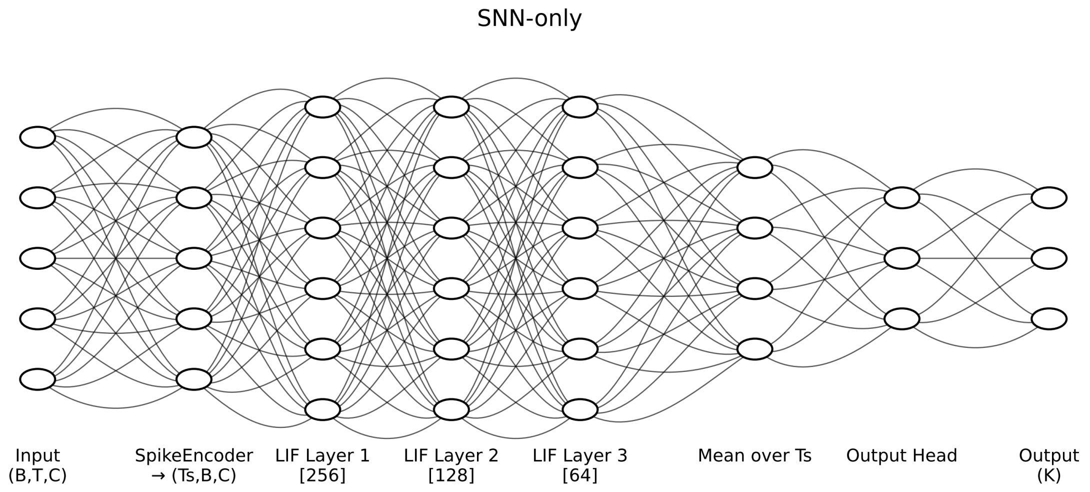


Figure 4.5: Architecture of the SNN.

4.3.5 Hybrid Fusion Model

Architecture

The Hybrid Fusion model combines the Temporal Convolutional Network (TCN) and the Spiking Neural Network (SNN) in a parallel configuration to exploit their complementary strengths. The TCN branch captures long-range temporal dependencies through dilated causal convolutions, while the SNN branch encodes sparse, event-driven temporal dynamics through spiking neurons. Similar parallel hybrid designs have been shown to improve robustness and efficiency in biosignal classification [69, 71]. The final representations from both branches are concatenated and passed through a fully connected classifier.

Input Representation

The sEMG input is windowed into segments of shape $(B, T = 200, C)$. For the TCN branch, the input is transposed to (B, C, T) to match the convolutional format. For the SNN branch, the same input is encoded into spike trains of shape (T_s, B, T, C), and frame-wise averaging yields the spiking representation.

Parallel Branches

- **TCN branch:** Input (B, T, C) is transposed to (B, C, T), followed by a TemporalConvNet with channel sizes [64, 128, 256]. The output is aggregated via global average pooling, producing $(B, 256)$.
- **SNN branch:** Input (B, T, C) is transformed by a SpikeEncoder $(T_s = 20)$, passed through stacked LIF layers with hidden sizes [256, 128, 64], and then averaged by mean spike-rate pooling, producing $(B, 256)$.

Fusion and Classifier Head

The feature vectors from both branches are concatenated to form $(B, 512)$:

$$f = [\bar{h}_{\text{tcn}}; \bar{s}_{\text{snn}}] \in \mathbb{R}^{512},$$

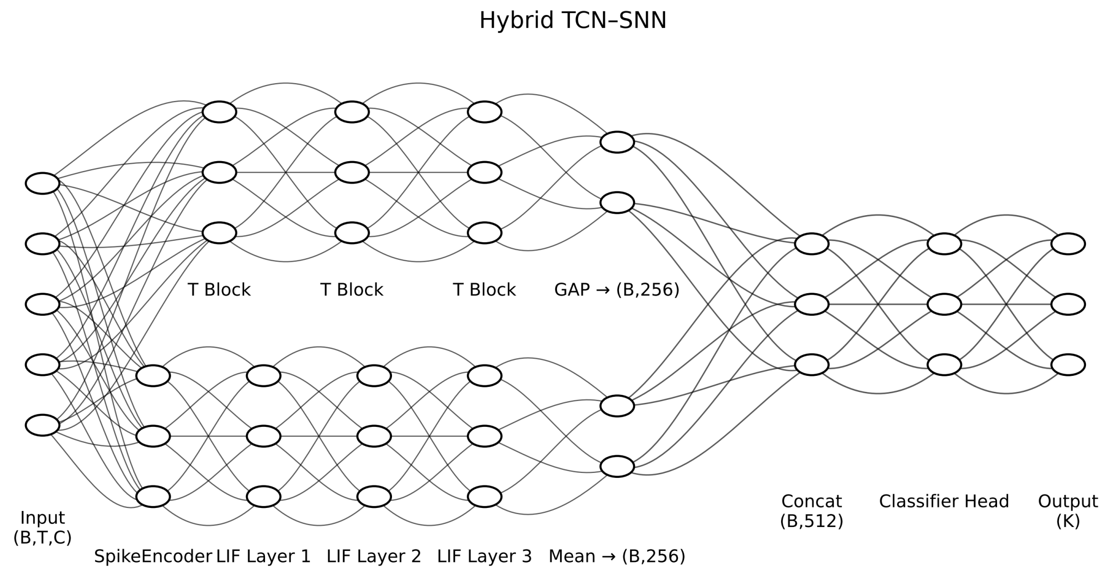


Figure 4.6: Hybrid TCN–SNN fusion: parallel feature extraction for classification

4.3.6 SpikingTCN

In this study, accuracy and efficiency were balanced in sEMG-based gesture recognition through the design of a hybrid SpikingTCN architecture. The model combines the long-range dependency modeling ability of the **Temporal Convolutional Network (TCN)** with the sparse, event-driven computation of the **Spiking Neural Network (SNN)**. By integrating convolutional parallel processing with temporal integration of membrane potentials in LIF neurons, the model captures temporal patterns accurately while maintaining low computational cost. Prior studies such as Spike-TCN [63], MFSNN [70], and iSpikformer [68] have clearly demonstrated that the integration of spiking structures enables a favorable balance between accuracy and efficiency. Building on this line of research on Spike-TCN, the proposed SpikingTCN is specifically tailored for real-time sEMG classification.

Input Representation

The windowed sEMG input has tensor shape (B, T, C), where B denotes batch size, $T = 200$ the window length, and C the number of channels. A SpikeEncoder transforms the input into spike sequences, yielding (T_s, B, T, C) (with $T_s = 20$ in this study). For each frame $t \in \{1, \ldots, T_s\}$, the spike input (B, T, C) is transposed to (B, C, T) before entering the TCN.

Spiking Temporal Block (STB)

The STB integrates **dilated causal convolutions** with LIF neurons and serves as the fundamental module. Weight normalization is applied to each convolution, and right-padding followed by a Chomp operation enforces strict causality.

(a) *Computation flow:*

$$\mathbf{h}_t^{(1)} = \mathrm{Conv1D}_1(\mathbf{x}_t; k, d), \quad \mathbf{s}_t^{(1)} = \mathrm{LIF}_1(\mathbf{h}_t^{(1)}),$$

$$\mathbf{h}_t^{(2)} = \mathrm{Conv1D}_2(\mathbf{s}_t^{(1)}; k, d), \quad \mathbf{s}_t^{(2)} = \mathrm{LIF}_2(\mathbf{h}_t^{(2)}),$$

$$\mathbf{y}_t = \mathrm{Dropout}(\mathbf{s}_t^{(2)}) + \mathrm{Res}(\mathbf{x}_t).$$

Here, $k = 3$ and $d = 2^i$ denote the kernel size and dilation rate of the i-th block, respectively. The residual connection Res is adjusted using a 1×1 convolution when required.

(b) *LIF dynamics:*

$$V_t = \beta V_{t-1} + I_t, \qquad S_t = \Theta(V_t - V_{\mathrm{th}})$$

with $\beta = 0.9$ and $V_{\mathrm{th}} = 1.2$. The non-differentiable thresholding function Θ is approximated by a **fast-sigmoid surrogate gradient** (slope = 25). Membrane potentials are preserved across frames ($t = 1, \ldots, T_s$), achieving temporal integration.

Overall SpikingTCN Network

Multiple STBs are stacked with dilation factors $d = 2^i$ to capture both short- and long-term dependencies. The final block output $(B, C_{\mathrm{last}}, T)$ is aggregated via global average pooling:

$$\mathbf{z}_t = \frac{1}{T} \sum_{\tau=1}^{T} \mathbf{y}_{t,\tau},$$

producing frame-level feature vectors. Frame-wise classification scores are then computed as

$$\hat{\mathbf{y}}_t = \mathrm{Classifier}(\mathbf{z}_t), \qquad \hat{\mathbf{y}} = \frac{1}{T_s} \sum_{t=1}^{T_s} \hat{\mathbf{y}}_t,$$

where the final prediction is the mean of logits across T_s frames. The classifier consists of

3 Layers [Linear(256, 128, 64) $\rightarrow$ ReLU $\rightarrow$ BN $\rightarrow$ Dropout(0.2)] $\rightarrow$ Output(K)

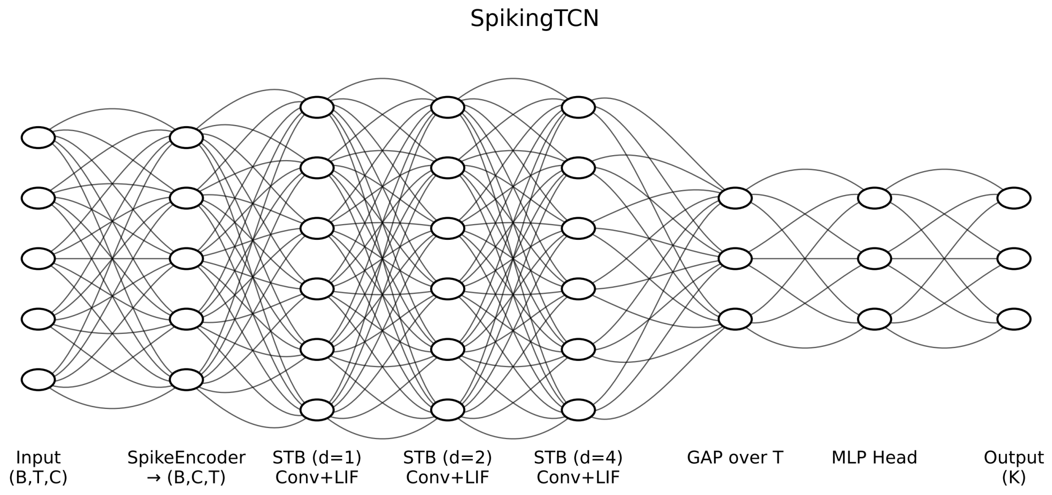


Figure 4.7: Architecture of the SpikingTCN.

