Training neuromorphic neural networks for speech recognition and Bayesian optimisation of network hyperparameters

Rihards klotiņš

Supervisor: Dorian Florescu

April 21, 2025

Abstract

Filler abstract - Spiking Neural Networks (SNNs), the third generation of neural networks, offer a promising alternative to traditional Artificial Neural Networks (ANNs) due to their energy efficiency, temporal processing capabilities, and potential for neuromorphic hardware implementation. Unlike ANNs, which rely on continuous-valued activations, SNNs process information through discrete spike events, closely mimicking biological neural systems. This characteristic enables them to efficiently handle sequential data, making them suitable for applications such as speech recognition, event-based vision, and edge computing. Training SNNs, however, remains a significant challenge due to the non-differentiability of spike events. While ANN-to-SNN conversion provides a workaround, it imposes computational costs and limits architectural flexibility. Direct training methods, such as Backpropagation-Through-Time (BPTT) with surrogate gradients, local learning rules, and biologically inspired approaches like e-prop and EventProp, have emerged as viable alternatives. Each method presents trade-offs in terms of computational complexity, biological plausibility, and performance. Notably, EventProp has demonstrated state-of-the-art results while reducing memory and computational overhead compared to BPTT. This work explores the theoretical advantages of SNNs, their energy-efficient processing, and recent advancements in training methodologies. It also highlights their potential impact on low-power computing applications, particularly in audio processing, where temporal encoding is crucial. By addressing current limitations and leveraging novel training strategies, SNNs could play a pivotal role in next-generation AI systems.

1 Introduction

Artificial Neural Networks (ANN) are computational models inspired by the brain; made up of layers of interconnected "neurons" which can learn patterns when given large amounts of data. After learning patterns, ANNs can be used to make inferences, predicting an output based on some given input. ANNs have long been used for application ranging from computer vision to financial forecasting. However the use of ANNs has surged recently, increasing fourfold since 2017, driven by the development of generative AI models like ChatGPT [1]. To compound this, models are growing in size and becoming more complex, consequently the global AI power consumption is increasing at an exponential rate [2]. Not only does this make models expensive to operate, but it also raises sustainability concerns.

A graphics processing unit (GPU) running a large language model (LLM) consumes hundreds of watts of power. On the other hand, the human brain processes sensory information, keeps involuntary biological systems functioning, runs its own language model, and enables conscious thought — all while using only about 20 watts. Such efficient information processing has motivated research into neuromorphic computing - a field aiming to emulate the brain's neural architecture. Spiking Neural Networks (SNNs) are considered the third generation of neural network models [3], where ANNs are considered the second generation. Compared to neurons in ANNs, SNN neurons more closely model how biological neurons behave in the brain. For a second generation neuron, the output is a weighted sum of the inputs passed through an activation function (see Figure 1 (a)). For a third generation neuron, the output is a spike, sometimes referred to as an event, which occurs when the voltage of a neuron surpasses a threshold (Figure 1 (b)), this results in neurons communicating with each other via series of spikes, which vary in timing and frequency. Since information is encoded in the timing of outputs, SNNs can inherently capture temporal patterns and dynamic behaviours in sequential data, making them inherently capable at processing tasks involving time-dependent signals, such as speech recognition, sensory processing, and event-based data streams. Moreover, it has been shown that

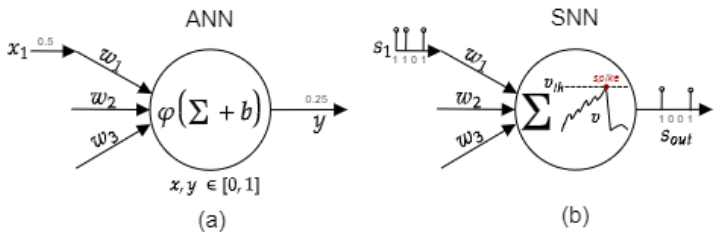

Figure 1: (a) ANN neuron (b) SNN neuron

SNNs theoretically posses higher computational power than the previous generation of ANNs [4]. Spiking neurons only activate when necessary, and don't require a clock signal, enabling massive power savings. For instance, the leading neuromorphic platform TrueNorth is capable of simulating a million spiking neurons in real-time while consuming 63 mW. The equivalent network executed on a high-performance computing platform was 100–200× slower than real-time and consumed 100,000 to 300,000× more energy per synaptic event [5]. The power savings of SNNs could be game-changing in edge computing devices that run on battery or have limited power budgets - like smartphones or remote environmental sensors. On top of this, their low-latency could benefit autonomous vehicles and robotics.

