Energy Efficient Spiking Neural Networks

by Saipraneet Darla

Student ID: F224272

Loughborough University

Final Year Project

Supervisor: Dr. Shirin Dora

Submitted may 2025

Abstract

This paper explores energy-efficient SNNs specifically through the lens of temporal coding through ISI-modulated synapses. With edge computing and IoT use cases on the rise, the demand for neural networks operating in very energy-limited regimes has become increasingly important. This work implements and compares a simplified form of ISI-modulated SNNs (IMSNNs) which adaptively modulate synaptic efficacy according to the timing of presynaptic spikes. Modelling on the MNIST dataset, the implementation achieves a 33.3% decrease in spike counts relative to standard SNNs and exemplifies the energy-saving potential of temporal coding. Although with the cost of a modest accuracy loss (77.02% compared to 81.40%), the paper highlights the intrinsic spike-energy relationship. The work also outlines important implementation hurdles regarding hardware limitations and computational cost that need to be overcome to fully achieve the energy savings of IMSNNs. By investigating both the algorithmic techniques and the practicalities of deployment, the paper contributes to the ongoing progress of developing neural networks that better reflect the energy economy of biological components but remain compatible with real-world applications.

Acknowledgement

I would specifically like to thank Dr. Shirin Dora for his valuable guidance, expertise and encouragement during the course of this final year project. His knowledge on spiking neural networks and energy efficient computation has played an extremely crucial role in setting the direction and approach of the project. I appreciate his patience and feedback during the times of implementation difficulty.

1 Introduction

1.1 Motivation

The rising demand for applications to be executed in real-time on low-energy devices has led to the need for energy efficient neural networks. Traditional deep learning models including Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs), effective though they are, are typically energy-intensive. Their massive computational demands make them less suitable for deployment on energy-restricted devices at the edge. This has necessitated the need for other models of the neural network that can operate within energy restrictions [9]. Spiking Neural Networks (SNNs), inspired by the energy-efficient operation of the actual neurons in the biological context, are one potential candidate. SNNs apply event-based, spike-based processing to reduce computational cost and are, as such, suitable for energy-restricted settings including wearables and IoT devices [13].

But despite the potential of SNNs, achieving optimal energy efficiency is challenging because network performance has to be balanced with lowering spike activity. Adaptive clocking and event-based computation schemes are among the techniques promising to decrease the energy consumption by adjusting the dynamical neuron activity according to the workload demands [9]. Other learning rules for synapses, for example, Spike-Timing-Dependent Plasticity (STDP), are able to let SNNs learn in an energy-efficient fashion by learning the weight of the synapse as a function of the spike timing of neurons [13, 12]. Thus, enhancing the energy efficiency of SNNs is critical to make them widely applicable to low-power situations with good computational efficacy [8].

1.2 Project Aims

The purpose of the present research is to develop energy-efficient SNNs through investigating the performance of the use of temporal coding elements, specifically ISI-modulated synapses. With increasing use of neural networks within edge computing and IoT devices, the overall objective is to design and analyze how energy expenditure is drastically cut down without sacrificing acceptable accuracy. This encompasses an examination of how dynamic synapse efficacy adjustment through spike timing can lead to improved processing within the neural network.

1.2.1 Goals

1. **Appraise temporal coding methodologies for energy efficiency:** This project critically assesses ISI-modulated SNNs and how well they can decrease

spike activity without sacrificing classification performance.

2. **Implement and verify an ISI-modulation mechanism:** The work shall design an efficient implementation of ISI-modulated synapses and analyze its performance in lowering energy expenditure as compared to traditional SNNs.

3. **Identify the hardware implementation difficulties:** Through the introduction of these methodologies, the work aims to outline the pragmatic limitations and considerations that need to be overcome when implementing energy-efficient SNNs on existing computing hardware.

4. **Make energy-accuracy trade-off assessments:** One of the main targets is to quantify the energy consumption (as measured during spike activity) and accuracy relationship, enabling application-specific optimization insights.

1.2.2 Objectives

1. **Comprehensive literature review:** Examine existing techniques for SNN energy efficiency, including spike cardinality techniques, compression techniques, and hardware optimizations.

2. **Design and develop an ISI-modulation framework at the neuron level:** Implement and design an optimized IMSNN to be implemented under existing hardware conditions.

3. **Comparative Performance Analysis:** Benchmark the IMSNN with traditional SNNs on standard datasets to compare both the accuracy of classification and the decrease in spikes.