4.4 Training & Reporting

- **Loss Function**: Cross-Entropy Loss
- **Optimiser**: Adam ($lr = 10^{-3}$)
- **Scheduler**: ReduceLROnPlateau (`mode='max'`, factor=0.5, patience=10)
- **Early Stopping**: patience = 20, based on validation accuracy
- **Evaluation Metrics**: test accuracy, macro-/weighted-F1 scores, and confusion matrices
- **Visualization**: training/validation accuracy and loss curves, spike histograms, and raster plots (for SNNs)

All models were trained under identical settings (batch size 32, shuffled data loader, Adam optimiser with $lr = 10^{-3}$, ReduceLROnPlateau, and early stopping). Evaluation on the test set reported Accuracy, F1-macro, and F1-weighted. Additional visualizations, including training/validation curves and spike statistics (firing-rate histograms and raster plots), are provided in the appendix to discuss convergence stability and spiking dynamics.

Additionally, the values of β together with the firing threshold were calibrated such that the mean firing rate for each encoding scheme remained within the range of approximately 1-26%. The simulation step count (T_s) was further adjusted to be incorporated into as many experimental sets as possible, with tests conducted under diverse conditions. These settings enabled control of sparsity through firing rates to reflect energy efficiency, while also facilitating a systematic comparison of how encoding schemes and variations in T_s influence both accuracy and efficiency.

The TCN captures temporal correlations robustly using dilated convolutions and attention, while the SNN leverage spike-based temporal coding for sparsity and temporal precision.

By comparing these models under the same preprocessing, partitioning, and optimization framework, this study empirically explores the accuracy–efficiency trade-off in sEMG-based gesture recognition.

Chapter 5

Experiments and Results

This chapter provides a comprehensive evaluation of the proposed spiking-based architectures for sEMG-driven gesture recognition, with particular emphasis on the SpikingTCN and Hybrid TCN–SNN models. For systematic benchmarking, comparisons are also drawn against baseline classifiers, including SNN-only, TCN-only, and LSTM.

The analysis is organized into sections that examine different aspects of model performance and applicability within neuroprosthetic control systems. First, we report standard classification metrics namely Top-1 Accuracy and macro-/weighted-F1 scores to quantify recognition capability. Second, we investigate energy efficiency through spike-based statistics such as total spike counts and mean firing rates, which serve as proxies for computational cost on embedded neuromorphic hardware. Furthermore, the experiments analyze the influence of spike encoding schemes (rate, latency, delta) and temporal resolution (T_s), offering insights into the trade-off between temporal fidelity and efficiency.

Taken together, these evaluations aim to establish the effectiveness of the proposed models, while simultaneously assessing their suitability as spiking neural controllers for real-time neuroprosthetic systems, where accuracy, temporal stability, and energy constraints are intrinsically coupled.

5.1 Comparison of Feature Combinations with in Models

5.1.1 LSTM

Experimental Setup

The Long Short-Term Memory (LSTM) classifier was implemented as a baseline model to capture the sequential patterns of sEMG signals. The network architecture consisted of two stacked LSTM layers with a hidden size of 128, a dropout rate of 0.2, and bidirectional recurrence enabled. The final hidden states from the bidirectional LSTM were concatenated and passed through a fully connected classifier ($128 \rightarrow 64 \rightarrow 8$), which incorporated ReLU activations, dropout, and batch normalization to produce the final outputs.

The input sEMG data were segmented into windows of shape $(B, T = 200, C)$, where each timestep represented multichannel EMG recordings. The LSTM encoded each sequence into hidden representations, and the last hidden state of the final layer was used for classification.

Experimental Results

Table 5.1 reports the performance of the LSTM classifier on the NinaPro DB6 dataset under identical preprocessing (bandpass filtering, window segmentation, and z-score normalization) and training conditions. Evaluation metrics included Top-1 accuracy and macro-F1 scores.

The LSTM achieved stable recognition accuracy across gesture classes, with macro-F1 scores averaging around 69.01%. The recurrent memory mechanism effectively captured temporal dependencies inherent in muscle activation patterns. However, due to its sequential hidden state updates at each timestep, the LSTM incurred higher computational costs and inference latency compared to convolution-based models.

Table 5.1: LSTM classifier performance across three evaluation runs on NinaPro DB6.

Run	**Accuracy (%)**	**Macro-F1 (%)**
1	67.16	67.37
2	71.23	71.41
3	68.05	68.26
Average	68.81	69.01

Discussion and Conclusion

The results demonstrate that LSTM provides a strong sequential baseline for gesture classification, offering reliable accuracy. Nonetheless, its limitations, namely slow convergence, increased energy consumption, and substantial inference latency, make it less suitable for real-time, low-power scenarios. These findings underscore the need for more energy-efficient and parallelizable alternatives, such as TCN and SpikingTCN, to better meet the demands of wearable and neuromorphic neuroprosthetic systems.

5.1.2 TCN-only Model

Experimental Setup

The Temporal Convolutional Network (TCN) was implemented as a convolutional baseline for modelling the temporal dependencies of sEMG signals without the use of recurrent structures. The architecture comprised multiple stacked Temporal Blocks, each consisting of dilated causal convolutions, Chomp operations (right-side padding removal), residual connections, batch normalization, and dropout (0.2). The kernel size was set to 3, and the dilation factor increased exponentially across layers, thereby enabling the receptive field to cover long temporal spans while preserving causality.

An 8-head multi-head self-attention mechanism was applied to the final TCN feature sequence to learn the relative importance of long-term dependencies. Attention outputs were aggregated via global average pooling and subsequently passed through a fully connected classifier ($128 \to 64 \to 8$ classes) to generate final gesture predictions.

The input consisted of windowed sEMG signals of shape $(B, T = 200, C)$, where B is the batch size, T is the window length, and C is the number of EMG channels. Convolutional filters processed each window in parallel to produce a feature sequence of shape (B, T, F). This sequence was refined through attention and pooling before being passed to the classifier.

Experimental Results

Table 5.2 reports the performance of the TCN classifier on the NinaPro DB6 dataset under identical preprocessing conditions (bandpass filtering, window segmentation, and per-channel z-score normalization) and training setups. Evaluation metrics included Top-1 accuracy and macro-F1 scores.

The TCN(+SA) consistently outperformed the LSTM baseline, achieving an average macro-F1 score of approximately 85%. Its dilated causal convolutions effectively captured long-range temporal dependencies while enabling parallel computation, resulting in faster convergence and lower inference latency compared with the recurrent model.

Table 5.2: TCN classifier performance across three evaluation runs on NinaPro DB6.

Run	**Accuracy (%)**	**Macro-F1 (%)**
1	83.62	83.69
2	85.66	85.71
3	85.57	85.60
Average	84.95	85.00

Discussion and Conclusion

In the context of sEMG-based gesture recognition, the TCN(+SA) therefore provides a strong convolutional baseline. It demonstrated higher accuracy, faster convergence, and reduced inference latency relative to LSTM, attributable to its parallelizable convolutional structure and attention-based temporal feature aggregation. Nonetheless, unlike spiking architectures, the TCN does not exploit event-driven sparsity, limiting its energy efficiency on neuromorphic hardware. These limitations motivate the exploration of hybrid and spiking-based models, such as SpikingTCN and Hybrid TCN-SNN, which aim to combine efficiency with biological plausibility.

5.1.3 SNN-only

Experimental Setup

The SNN-only classifier was implemented using layers of Leaky Integrate-and-Fire (LIF) neurons activated by spike trains generated via a `SpikeEncoder`. The input data consisted of sEMG windows of shape $(B, T = 200, C)$, which were converted into spikes using three encoding schemes: *delta*, *latency*, and *rate*. For latency encoding, per-channel min-max normalization was applied within each window, with an optional threshold employed to suppress sub-threshold activations.

The network architecture comprised three hidden layers of sizes [64, 128, 256]. Each layer included a fully connected projection followed by an LIF neuron. Membrane potentials were integrated across simulation steps, and neurons employed a fast-sigmoid surrogate gradient. The leak factor (β) and firing threshold (V_{th}) were calibrated such that the average neuronal firing rate remained between 5-26%. To analyze the impact of temporal resolution, the number of spike simulation steps (T_s) was varied over $\{10, 14, 20, 30, 40, 50\}$.

Experimental Results

The trial results showed marked performance differences depending on the encoding scheme. Latency encoding consistently demonstrated the best recognition performance, achieving a peak macro-F1 of 62.8% at $T_s = 14$. It maintained stable accuracy of around 62% across the range $T_s \in \{10, 30\}$. In contrast, delta encoding achieved moderate accuracy of approximately 33%, but its performance varied substantially and training stability degraded as the number of simulation steps increased. Rate encoding performed poorly overall, yielding below 15% accuracy for most T_s values, with only a minor increase to 22.2% at $T_s = 50$.