2 Spiking Neuron Model

2.1 Leaky Integrate-and-Fire Neuron

The leaky integrate-and-fire (LIF) neuron model is the most widely used neuronal model in SNNs due to its simplicity and efficiency which helps it scale for larger networks. The neuron receives weighted spiking inputs from other neurons as can be seen in figure **??** (b). Spikes received by a neuron at time t induce a current $X[t]$. This input current increases the membrane potential, $V[t]$, which is the voltage between the inside of the neuron and the outside. The voltage slowly decays over time, the speed of the decay depends on the membrane time constant, τ. This is the "Leaky Integrate" aspect of the neuron; $X[t]$ is integrated to give $V[t]$ at the same time as $V[t]$ leaks voltage. The neuron leaks voltage till it reaches it's baseline voltage level, V_{reset}. This behaviour can be seen in equation (1), $H[t]$ is equal equation (1)to $V[t]$, unless a spike occurs during t. This spiking logic is defined in equation (2), where $\Theta(x)$ is a function that is 0 unless $x \geq 0$. In other words this function becomes 1 only when the threshold voltage V_{th} is reached. When the threshold voltage is reached, $V[t]$ is set to V_{reset} and a spike is released; if the threshold voltage is not reached then $V[t] = H[t]$. The membrane potential behaviour at spike time is defined in equation (3).

$$H[t] = V[t-1] + \frac{1}{\tau}(X[t] - (V[t-1] - V_{reset})), \tag{1}$$

$$S[t] = \Theta(H[t] - V_{th}), \tag{2}$$

$$V[t] = H[t](1 - S[t]) + V_{reset}S[t] \tag{3}$$

Synapse Models

Literature discussing different parameters to adjust: firing threshold [6], membrane time constant [7], weights [8], activation function between charging and firing [9].

Other models of neurons include: **Parallel Spiking neuron model**: [10]
Izhikevich – efficiently reproduces diverse spiking patterns with low computational cost;
Hodgkin–Huxley – offers high biophysical accuracy but is computationally intensive.

Todo: Information coding In digital images, each pixel is stored as a number which represents the red, green, and blue light intensity at that point - a higher number indicates greater intensity of light. In digital audio, each timestep is encoded as a number which indicates the strength of the pressure perturbation at that particular time. So how can an image or audio be passed into a spiking neural network? In other words, how is it turned into spikes? Methods for coding information into spike trains can be classified into two groups - rate coding and temporal coding. A good place to start would be taking inspiration from the brain. Rate coding transfers information in the frequency of spikes at a given moment. Initially it was thought that rate coding was the predominant technique to transmit information within the nervous system [**?**], however

---

3 Training Neural Networks

The objective of training a neural network for spoken word recognition is to construct a model that accurately maps audio input to the correct lexical output. **Supervised learning** remains the dominant training paradigm in this domain, consistently achieving state-of-the-art results in speech recognition tasks [**? ?**]. Supervised learning relies on labelled datasets—typically composed of audio clips annotated with their corresponding transcriptions—providing explicit guidance for the model to learn the mapping between acoustic features and linguistic units. In contrast, unsupervised learning operates on unlabelled data, requiring the model to uncover inherent structure or patterns within the input without external annotation. Next I will discuss different ways that neural networks can be trained.

3.1 Backpropagation

Backpropagation is the most effective and widely used method for training second generation artificial neural networks. It provides a systematic way to adjust the internal parameters—or weights—of a network to minimize the discrepancy between the network's predictions and the desired outputs. An overview of the backpropagation process is as follows:

1. **Forward Pass**: The model receives an input and processes it through its layers, generating an output.
2. **Error Calculation**: The output is compared to the desired target, and a **loss function** quantifies the error.
3. **Gradient Computation**: The derivative of the loss function is computed with respect to each weight in the network using the **chain rule of calculus**.
4. **Weight Update**: The weights are adjusted by subtracting a fraction of their corresponding gradient, typically scaled by a learning rate. This step moves the model toward a configuration that reduces error.

In supervised learning, the desired output is the label of the data. When a model predicts an output, the "loss" is calculated to quantify the error between the actual output and the desired output. For instance, a common way to calculate the loss is by calculating the mean squared error (MSE), where you find the sum of the squared differences between the actual output neuron values and the desired output neuron values (Equation (4)).