4. **Analysis of computational overhead:** Evaluate the extra computational expenses related to ISI-modulation and how these impact training and deployment.

5. **Suggest future research opportunities:** Based on implementation experiences, outline promising areas for further energy-efficient SNN optimization and potential application areas in resource-limited environments.

2 Literature Review

There has been a lot of research done regards to reducing the energy consumption of SNN. The methods developed primarily, can be categorised into two - Compression and Spike cardinality methods [1]. The compression techniques focus on making

the network smaller or more efficient by pruning connections, reducing precision or automating the search for optimal architecture [1]. Whereas Spike Cardinality methods aim to minimise the numeber of spikes generated during computation, thereby directly reducing the energy required for processing [1]. This review explores several ways under each method.

2.1 Compression Techniques

Kundu and Souvik introduced an approach called the Spike-Thrift, which addresses the inefficiencies in SNN by selectively pruning neurons and synapses based on their importance. The authors recognised that not all neurons and connections contribute equally to the network's output and that unnecessary spiking activity can significantly increase energy consumption[8]. In order to tackle this they integrated an attention mechanism that evaluates the significance of each neuron and synapse during the training process.

In this method each connection is assigned an importance score derived from the attention mechanism. Connections with lower scores are gradually pruned, making the network sparser without compromising it's overall performance[8]. In their method each connection is assigned an importance score derived from the attention mechanism. Connections with lower scores are gradually pruned, making the network sparser without compromising it's overall performance [8]. The pruning process is guiding by the following equation:

$$w_i' = w_i \cdot A_i \tag{1}$$

where w_i is the original synaptic weight, and A_i is the attention based factor[8]. The iterative nature of this after pruning makes it particularly effective. The network is fine tuned to recover any lost performance, ensuring the accuracy remains intact [8]. They tested this method on the CIFAR-10, showing that their approach led upto 12.2 times greater energy efficiency while maintaining the same level of accuracy[8] as the original network.

Instead of blindly cutting connections that could result in the lose of accuracy, they carefully consider network structure, making deliberate decisions about which elements to prune. Designing efficient SNN architectures manually can be a daunting task, as it involves balancing multiple parameters and understanding complex spiking dynamics. To address this challenge, Byunggook Na, proposed AutoSNN, an automated framework for discovering optimal SNN architectures through Neural Architecture search (NAS) [10]. The primary objective of this method was eliminate the manual way of identifying the most efficient and making use of computational algorithms to do so.

This method works by exploring a search space of possible network architectures and evaluating each of the architecture based on a fitness function[10] that is mentioned below.

$$Fitness = \alpha \cdot Accuracy - \beta \cdot SpikeCount \tag{2}$$

Here, α and β are weights that balance the trade-off between accuracy and energy consumption. During the search process this fitness function helps refine architectures by selecting only those that achieve high performance with lower spikes[10].

This was tested data sets like CIFAR-10, where the results show that this proposed method not only outperforms the manual method in finding the most efficient configuration but also generates fewer spikes[10]. This can be seen in figure 1 which was plotted by the author.