Accuracy was also strongly influenced by temporal resolution (T_s). For latency encoding, performance improved steadily up to $T_s = 14$ to $T_s = 20$, beyond which accuracy plateaued or declined slightly due to spike noise accumulation and increased gradient variance. Delta encoding improved modestly up to $T_s = 20$, but exhibited no consistent gains thereafter, while rate encoding remained unsuitable regardless of T_s. Overall, the optimal operating range for latency encoding was confirmed to be between $T_s = 14$ and $T_s = 20$.

Encoding	$T_s = 10$	$T_s = 14$	$T_s = 20$	$T_s = 30$	$T_s = 40$	$T_s = 50$
Rate	14.34 / 3.58	14.91 / 8.70	14.02 / 4.16	14.34 / 3.58	14.34 / 3.58	22.17 / 17.31
Latency	61.78 / 61.55	62.84 / 62.81	62.10 / 62.08	62.51 / 62.18	60.39 / 59.97	52.49 / 52.76
Delta	33.66 / 33.15	31.13 / 30.39	33.90 / 33.27	33.90 / 32.36	33.66 / 32.99	23.96 / 20.84

Discussion and Conclusion

These results demonstrate that the SNN-only model can feasibly perform sEMG-based gesture recognition using purely spiking computation. However, performance depends critically on both the encoding scheme and temporal resolution. Latency encoding provided the most favorable trade-off between accuracy and efficiency, whereas rate encoding proved unsuitable and delta encoding offered only limited benefits. The high sensitivity of the SNN-only model to T_s highlights limitations in robustness and training stability. These findings support the need for hybrid architectures such as SpikingTCN, which combine convolutional feature extraction with spiking dynamics to reduce spike activity and improve stability during both training and inference. Such hybrid approaches represent a more practical solution for neuromorphic neuroprosthetic applications.

5.1.4 Hybrid TCN-SNN

Experimental Setup

The Hybrid TCN–SNN classifier was designed by integrating a convolutional temporal backbone (TCN) with a spiking branch (SNN), thereby combining high-capacity feature extraction with energy-efficient temporal integration.

Outputs from the TCN and SNN branches were processed in parallel and concatenated to form a joint representation, which was then passed through a fully connected classifier to produce gesture class predictions.

Experimental Results

The Hybrid TCN–SNN model consistently exhibited high recognition performance across all three spike encoding schemes. At $T_s = 20$, delta encoding achieved the highest Macro-F1 (87.83%), while rate (85.90%) and latency (86.81%) encodings also delivered competitive results with only minor differences. Top-1 accuracy reflected the same ranking.

Performance steadily improved up to $T_s = 20$, reaching a peak accuracy of 87.78%. Beyond this point, accuracy plateaued at 87.04% for both $T_s = 30$ and $T_s = 50$, while it declined to 82.07% at $T_s = 40$. This trend suggests that spike noise accumulation and increased gradient variance caused the degradation. An intermediate range of approximately 14–20 steps therefore provides the optimal temporal integration.

Table 5.3: Hybrid TCN-SNN performance by spike encoding at T_s=20 (Macro-F1, %).

Encoding	**Rate**	**Latency**	**Delta**
Macro-F1 (%)	85.90	86.81	87.83

Effect of temporal resolution. Varying T_s shows that performance improves up to T_s=20 and then saturates. This indicates that the SNN branch benefits from moderate temporal integration while the TCN branch stabilizes the overall optimization.

Table 5.4: Hybrid TCN-SNN Top-1 Accuracy (%) across spike simulation steps T_s.

T_s	**10**	**14**	**20**	**30**	**40**	**50**
Accuracy (%)	86.15	84.11	87.78	87.04	82.07	87.04

Discussion and Conclusion

The Hybrid TCN–SNN consistently outperformed the SNN-only model and achieved results comparable to or surpassing the TCN-only model, while requiring substantially fewer spikes than a pure SNN. Among the encoding methods, delta encoding produced the best performance at $T_s = 20$, though rate and latency encodings also remain viable alternatives depending on hardware constraints.

Overall, the hybrid architecture demonstrates a practical solution that balances accuracy and efficiency for neuroprosthetic control. The TCN branch provides stable and parallelizable temporal feature extraction, while the SNN branch contributes event-driven integration and neuromorphic compatibility. This combination achieves an advantageous trade-off between accuracy and efficiency for real-time, low-power sEMG-based gesture recognition, making it a promising approach for wearable neuroprosthetic systems.

5.1.5 SpikingTCN

Experimental Setup

The SpikingTCN classifier extends the conventional TCN by integrating spiking dynamics within each convolutional block. Specifically, each Temporal Block incorporates a Leaky Integrate-and-Fire (LIF) neuron that maintains membrane potential across simulation steps.

Each block consists of dilated causal convolutions, LIF layers, and dropout, with membrane potentials accumulated over time. The leak factor was calibrated to keep the average neuronal firing rate within 5–26% on Table 5.7. The architecture used output channel sizes of $[64, 128, 256]$, followed by a fully connected classifier ($128 \rightarrow 64 \rightarrow 8$) for gesture prediction.

Experimental Restuls

Table 5.5 reports the performance of SpikingTCN with different spike encoding schemes at $T_s = 20$. Rate encoding achieved the highest accuracy and Macro-F1 (71.88% / 72.30%), followed by latency (62.35% / 62.21%) and delta (42.22% / 42.67%). This indicates that rate-based spike generation works most effectively within the SpikingTCN structure, while latency encoding—superior in the SNN-only experiments—degrades when combined with convolutional temporal blocks. Delta encoding produced the lowest performance, reflecting limited discriminative capacity.

Table 5.6 summarizes the effect of varying spike simulation steps (T_s). Performance improved steadily up to $T_s = 20$, peaking with rate encoding at 71.88%. Accuracy increased slightly at $T_s = 30$ (72.29%) and $T_s = 40$ (76.61%), but declined at $T_s = 50$ (75.96%). This suggests that beyond a certain range, performance gains are constrained by spike noise accumulation and gradient variance. Latency encoding benefited at intermediate resolutions ($T_s = 14, 20$), but did not scale as effectively as rate encoding at higher T_s.

Table 5.5: SpikingTCN performance by spike encoding at T_s=20 (Accuracy / Macro-F1, %).

Encoding	**Rate**	**Latency**	**Delta**
Accuracy / Macro-F1 (%)	71.88 / 72.30	62.35 / 62.21	42.22 / 42.67

Table 5.6: SpikingTCN Top-1 Accuracy (%) across spike simulation steps T_s.

T_s	**10**	**14**	**20**	**30**	**40**	**50**
Acc (%)	63.00 (latency)	65.28 (latency)	71.88 (rate)	72.29 (rate)	76.61 (rate)	75.96 (rate)

Discussion and Conclusion

Overall, SpikingTCN demonstrates that convolutional temporal feature extraction can be combined with spiking computation effectively. Unlike the SNN-only model, where latency encoding was optimal, SpikingTCN benefits most from rate encoding, highlighting the importance of interactions between spike generation schemes and convolutional filters. Temporal resolution proved critical, with peak performance in the 20–40 step range, beyond which additional steps provided no further benefits. Importantly, SpikingTCN consistently outperformed the SNN-only model in accuracy and F1 score, while requiring far fewer spikes for comparable discriminative performance.

5.2 Comparison Across Models

This section provides a comparative analysis of the best-performing configurations of all evaluated models: LSTM, SNN-only, TCN, Hybrid TCN-SNN, and SpikingTCN. The models are assessed based on their Top-1 Accuracy and Macro-F1 metrics, focusing on the highest-performing setups to determine which architecture offers the most robust capability for sEMG-based gesture recognition. Across all spiking models, the average neuronal firing rate was maintained within the range of 5-26%, ensuring biologically plausible yet energy-efficient operation.

5.2.1 Training Dynamics

Figure 5.1 depicts the training and validation dynamics under latency encoding. The SNN-only model gradually improved but plateaued at around 60%, reflecting its limited performance. In contrast, SpikingTCN did not converge as rapidly as TCN or Hybrid; however, it consistently achieved higher accuracy and more stable convergence than SNN-only.

SpikingTCN exhibited greater variance due to temporal membrane integration, but this substantially reduced the instability of purely spiking-based training and provided more practical reliability for real-world applications. TCN and Hybrid still converged fastest with stable validation accuracy, whereas LSTM showed overfitting after 40 epochs, as indicated by increasing validation loss.

These results reaffirm the robustness of convolutional backbones while highlighting that SpikingTCN delivers markedly improved performance and stability over SNN-only. This demonstrates that spiking architectures can offer a realistic trade-off between efficiency and stability.