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4}$$

To demonstrate how backpropagation works, let's consider this simplified, fully-connected network consisting of a layer of input neurons, hidden neurons, and output neurons (Figure 1). Fully-connected means each neuron in a layer is connected to all neurons in the following layer.

The highlighted neuron, h_1, is connected to all input neurons, x_1, x_2, and x_3. This means h_1 is a weighted sum of the input neurons, x (Equation 5).

$$h_1 = \sum_{i=1}^{3} x_i \times W_i^1 \tag{5}$$

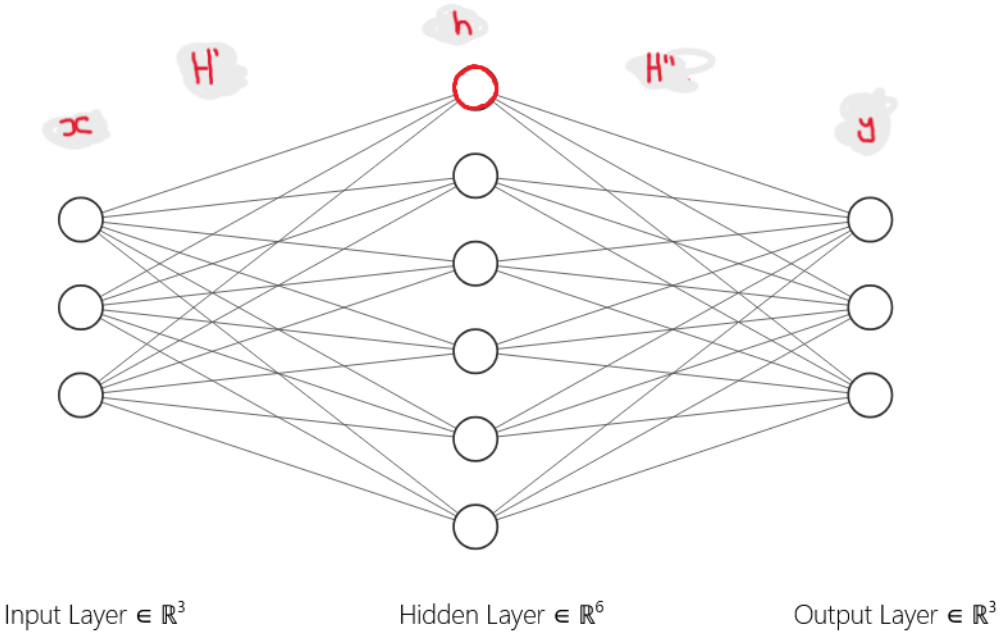

Figure 2: X is a 3 by N matrix. H is an 6 by N matrix. Y is a 3 by N matrix. N is the batch length. w_1 and w_2 are vectors of weights

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \times \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} & w_{1,5} & w_{1,6} \\ w_{21} & w_{2,2} & w_{2,3} & w_{2,4} & w_{2,5} & w_{2,6} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} & w_{3,5} & w_{3,6} \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 \end{bmatrix}$$

Figure 3: Matrix multiplication visualisation.

An efficient way to write these weighted summations for all of the neurons in a layer is by using matrix algebra.

$$h = x \times W^{'}$$
$$y = h \times W^{''}$$

The relationship between the equation 5 and the matrix multiplication representation of the network is highlighted in figure 1.

To train the network and adjust the weights, we perform inference on labelled data and calculate the error using a loss function $l()$, such as mean squared error (MSE):

$$L = l(y, y_{\text{label}}) \tag{6}$$

The goal is to minimize this loss by updating the network's weights in the direction of the negative gradient. This requires computing the gradient of the loss with respect to each weight w_i:

$$\frac{\partial L}{\partial w_i} \tag{7}$$

We first calculate the gradient of the layer closest to the output, $W^{''}$.

$$\frac{\partial L}{\partial W''} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial W''} = \frac{\partial L}{\partial y} \cdot h \tag{8}$$

We then use the chain rule again to calculate the gradient of L with respect to $W^{'}$.

$$\frac{\partial L}{\partial W'} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial h} \cdot \frac{\partial h}{\partial W'} = \frac{\partial L}{\partial y} \cdot W'' \cdot x \tag{9}$$

This step-by-step application of the chain rule allows the error to be propagated backward through the network, enabling efficient computation of gradients at each layer—this is the essence of **backpropagation**.