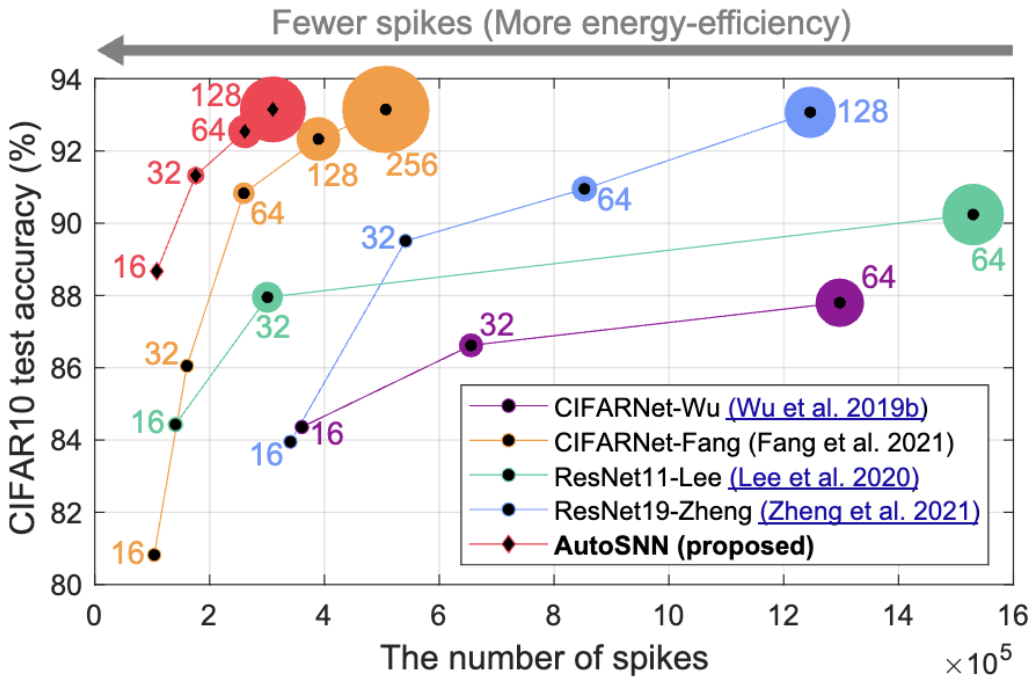

Figure 1: Results obtained when the proposed method was tested on CIFAR-10.

Black circle- and diamond-shaped markers denote hand-crafted and automatically-designed SNN architectures, respectively. The size of a coloured circle is proportionate to the model size and the number next to the circle indicates the initial channel of each SNN architecture[10].

The strength of AutoSNN lies in it's ability to adapt to the specific requirement of different tasks. Through the automated search process Byunggook Na removed

the need for human intuition, allowing the AutoSNN to uncover configurations that might have been overlooked otherwise.

Another remarkable work done in order to reduce both memory usage and energy consumption was by introducing FSpiNN. Putra and Shafique identified that one of the primary bottlenecks in SNN efficiency is the high cost associated with synaptic operations and weight storage[11]. In order to address this they tried multiple strategies like computational reduction and weight quantisation[11]. Computational reduction minimises the number of neural operations synaptic weight updates, decreasing the overall process load and weight quantisation compresses the weights by converting them to fixed-point representations, which significantly reduces memory requirements[11]. The quantisation process is defines by:

$$w_i' = \frac{round(w_i \cdot Q)}{Q} \tag{3}$$

where w_i' is the original weight, and Q is the quantisation factor. By rounding weights to the nearest fixed-point value, FSpiNN achieves a substantial reduction in the memory footprint without sacrificing performance [11]. Putra and Shafique tested FSpiNN on datasets like MNIST and Fashion MNIST.Their results showed a 7.5x reduction in memory usage and a 3.5x improvement in energy efficiency during training[11].

BitSNN's is another framework that solely focus on improving the energy efficiency. This idea emerged idea of integrating BNN's(Binary Neural Networks) with SNNs[6]. Traditional SNNs often require complex computations that involve floating point weights, which are energy-intensive, therefore to simplify this the researchers constrained the weights to binary values, typically +1 or -1 [6]. This method reduces the complexity of synaptic operations making this kind of SNN more suitable for hardware implementation.

As the synaptic weights are restricted to binary values, they are binarised using the following sign function:

$$w_i' = \text{sign}(w_i) = \begin{cases} +1 & \text{if } w_i \geq 0, \\ -1 & \text{if } w_i < 0. \end{cases} \tag{4}$$

One of the challenges that the researchers identified with this was maintaining a gradient-based learning, which seemed difficult due to the non-differentiable nature of the sign function[6]. To address this the researchers made use of the Straight-Through Estimator(STE) in order to approximate the gradient[6]. The formulae it as follows:

$$\frac{\partial L}{\partial w_i} \approx \frac{\partial L}{\partial w_i'} \cdot 1_{|w_i| \leq 1} \tag{5}$$

where ∂L is the loss function and $1_{|w_i| \leq 1}$ is an indicator function that constrains the gradient to the range [-1,1]. The major advantage of this is that it also enables

activation sparsity to minimise the number of spikes[6]. This is achieved through the following sparsity regularizer equation that penalises excessive spiking during the training process.

$$\mathcal{L}_{\text{spike}} = \lambda \sum_{j} S_j \tag{6}$$

where S_j represents number of spikes produced by neuron j and λ is the regularization coefficient[6].

Researchers experimented this using the CIFAR-10 datasets that shows BitSNN's achieve better accuracy than the traditional SNNs while significantly reducing computational complexity and energy consumption[6].

As stated earlier by the authors of [1], the second way of reducing energy consumption of SNNs is through Spike Cardinality methods.