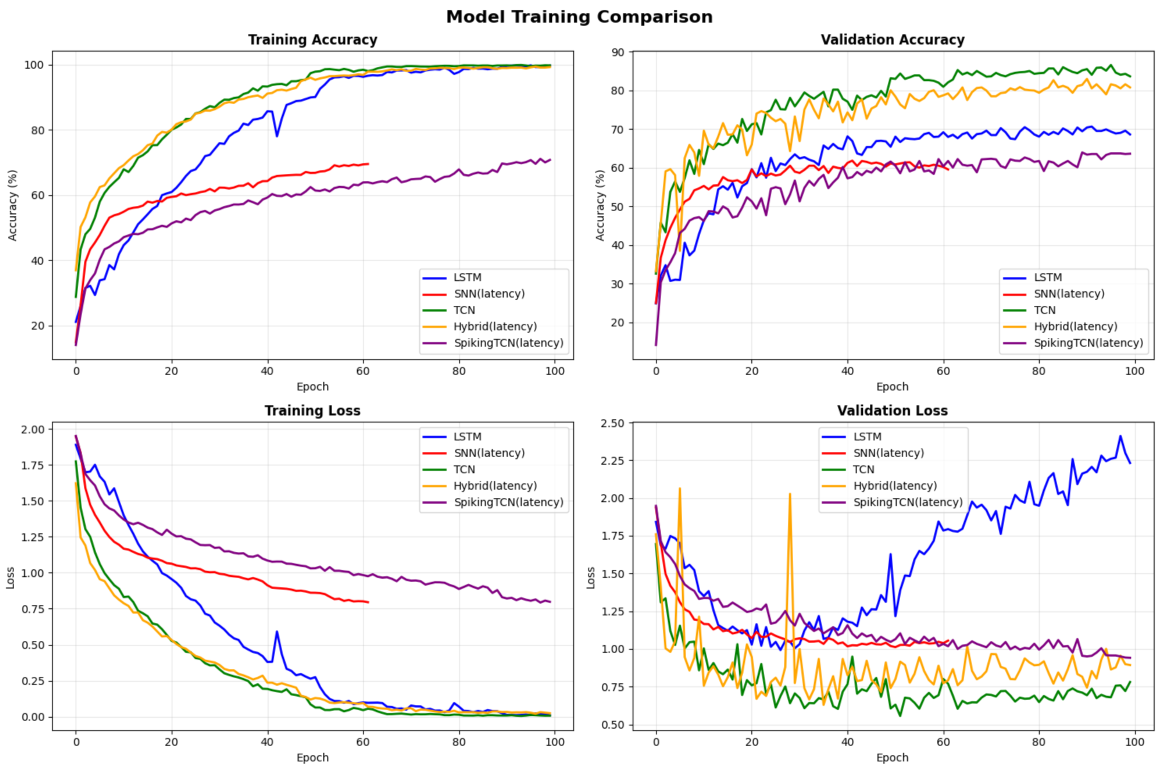


Figure 5.1: Training and validation curves for all models under latency encoding. TCN and Hybrid exhibit rapid and stable convergence, while SNN-only achieves moderate accuracy. LSTM suffers from overfitting, and SpikingTCN shows higher variance due to dense spiking dynamics.

5.2.2 Neuron firing-rate comparison

All spiking-based models, including SNN-only, Hybrid, and SpikingTCN, maintained mean firing rates between 1% and 26%. Since the dynamic power consumption of neuromorphic hardware is proportional to the number of spike and synapse events, lower firing rates directly translate into reduced energy expenditure [74, 75]. Table 5.7 compares the spike statistics and average firing rates of SNN-only, Hybrid TCN–SNN, and SpikingTCN models at $T_s = 20$ time steps. Here, Total Spikes denotes the average number of spikes per sample, while Mean Firing Rate represents the average neuronal firing rate expressed as a percentage. This relationship indicates that models with lower firing rates provide an energy advantage at comparable accuracy levels. Nevertheless, the Hybrid TCN–SNN, due to its parallel design, requires further validation to fully establish its energy efficiency benefits.

Table 5.7: Spike statistics across models and encodings ($T = 20$, $N = 256$). Spikes are averaged per sample; firing rate is mean per neuron.

Model	Encoding	Total Spikes	Mean Firing Rate(%)
SNN-only	rate	695	13.57
Hybrid	rate	280	5.47
SpikingTCN	rate	665	12.98
SNN-only	latency	963	18.81
Hybrid	latency	108	2.11
SpikingTCN	latency 8	945	18.46
SNN-only	delta	1122	21.91
Hybrid	delta	80	1.56
SpikingTCN	delta	1379	26.93

5.2.3 Model Comparisons

Table 5.8: Comparison of best models across different architectures.

Model Type	**Best Configuration**	**Accuracy (%)**	**General Notes**
LSTM	Hidden=128, 2 layers	71.23	Slower convergence
SNN-only	Latency, $T_s = 14$, Hidden=[64,128,256]	62.84	Energy-efficient but less accurate
TCN	Channels=[64,128,256], +Attention	85.66	Fast convergence, stable validation
Hybrid	Delta, $T_s = 20$	87.78	Best accuracy, parallelism
SpikingTCN	Rate, $T_s = 40$	76.61	Balanced spiking, accuracy

LSTM

The LSTM baseline demonstrated strong capacity for sequential dependency modelling, achieving solid accuracy (71.23%). However, it required longer convergence time during both training and inference. Latency was greater than that of convolutional models, restricting its suitability for real-time applications.

SNN-only

The performance of the SNN-only model was competitive, attaining 62.84% accuracy with latency encoding at $T_s = 20$. It benefited from event-driven computation and a relatively sparse firing rate (5–20%), ensuring high energy efficiency. However, instability during training and reduced discriminative power limited its effectiveness as a standalone solution.

TCN

TCN consistently achieved high accuracy (85.66%) with the additional advantage of rapid convergence, enabled by attention-based temporal aggregation and dilated causal convolutions.

The model performed in a stable manner and inferred quickly, but it did not exploit event-driven efficiency.

Hybrid TCN-SNN

The Hybrid TCN–SNN model achieved the best overall results with 87.8% accuracy and strong macro-F1 when using delta encoding. It effectively reduced spike activity, particularly average firing rates across neurons (1–15%), by relying on the TCN branch for feature extraction while maintaining neuromorphic compatibility through spiking integration. This architecture provides a practical balance between accuracy and efficiency, making it well suited for low-power, real-time neuroprosthetic applications.

SpikingTCN

SpikingTCN integrated spiking dynamics directly within convolutional temporal blocks, attaining moderate accuracy (76.61%) with rate encoding. This required higher firing rates (10–26%). Compared with the SNN-only model, it consistently performed better, confirming the value of convolutional–spiking integration. It represents a biologically plausible compromise between precision and efficiency, although it incurred higher spike costs than the Hybrid.

5.2.4 Accuracy–energy efficiency trade-off

The core objective of this study is to investigate the trade-off between energy efficiency and gesture recognition accuracy across diverse architectures. Since energy consumption on neuromorphic hardware is proportional to the number of spike and synapse events, this study employed mean firing rate as practical energy indicators.

Accordingly, the evaluation extended beyond classification performance to also include spiking activity itself. The SNN-only model, owing to its sparse firing and shallow network structure, was expected to exhibit the lowest theoretical energy consumption; however, its recognition accuracy remained limited (61.7%). In contrast, LSTM and TCN achieved higher accuracies of 71.2% and 85.0%, respectively, reflecting their strengths in sequential learning. Nevertheless, both architectures require spiking conversion for deployment in neuroprosthetic environments, and the associated computational overhead is likely to constrain their applicability in real-time, low-power scenarios. The Hybrid TCN–SNN architecture achieved high accuracy (87.8%) alongside relatively low spike activity, thus providing the most favorable balance between accuracy and energy efficiency. That said, the parallel structure may incur notable conversion costs in the TCN branch, leaving scope for further refinement. For example, confidence gating—where the activation of the TCN branch is selectively governed by the SNN branch's confidence—could enhance its practical applicability in real neuroprosthetic control. Meanwhile, SpikingTCN achieved 76.6% accuracy with moderate spike costs, indicating its potential as a practical trade-off for lightweight applications.

Figure 5.2 illustrates the overall trade-off between firing rate and accuracy. Across all spiking-based models, higher firing rates were consistently associated with improved accuracy, suggesting that increased spiking activity enhances signal representation and thereby classification performance. However, as higher firing rates directly translate into increased energy consumption, maximizing accuracy alone does not yield an efficient design. Instead, architectural strategies must balance accuracy improvements with firing sparsity.

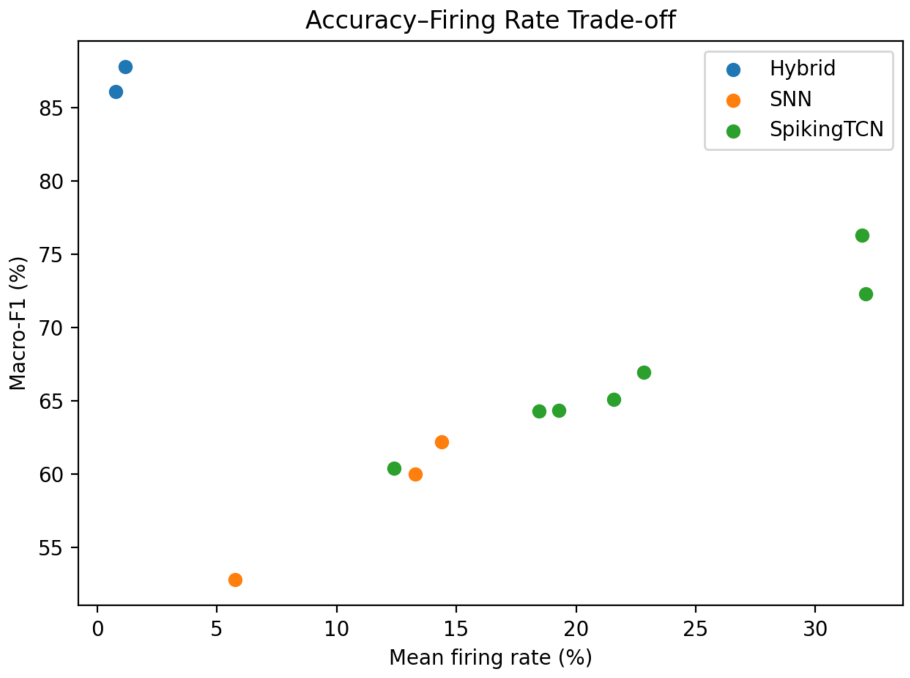


Figure 5.2: Accuracy–firing rate trade-off.