Once the gradients are known, the weights are updated using the gradient descent rule:

$$w_i \leftarrow w_i - \eta \frac{\partial L}{\partial w_i} \tag{10}$$

where η is the learning rate. Repeating this process across many training examples allows the network to gradually learn the desired input-output mapping.

This iterative optimization nudges the network toward improved accuracy, ensuring better alignment between predictions and target values over multiple training cycles. The problem with applying this algorithm to SNNs is that spike events are non-differentiable, therefore the gradient network cannot be calculated and the weights cannot be adjusted. So how are SNNs trained?

3.2 ANN-to-SNN Conversion

Due to the popularity of ANNs, literature on training them is quite advanced, so a natural and popular choice for training SNNs has been by converting a trained ANN model into an SNN model. Such ways of training have had good results for some tests, however such a method incurs large computational costs during conversion and is limited by the architecture of ANNs which are less adaptable to dynamic data [11]. Thus, to fully harness the benefits of SNNs — from energy efficiency to novel architectures — effective direct training methods are essential.

3.3 Direct Training

There are several approaches to directly training SNNS. ### Local learning rules

Local learning rules - Biologically feasible - Asynchronous unsupervised learning - More efficient learning using spiking behaviours

Memristor based STDP is an algorithm which is tailored to neuromorphic hardware - taking advantage of memristor behaviours. It uses local learning rules and is deployable on neuromorphic devices. [12]

3.3.1 BPTT + SG

- A very prominent method for directly training SNNs
- SG used to approximate the gradient
- Does well on static datasets
- Needs a ton of memory [13] The most prominent approach is backpropagation-through-time (BPTT), often paired with the surrogate gradient (SG) method. Since spikes are inherently non-differentiable, SG provides an approximation of their gradients, enabling the use of BPTT to optimize the model's parameters. This combination has become a prominent approach for training SNNs, allowing models to learn from temporal data and allowing further exploration of novel SNN architectures. However, BPTT is not considered a biologically plausible learning method, as neurons in the brain are believed to learn using local information [13] — that is, signals received directly from neighbouring neurons. As a result, BPTT may not fully leverage the efficiencies observed in biological systems. For example, implementing local learning rules could reduce data movement across the chip, potentially enhancing energy efficiency.

3.3.2 e-prop

An interesting approach. BPTT is biologically implausible as it requires storage of past neuron states and propagating errors backwards. Moreover, e-prop facilitates online real-time learning. Also e-prop requires far less energy than BPTT. Though it has it's advantages, models trained using e-prop aren't as accurate as BPTT. [11]

3.3.3 Parallelizable LIF

Neuron spike events are sequential

3.3.4 Eventprop

A novel training algorithm which utilises precise spike gradients to train the network. Eventprop computes gradients in a more efficient manner than BPTT. It reaches SOTA performance while using 4x less memory and running 3x faster. [14]

Information about the experiments: 1) Problem to solve (learning spoken digits.. further details - language etc) 2) Dataset used: how many datapoints for training, cross-validation, testing etc. 3) Model description: network size, layers, recurrent/feedforward, neuron model, synapse model etc. 4) Training process: method used, performance (plots for accuracy vs time for training, cross-val, testing)

SNNs have inherent temporal dependence and show promise for efficiently processing time-based data.

4 Problem To Solve

Spiking neural networks are inherently time dependent since information is encoded in the timing of spikes. This makes them naturally applicable for temporal tasks - tasks where the data evolves with time. Artificial neural networks can be tweaked to deal with temporal data of unspecified duration by introducing recurrence. Temporal data can come in many forms, for instance stock market information, video, or audio. This contrasts to static datasets where data is a single block and not related to other samples temporally, for instance image classification, pattern recognition, large language models. The efficiency and potential effectiveness of SNNs to process temporal information could be game changing in edge devices such as mobile phones, wearable devices, and remote sensors which have small power budgets. Large language models are revolutionising how we interact with computers, they provide a way humans to interface with machines using natural language. As a result, companies are eagerly integrating LLMs into their products and services - e.g. Siri, Raybans. Speech recognition is therefore a type of temporal data which is highly relevant and potentially game changing.

5 Spiking Heidelberg Digits Dataset

The Spiking Heidelberg Digits (SHD) dataset is a prominent spiking neural network benchmark. The wide use of SHD makes it good for fairly comparing different methods of training models. It is based on the Heidelberg Digits dataset, which is a collection of 10,000 high-quality recordings of spoken digits (0 to 9) in English and German. It is spoken by a relatively representative group of 6 males and 6 females of the ages 21 to 56 years old, with a mean age of 29. The HD dataset was converted into spike trains using a biologically realistic model of the cochlea, outputting 700 channels of spikes. It is a more challenging benchmark than MNIST which is nearing saturation. Moreover it is less computationally intensive than a visual dataset like DVS128 - which has recordings of gestures.