In summary, convolution-based architectures such as TCN, Hybrid, and SpikingTCN ensure stable optimization and high accuracy, while purely spiking architectures provide energy efficiency at the expense of discriminative power. The Hybrid TCN–SNN demonstrates both high accuracy and energy efficiency, though further validation in real neuroprosthetic environments is required. SpikingTCN, while less accurate, offers a realistic compromise by combining biological plausibility with adequate performance.

5.2.5 Discussion

The comparative analysis highlights the limitations and distinct strengths of each model architecture. LSTM functions as a strong sequential baseline, effectively capturing temporal dependencies in sEMG signals. However, because it relies on recurrent updates, its convergence is slower and inference latency higher, making it less suitable for real-time, resource-constrained neuroprosthetic systems. In contrast, TCN demonstrates robust performance with rapid convergence, benefiting from dilated causal convolutions and attention mechanisms to achieve consistently high accuracy (85%). Yet, its inability to exploit event-driven sparsity limits its energy efficiency on neuroprosthetic hardware.

SNN-only models leverage spiking computation to achieve energy efficiency through sparse event-driven firing. Latency encoding proved most effective among encoding schemes, achieving macro-F1 scores above 60%. However, the SNN-only approach exhibited instability and high sensitivity to temporal resolution (T_s), with performance degrading at longer simulation steps. These limitations constrain its applicability as a reliable standalone solution for neuroprosthetic control.

The Hybrid TCN–SNN model consistently outperformed both SNN-only and TCN-only models, achieving the best overall performance with a macro-F1 of 87.8% under delta encoding ($T_s = 20$). By combining convolutional temporal feature extraction with spiking-based integration, the hybrid design reduced spike counts by more than an order of magnitude compared to SNN-only while maintaining neuroprosthetic plausibility. The TCN branch provides parallelizable and stable optimization, while the SNN branch preserves biological relevance. However, reduced firing rates alone cannot be considered conclusive evidence of energy efficiency, since the Hybrid structure processes TCN and SNN branches in parallel before concatenation. Verification using hardware measurements or neuroprosthetic simulators is therefore required.

SpikingTCN occupies an intermediate position by embedding LIF-based spiking dynamics directly within convolutional blocks. Whilst it did not reach the performance level of the Hybrid model, SpikingTCN consistently surpassed the SNN-only baseline, achieving accuracies above 76.61% with rate encoding. Its performance peaked in the $T_s = 20$–40 range but declined beyond this due to spike noise accumulation and increased gradient variance. These findings emphasize the importance of interactions between encoding schemes and convolutional filters: latency encoding was optimal in SNN-only, whereas rate encoding emerged as the most effective in SpikingTCN.

All spiking-based models maintained mean neuronal firing rates within 1–26%, reflecting biologically plausible activity that directly correlates with reduced dynamic power consumption on neuroprosthetic hardware. As shown in Table 5.7, Hybrid TCN–SNN achieved the lowest spike activity (e.g., 80 spikes/sample under delta encoding, with a mean firing rate of 1.56%), whereas SNN-only generated the highest spike counts, exceeding 1100 spikes/sample. This suggests that the Hybrid design may offer the greatest energy efficiency; however, such claims must ultimately be validated through hardware experiments or simulator-based evaluation.

In summary, convolutional backbones (TCN) provide robust and stable training, while spiking architectures contribute neuromorphic compatibility and event-driven efficiency. Hybrid TCN–SNN emerges as the most practical design, combining accuracy, temporal stability, and potential energy efficiency, making it a strong candidate for real-time neuroprosthetic systems. Nonetheless, assertions regarding energy efficiency must be empirically validated. Although SpikingTCN did not achieve the same absolute performance as Hybrid, it consistently improved accuracy and stability over SNN-only, confirming its role as a meaningful middle-ground compromise. With its simpler integration of convolutional and spiking dynamics, SpikingTCN remains a promising direction for future research, particularly for lightweight and neuromorphic-friendly applications.

Chapter 6

Conclusion

This dissertation has presented an in-depth investigation into the design and evaluation of spiking-based neural controllers for sEMG-driven neuroprosthetic systems. By systematically benchmarking LSTM, TCN-only, SNN-only, SpikingTCN, and Hybrid TCN–SNN models, the study aimed to identify an architecture that balances accuracy, and energy efficiency—key requirements for real-time neuroprosthetic applications.

Extensive experiments on the NinaPro DB6 dataset revealed several important findings. Classical deep learning models such as LSTM and TCN demonstrated strong recognition capability, with TCN in particular achieving high accuracy and stable convergence due to its convolutional backbone. However, both models exhibited limited energy efficiency, constraining their deployment in neuromorphic and wearable contexts. In contrast, SNN-only models demonstrated low power consumption through sparse firing rates (5–20%), but their discriminative capacity and training stability remained restricted.

The Hybrid TCN–SNN architecture achieved the best overall performance, reaching 87.8% Top-1 accuracy and macro-F1 while reducing spike counts by more than an order of magnitude compared to SNN-only. This confirms that hybrid architectures can effectively combine the strengths of convolutional feature extraction and spiking efficiency, offering a highly practical solution for neuroprosthetic control systems.

SpikingTCN provided a biologically plausible compromise. Although moderately less accurate (76.6%), it consistently outperformed SNN-only while preserving event-driven computation and membrane integration dynamics. Notably, SpikingTCN achieved this within a simpler architecture compared to the Hybrid model, avoiding the parallel dual-branch structure. This highlights its potential as a lightweight and neuromorphic-friendly design, with scope for future refinement through optimized spike encoding, residual connections, and hardware-oriented pruning or quantization.

A key contribution of this work lies in its analysis of spike statistics and firing rates. All spiking-based models maintained firing rates within 1–26%, aligning with biologically plausible activity and the principle that energy consumption scales with spike/synapse events. For instance, the Hybrid TCN–SNN exhibited the lowest spike activity (e.g., 80 spikes/sample under delta encoding, with a mean firing rate of 1.56%), whereas SNN-only generated more than 1100 spikes/sample. While this suggests a favorable balance for the Hybrid design, claims regarding energy efficiency must ultimately be validated through hardware experiments or neuroprosthetic simulators.

This study also presents several limitations. While the parallel structure of the Hybrid model effectively improves accuracy, it may introduce computational redundancy. Moreover, the anticipated energy-saving effects require validation at the hardware level. In addition, since both training and evaluation were conducted on a single dataset, further verification on additional datasets is necessary to ensure broader generalizability.

Future research directions include: (1) deployment on neuroprosthetic platforms such as Intel Loihi and SpiNNaker to empirically validate energy efficiency, and (2) model compression and refinement through advanced hybrid techniques such as confidence gating and zero-spiking skipping.

In conclusion, this thesis has demonstrated that hybrid neural controllers, particularly the TCN–SNN fusion, represent a promising pathway toward real-time, energy-efficient, and biologically inspired neuroprosthetic systems. At the same time, SpikingTCN shows meaningful potential as a lightweight and neuromorphic-friendly alternative, offering a valuable middle ground between accuracy and efficiency. These findings contribute both to the academic understanding of spiking-based time-series modeling and to the practical development of next-generation assistive technologies capable of operating under strict power and latency constraints.

Bibliography

[1] Akram Fatayer, Wenpeng Gao, and Yili Fu. semg-based gesture recognition using deep learning from noisy labels. *IEEE journal of biomedical and health informatics*, 26(9):4462–4473, 2022.

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.

[3] Ankur Gupta, Nikolaos Vardalakis, and Fabien B. Wagner. Neuroprosthetics: from sensorimotor to cognitive disorders. *Communications biology*, 6(1):14–17, 2023.

[4] Muhammad Al-Ayyad, Hamza Abu Owida, Roberto De Fazio, Bassam Al-Naami, and Paolo Visconti. Electromyography monitoring systems in rehabilitation: A review of clinical applications, wearable devices and signal acquisition methodologies. *Electronics (Basel)*, 12(7):1520–, 2023.

[5] Xu Zhang, Yonggang Qu, Gang Zhang, Zhiqiang Wang, Changbing Chen, and Xin Xu. Review of sEMG for exoskeleton robots: Motion intention recognition techniques and applications. 25(8):2448–. Place: Switzerland Publisher: MDPI AG.

[6] Isabella Campanini, Catherine Disselhorst-Klug, William Z. Rymer, and Roberto Merletti. Surface emg in clinical assessment and neurorehabilitation: Barriers limiting its use. *Frontiers in neurology*, 11:934–, 2020.

[7] Mingde Zheng, Michael S. Crouch, and Michael S. Eggleston. Surface electromyography as a natural human-machine interface: A review. *IEEE sensors journal*, 22(10):9198–9214, 2022.

[8] Afroza Sultana, Farruk Ahmed, and Md. Shafiul Alam. A systematic review on surface electromyography-based classification system for identifying hand and finger movements. *Healthcare analytics (New York, N.Y.)*, 3:100126–100126, 2023.

[9] Dario Farina, Roberto Merletti, and Roger M Enoka. The extraction of neural strategies from the surface emg. *Journal of applied physiology (1985)*, 96(4):1486–1495, 2004.

[10] Crefeda Faviola Rodrigues, Graham Riley, and Mikel Lujan. Energy predictive models for convolutional neural networks on mobile platforms. Place: Ithaca Publisher: Cornell University Library, arXiv.org.

[11] Yonghui Zhang, Yugen Yi, Jiangyan Dai, Huixiang Zhang, Peng Zhang, Chunlei Chen, Huihui Zhang, and Malik Jahan Khan. Deep learning on computational-resource-limited platforms: A survey. *Mobile information systems*, 2020(2020):1–19, 2020.

[12] Steve. Furber and Petruț. Bogdan. *SpiNNaker - a Spiking Neural Network Architecture.* NowOpen. Now Publishers, Norwell, MA, 1st ed. edition, 2020.

[13] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[14] Filip Ponulak, Ponulak F, Andrzej Kasinski, and Kasinski A. Introduction to spiking neural networks: Information processing, learning and applications. *Acta neurobiologiae experimentalis JID - 1246675*, 71(4):409–33, 2011.

[15] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature (London)*, 575(7784):607–617, 2019.

[16] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. 2016.

[17] S Micera, J Carpaneto, and S Raspopovic. Control of hand prostheses using peripheral information. *IEEE reviews in biomedical engineering*, 3(1):48–68, 2010.

[18] Mikhail A. Lebedev and Miguel A. L. Nicolelis. Brain-machine interfaces: From basic science to neuroprostheses and neurorehabilitation. *Physiological reviews*, 97(2):767–837, 2017.

[19] Stanisa Raspopovic, Marco Capogrosso, Francesco Maria Petrini, Marco Bonizzato, Jacopo Rigosa, Giovanni Di Pino, Jacopo Carpaneto, Marco Controzzi, Tim Boretius, Eduardo Fernandez, Giuseppe Granata, Calogero Maria Oddo, Luca Citi, Anna Lisa Ciancio, Christian Cipriani, Maria Chiara Carrozza, Winnie Jensen, Eugenio Guglielmelli, Thomas Stieglitz, Paolo Maria Rossini, and Silvestro Micera. Restoring natural sensory feedback in real-time bidirectional hand prostheses. *Science translational medicine*, 6(222):222ra19–, 2014.

[20] Max Ortiz-Catalan, Enzo Mastinu, Paolo Sassu, Oskar Aszmann, and Rickard Brånemark. Self-contained neuromusculoskeletal arm prostheses. *The New England journal of medicine*, 382(18):1732–1738, 2020.

[21] Angkoon Phinyomark, Rami N. Khushaba, and Erik Scheme. Feature extraction and selection for myoelectric control based on wearable emg sensors. *Sensors (Basel, Switzerland)*, 18(5):1615–, 2018.

[22] K. Englehart and B. Hudgins. A robust, real-time control scheme for multifunction myoelectric control. *IEEE transactions on biomedical engineering*, 50(7):848–854, 2003.

[23] Weidong Geng, Yu Du, Wenguang Jin, Wentao Wei, Yu Hu, and Jiajun Li. Gesture recognition by instantaneous surface emg images. *Scientific reports*, 6(1):36571–, 2016.

[24] Wentao Wei, Yongkang Wong, Yu Du, Yu Hu, Mohan Kankanhalli, and Weidong Geng. A multi-stream convolutional neural network for semg-based gesture recognition in muscle-computer interface. *Pattern recognition letters*, 119:131–138, 2019.

[25] Carlo J. De Luca. The wartenweiler memorial lecture the use of surface electromyography in biomechanics. *Journal of biomechanics*, 27(6):724–724, 1994.

[26] Isabella Campanini, Catherine Disselhorst-Klug, William Z. Rymer, and Roberto Merletti. Surface emg in clinical assessment and neurorehabilitation: Barriers limiting its use. *Frontiers in neurology*, 11:934–, 2020.

[27] G. Staude and W. Wolf. Objective motor response onset detection in surface myoelectric signals. *Medical engineering physics*, 21(6):449–467, 1999.

[28] David Leserri, Nils Grimmelsmann, Malte Mechtenberg, Hanno Gerd Meyer, and Axel Schneider. Evaluation of semg signal features and segmentation parameters for limb movement prediction using a feedforward neural network. *Mathematics (Basel)*, 10(6):932–, 2022.

[29] Mohammad Abdoli-Eramaki, Caroline Damecour, John Christenson, and Joan Stevenson. The effect of perspiration on the sEMG amplitude and power spectrum. 22(6):908–913.

[30] Tutorial. surface emg detection in space and time: Best practices. *Journal of electromyography and kinesiology*, 2019.

[31] K. Englehart, B. Hudgin, and P.A. Parker. A wavelet-based continuous classification scheme for multifunction myoelectric control. *IEEE transactions on biomedical engineering*, 48(3):302–311, 2001.

[32] Jiale Du, Zunyi Liu, Wenyuan Dong, Weifeng Zhang, and Zhonghua Miao. A novel tcn-lstm hybrid model for semg-based continuous estimation of wrist joint angles. *Sensors (Basel, Switzerland)*, 24(17):5631–, 2024.

[33] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, Rocky Rhodes, Tiezhen Wang, and Pete Warden. Tensorflow lite micro: Embedded machine learning on tinyml systems. 2020.

[34] Manfredo Atzori and Henning Muller. The ninapro database: A resource for semg naturally controlled robotic hand prosthetics. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, volume 2015, pages 7151–7154. IEEE, 2015.

[35] Yu Du, Wenguang Jin, Wentao Wei, Yu Hu, and Weidong Geng. Surface emg-based inter-session gesture recognition enhanced by deep domain adaptation. *Sensors (Basel, Switzerland)*, 17(3):458–, 2017.

[36] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation (New York, N.Y.)*, 101(23):E215–220, 2000.

[37] Madalena Costa, George B. Moody, Isaac Henry, and Ary L. Goldberger. Physionet: an nih research resource for complex signals. *Journal of electrocardiology*, 36:139–144, 2003.

[38] Christoph Amma, Thomas Krings, Jonas Böer, and Tanja Schultz. Advancing muscle-computer interfaces with high-density electromyography. In *CHI 2015 : proceedings of the 33rd annual CHI Conference on Human Factors in Computing Systems : April 18-23, 2015, Seoul, Republic of Korea*, volume 2015-, pages 929–938, New York, NY, USA, 2015. ACM.

[39] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature (London)*, 521(7553):436–444, 2015.

[40] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. 8(1):53–74. Place: Cham Publisher: Springer International Publishing.

[41] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[42] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks, 2017.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2016-, pages 770–778. IEEE, 2016.

[45] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need.

[47] Yang Zhang, Caiqi Liu, Mujiexin Liu, Tianyuan Liu, Hao Lin, Cheng-Bing Huang, and Lin Ning. Attention is all you need: utilizing attention in AI-enabled drug discovery. 25(1). Place: England Publisher: Oxford Publishing Limited England.

[48] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature (London)*, 529(7587):484–489, 2016.

[49] Jim X. Chen. The evolution of computing: AlphaGo. 18(4):4–7. Place: New York Publisher: IEEE.

[50] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398–, 2021.

[51] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. 2012.

[52] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[53] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[54] Klaus Greff, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, and Jurgen Schmidhuber. Lstm: A search space odyssey. *IEEE transaction on neural networks and learning systems*, 28(10):2222–2232, 2017.

[55] Qingfeng Chen, Jing Wu, Feihu Huang, Yu Han, and Qiming Zhao. Multi-layer LSTM parallel optimization based on hardware and software cooperation. In Gerard Memmi, Baijian Yang, Linghe Kong, Tianwei Zhang, and Meikang Qiu, editors, *Knowledge Science, Engineering and Management*, pages 681–693. Springer International Publishing.

[56] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. 2015.

[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2016-, pages 770–778. IEEE, 2016.

[58] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):640–651, 2017.

[59] Saumya Dash. Green AI: Enhancing sustainability and energy efficiency in AI-integrated enterprise systems. 13:21216–21228. Place: Piscataway Publisher: IEEE.

[60] Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023.

[61] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks, 2019.

[62] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

[63] Changze Lv, Yansen Wang, Dongqi Han, Xiaoqing Zheng, Xuanjing Huang, and Dongsheng Li. Efficient and effective time-series forecasting with spiking neural networks.

[64] Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of neuroscience*, 18(24):10464–10472, 1998.

[65] Sang Ho Choi. Spiking neural networks for biomedical signal analysis. *Biomedical engineering letters*, 14(5):955–966, 2024.

[66] Hansol Choi, Jangsoo Park, Jongseok Lee, and Donggyu Sim. Review on spiking neural network-based ecg classification methods for low-power environments. *Biomedical engineering letters*, 14(5):917–941, 2024.

[67] Hybrid spiking neural network – transformer video classification model - StarPlus.

[68] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer.

[69] K. Anup Kumar and C. Vanmathi. A hybrid parallel convolutional spiking neural network for enhanced skin cancer detection. 15(1):11137–22. Place: London Publisher: Nature Publishing Group UK.

[70] Yu Song, Liyuan Han, Tielin Zhang, and Bo Xu. Multiscale fusion enhanced spiking neural network for invasive bci neural signal decoding. *Frontiers in neuroscience*, 19:1551656–, 2025.

[71] Bang Hu, Changze Lv, Mingjie Li, Yunpeng Liu, Xiaoqing Zheng, Fengzhe Zhang, Wei cao, and Fan Zhang. SpikeSTAG: Spatial-temporal forecasting via GNN-SNN collaboration.

[72] Naoya Muramatsu and Hai-Tao Yu. Combining spiking neural network and artificial neural network for enhanced image classification. 2021.

[73] Changze Lv, Dongqi Han, Yansen Wang, Xiaoqing Zheng, Xuanjing Huang, and Dongsheng Li. Advancing spiking neural networks for sequential modeling with central pattern generators. 2024.

[74] Alexander Henkes, Jason K. Eshraghian, and Henning Wessels. Spiking neural networks for nonlinear regression. *Royal Society open science*, 11(5):231606–23, 2024.

[75] Wolfgang Maass. To spike or not to spike: That is the question. *Proceedings of the IEEE*, 103(12):2219–2224, 2015.