6 Model Description

The model uses Leaky Integrate-and-Fire neuron model with exponential synapses. It has an input layer of 700 neurons - corresponding to the 700 channels of the cochlea model used by SHD - and 20 output neurons for digits 0 to 9 in English and German. The model has a single hidden layer which has been tested with a size of 64, 128, 256, 512, and 1024. The hidden layer was tested using feed-forward only connections and fully connected recurrent connections, showing best results with a recurrent architecture.

7 Training Process

The input is provided into the model and it is allowed to develop under neuronal dynamics. The loss is calculated according to Equation (11). Using the adjoint method the loss is propagated backwards to find out how the timing of each event contributed to the loss so that the weights of the synapses can be adjusted accordingly.

$$\mathcal{L}_{sum-exp} = -\frac{1}{N_{batch}} \sum_{m=1}^{N_{bathc}} log(\frac{exp(\int_0^T e^{-t/T} V_{l(m)}^m(t)dt)}{\sum_{k=1}^{N_{out}} exp(\int_0^T e^{-t/T} V_k^m(t)dt)}) \tag{11}$$

This loss was chosen as it deals with the problem highlighted by the Gedanken Experiment.

References

[1] The state of AI in 2025: Global survey | McKinsey. [Online]. Available: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai

[2] B. Kindig. AI Power Consumption: Rapidly Becoming Mission-Critical. Forbes. [Online]. Available: https://www.forbes.com/sites/bethkindig/2024/06/20/ai-power-consumption-rapidly-becoming-mission-critical/

[3] W. Maass, "Networks of spiking neurons: The third generation of neural network models," vol. 10, no. 9, pp. 1659–1671. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608097000117

[4] W. Maass and H. Markram, "On the computational power of circuits of spiking neurons," vol. 69, no. 4, pp. 593–616. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022000004000406

[5] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," vol. 345, no. 6197, pp. 668–673. [Online]. Available: https://www.jstor.org/stable/24745271

[6] S. Wang, T. H. Cheng, and M.-H. Lim, "LTMD: Learning Improvement of Spiking Neural Networks with Learnable Thresholding Neurons and Moderate Dropout," vol. 35, pp. 28 350–28 362. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/b5fd95d6b16d3172e307103a97f19e1b-Abstract-Conference.html

[7] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks." IEEE Computer Society, pp. 2641–2651. [Online]. Available: https://www.computer.org/csdl/proceedings-article/iccv/2021/281200c641/1BmEpXS0r1m

[8] N. Rathi and K. Roy, "DIET-SNN: A Low-Latency Spiking Neural Network With Direct Input Encoding and Leakage and Threshold Optimization," vol. 34, no. 6, pp. 3174–3182. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9556508

[9] C. Jiang and Y. Zhang. KLIF: An optimized spiking neuron unit for tuning surrogate gradient slope and membrane potential. [Online]. Available: http://arxiv.org/abs/2302.09238

[10] S. Y. Arnaud Yarga and S. U. N. Wood, "Accelerating spiking neural networks with parallelizable leaky integrate-and-fire neurons*," vol. 5, no. 1, p. 014012. [Online]. Available: https://dx.doi.org/10.1088/2634-4386/adb7fe

[11] G. Bellec, F. Scherr, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets." [Online]. Available: https://www.proquest.com/publiccontent/docview/2172488501?pq-origsite=primo&sourcetype=Working%20Papers

[12] Spoken Digits Classification Based on Spiking Neural Networks with Memristor-Based STDP. [Online]. Available: https://ieeexplore.ieee.org/document/10216628/citations?tabFilter=papers#citations

[13] C. Zhou, H. Zhang, L. Yu, Y. Ye, Z. Zhou, L. Huang, Z. Ma, X. Fan, H. Zhou, and Y. Tian, "Direct training high-performance deep spiking neural networks: A review of theories and methods," vol. 18. [Online]. Available: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2024.1383844/full

[14] T. Nowotny, J. P. Turner, and J. C. Knight, "Loss shaping enhances exact gradient learning with Eventprop in spiking neural networks," vol. 5, no. 1, p. 014001. [Online]. Available: https://dx.doi.org/10.1088/2634-4386/ada852