Appendices

Appendix A

Appendix: Additional Figures by Encoding Method

Rate

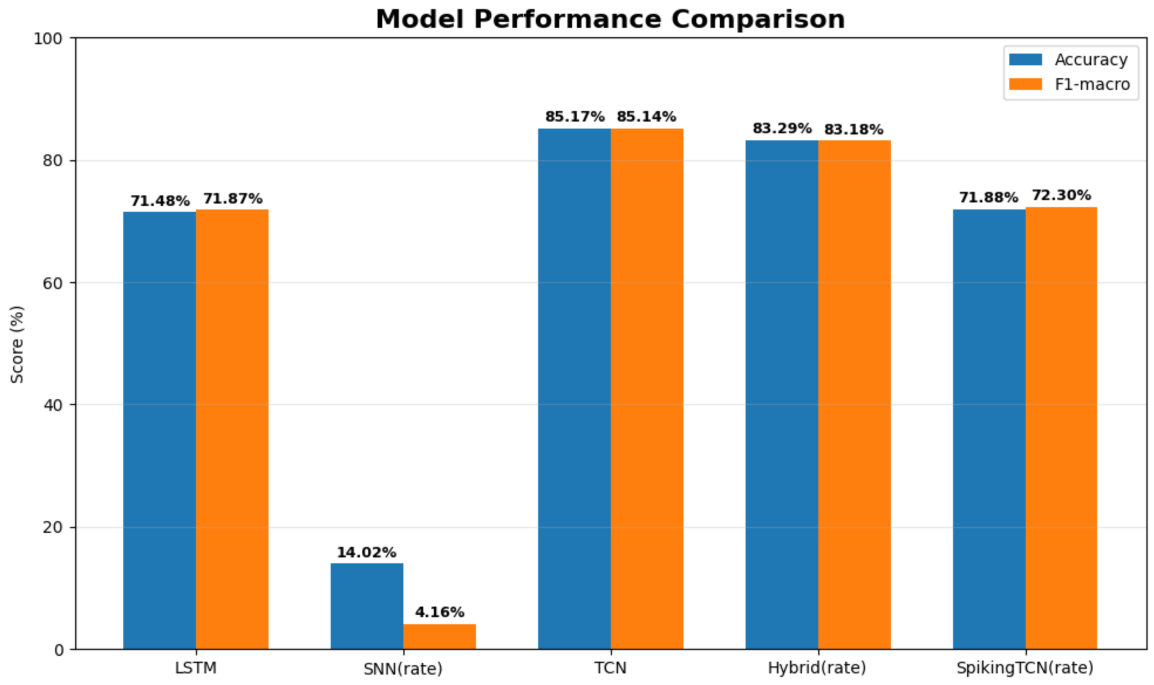


Figure A.1: Ts = 20, Encoding = rate

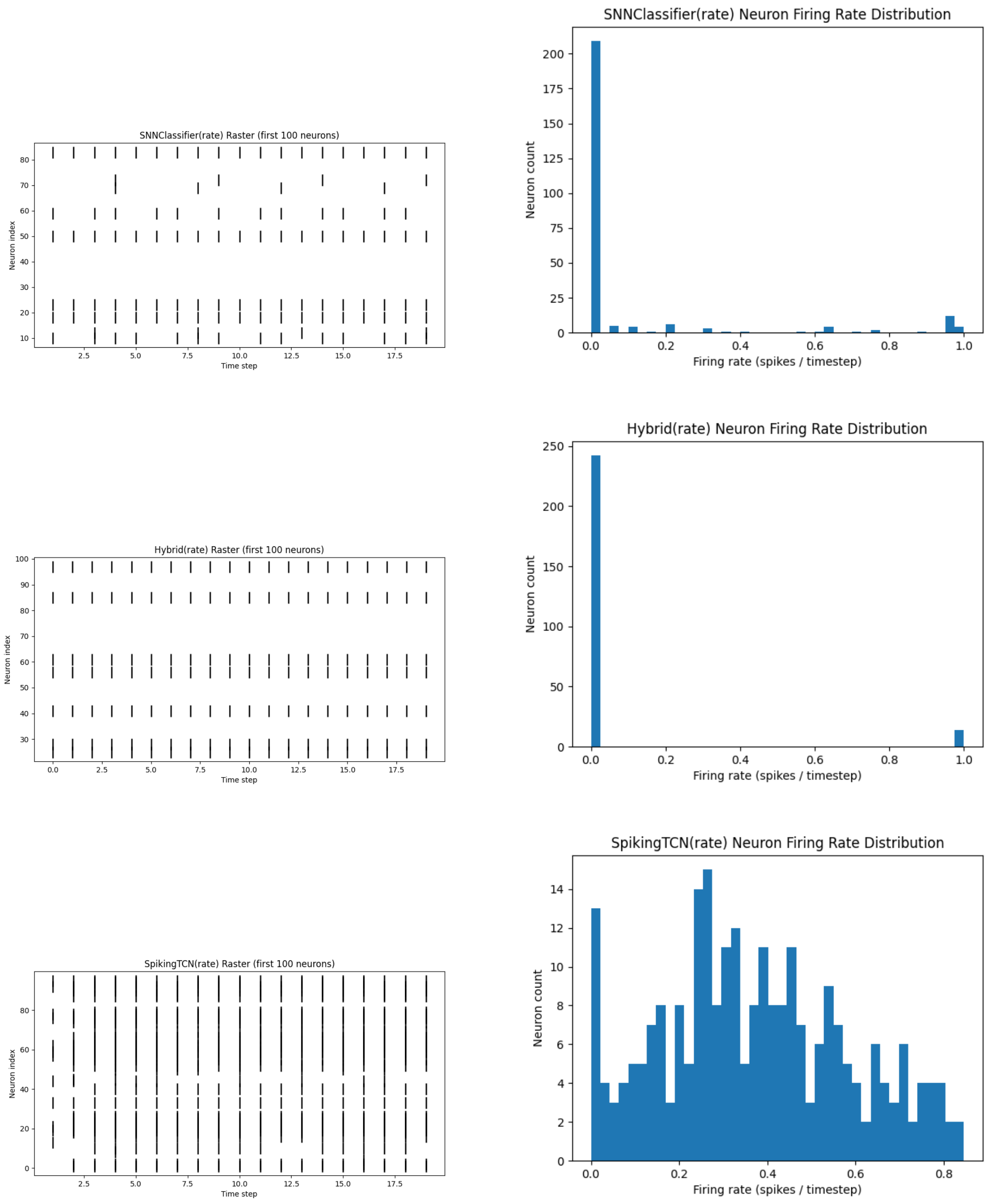


Figure A.2: Ts = 20, Encoding = rate

Latency

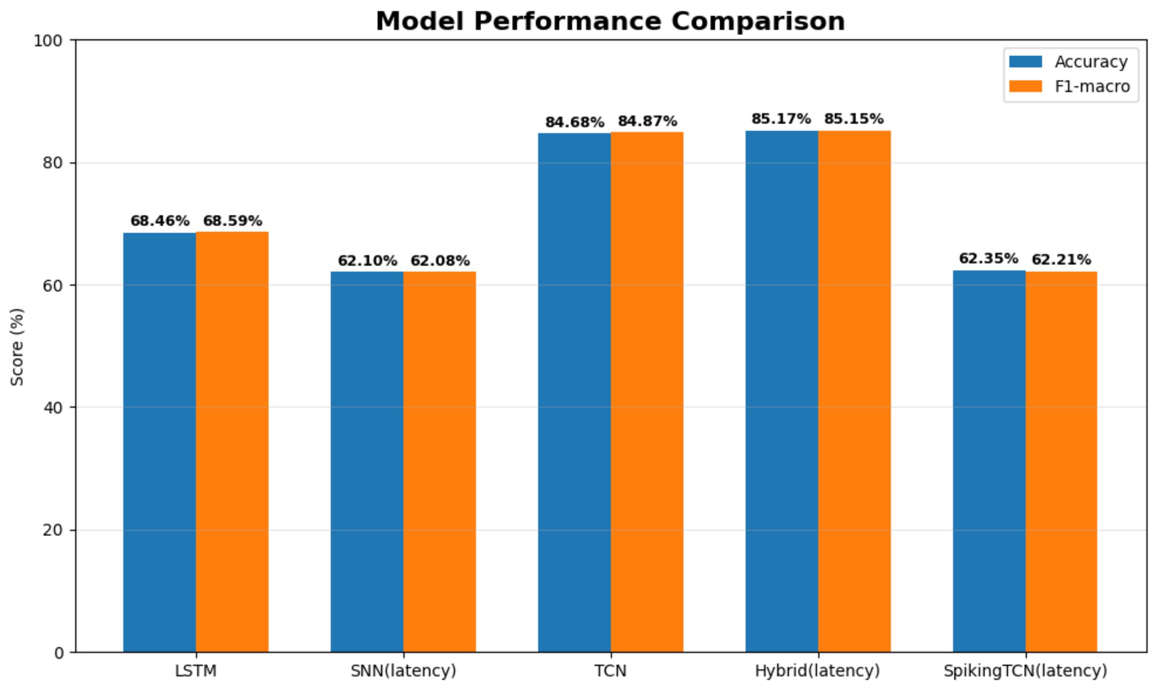


Figure A.3: Ts = 20, Encoding = latency

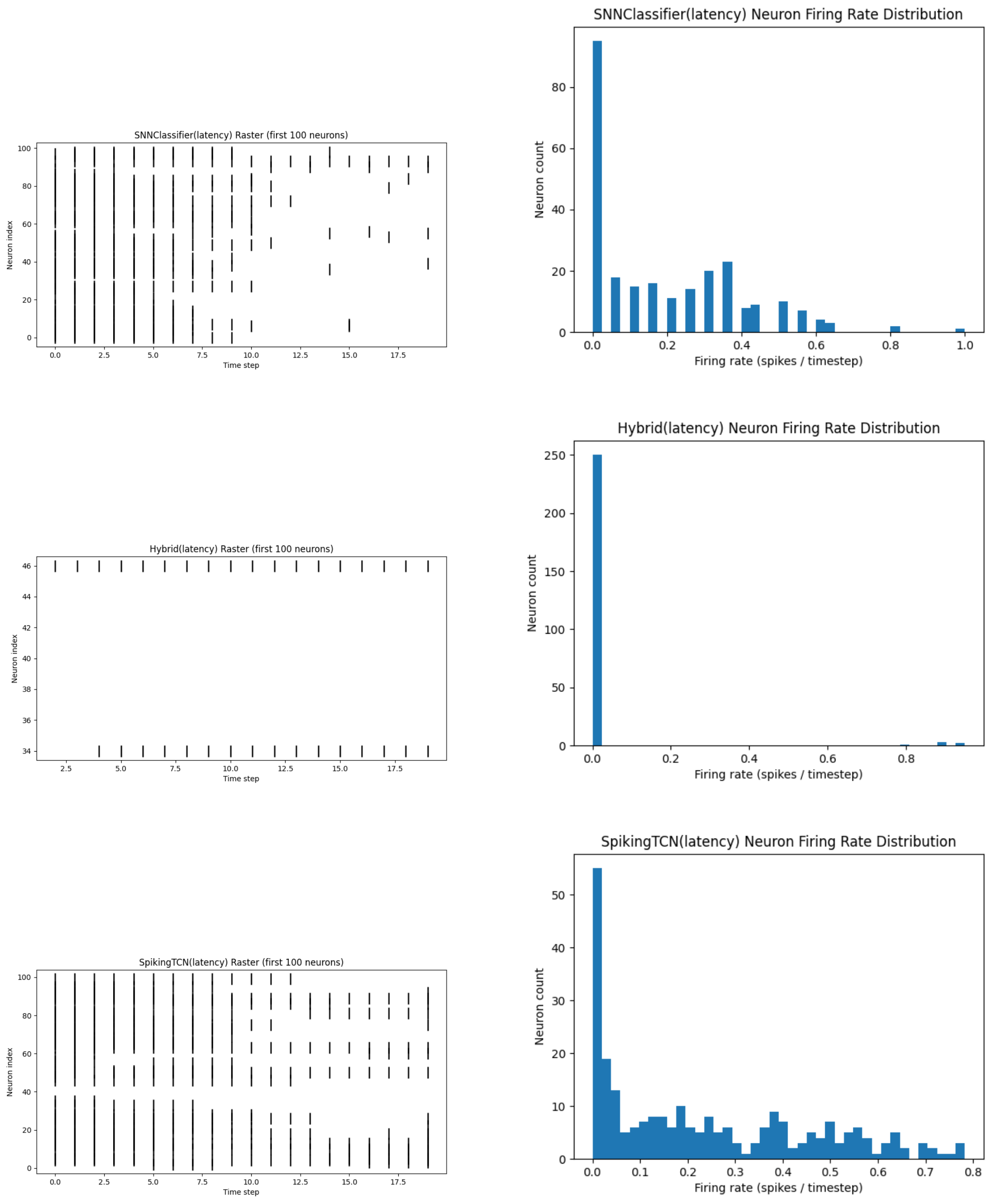


Figure A.4: Ts = 20, Encoding = latency

Delta

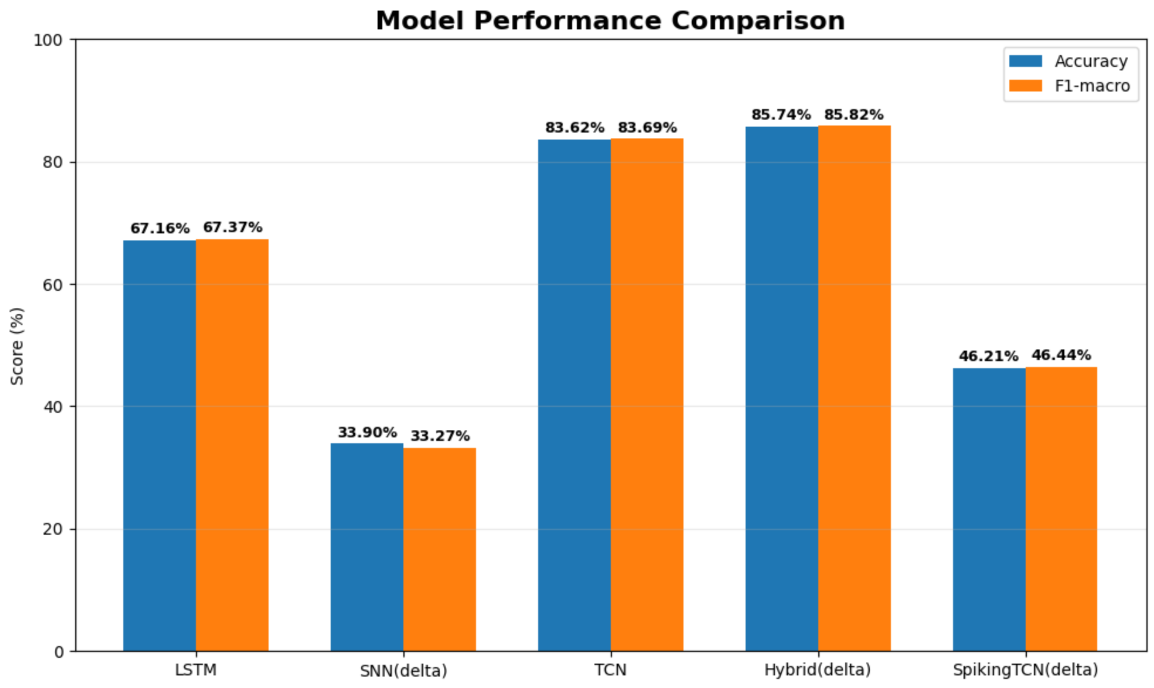


Figure A.5: Ts = 20, Encoding = delta

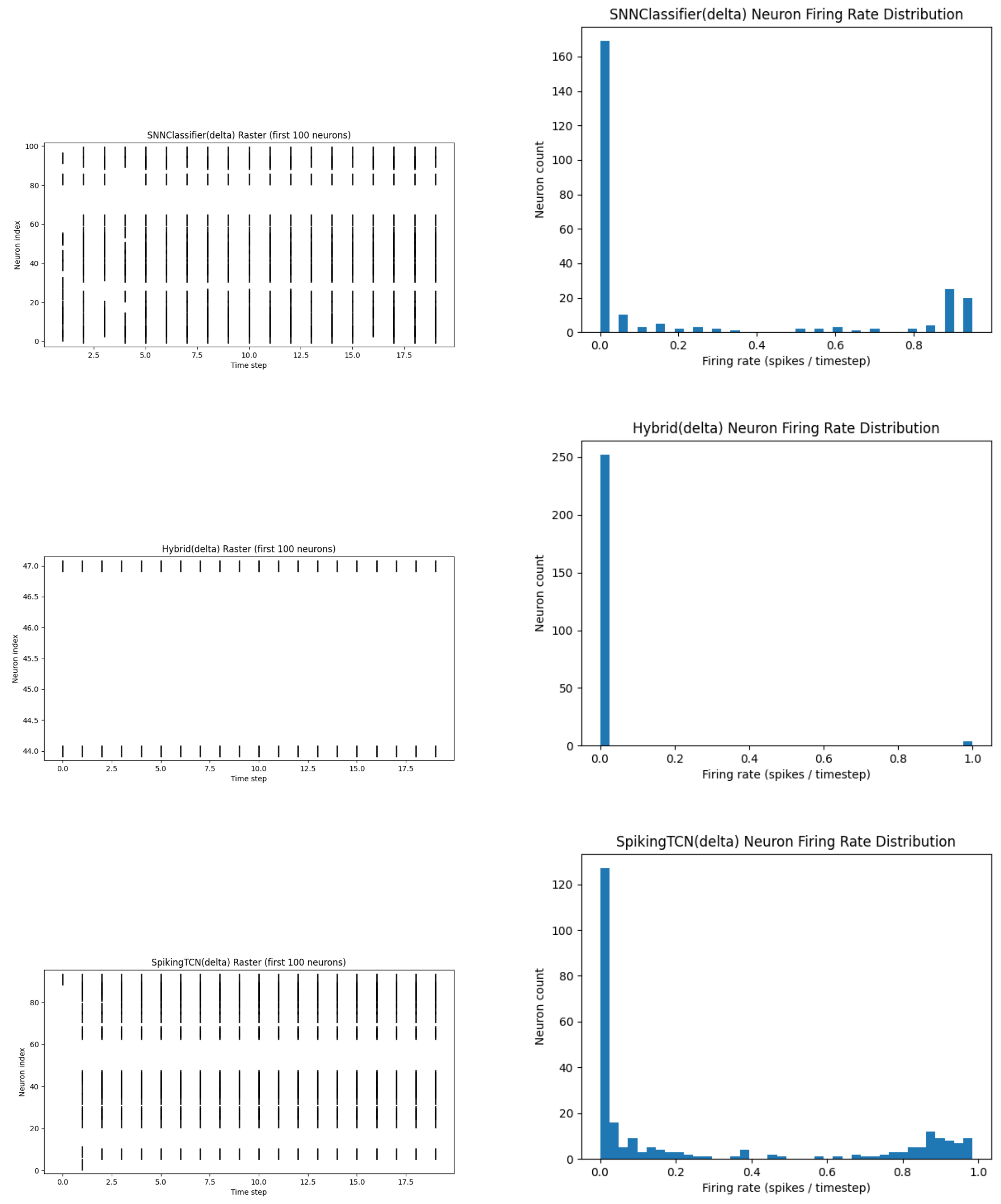


Figure A.6: Ts = 20, Encoding = delta