2.2 Spike Cardinality Methods

Dengyu Wu, Xinping Yi and Xiaowei Huang approach the challenge of reducing consumption of SNN while maintaining accuracy, by converting a pre-trained CNN (Convolutional Neural Network) into a SNN [15]. The major motivation behind this method was to take advantage of the training capabilities of CNN's and the energy efficiency of SNN. The authors recognised that training SNN's directly can be difficult due to the non-differential nature of spike events, therefore they first trained a CNN and then converted it into a SNN.

The conversion process involves mapping the continuous activations of CNN neurons to the firing rates or spike timings of SNN neurons [15]. In order to ensure that this mapping preserves the network's functionality, they adjusted the membrane potential V of the spiking neuron by the following formulae:

$$V(t) = V(t-1) + \sum w_i S_i(t) - \theta S_o(t) \tag{7}$$

where w_i is the synaptic weight, $S_i(t)$ is the incoming spike, and θ is the firing threshold. Through series of experiments on data sets like MNIST and Fashion MNIST, they proved that converted CNN's achieve higher accuracy while maintaining a lower rate of energy consumption [15]. This opened new ways for researchers as SNN could be combined with multiple other neural network frameworks to take maximum advantage of both the frameworks.

Apart from combing frameworks, restrict the spiking activity is another way to minimise energy consumption. The same idea was used to create the Single-Spike Hybrid Input Encoding, a technique that allows the neurons to fire at most only once per input [5]. This method combines analog input encoding with spike timing to achieve highly efficient neural computation while preserving accuracy.

In this method, the input is encoded using a hybrid approach that combines analogue and spike based representations. Each neuron integrates incoming signals and is allowed to fire a single spike based on the timing of its membrane potential crossing the threshold [5]. The membrane potential dynamics of a neuron are modelled as follows:

$$V_i(t) = V_i(t-1) + \sum_j w_{ij} x_j(t) - \theta \tag{8}$$

where $V_i(t)$ is the membrane potential of the neuron i at time t, w_{ij} represents synaptic weight between presynaptic neuron j and postsynaptic neuron i, $x_j(t)$ is the input signal at time t and θ is the firing threshold. A neuron fires a spike S_i when it's membrane potential crosses the threshold:

$$S_i = \begin{cases} 1, & \text{if } V_i(t) \geq \theta \text{ and no spike has been fired before,} \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The author deals with the non-differentiability of the spike function by making use of the surrogate gradients. Datta et al. define surrogate gradient $\sigma(V_i)$ as follows:

$$\sigma(V_i) = \frac{1}{(1 + e^{-k(V_i - \theta)})^2} \tag{10}$$

where k is a steepness parameter that controls the gradient approximation. They train this network using backpropogation through time(BPPT) [5] with surrogate gradients to optimise weights while allowing the single-spike constraint. They tested this network using datasets like MNIST and CIFAR-10. Datta et al state that from the results it was clear that this method achieved upto 125x energy reduction [5] compared to a traditional ANN however, the accuracy compared to the previous methods discussed is arguable lower.

Another architecture that was proposed in order to optimise the energy consumption is the synaptic modulation based on Interspike Intervals(ISI) proposed by Dylan Adams et al. Traditional SNNs rely on fixed synaptic weights, meaning that every spike contributes equally to the postsynaptic potential. However, in biological systems, the timing between spikes—known as the Interspike Interval (ISI)—affects synaptic strength [1]. Inspired by this, the authors developed ISI Modulated SNNs (IMSNN), where synaptic weights are dynamically adjusted based on the timing of presynaptic spikes.

In IMSNNs, the synaptic weight is no longer a fixed value but a function of the ISI of the presynaptic neuron. Specifically, the weight of a synapse is determined by a Gaussian function of the ISI:

$$\vartheta_{ij}(t) = w_{ij} \exp\left(-\frac{(\phi_i(t) - \mu_{ij})^2}{2\sigma_{ij}^2} \right) \tag{11}$$

where $\vartheta_{ij}(t)$ represents synaptic weight at time t,w_{ij} is the height of the Gaussian (learnable parameter), $\phi_i(t)$ is ISI of the presynaptic neuron, μ_{ij} is the mean ISI for maximum synaptic contribution and σ_{ij} stands for the standard deviation controlling the spread of the Gaussian [1].

This dynamic adjustment allows synapses to favour specific ISIs, meaning spikes that arrive with optimal timing contribute more to the postsynaptic potential, while spikes with suboptimal timing contribute less. The learning algorithm selectively updates synaptic parameters to maximize ISIs, thereby reducing the frequency of spikes. The authors test this suggested method using a gradient descent-based learning algorithm which is adapted for the IMSNNs. The algorithm includes a mechanism to prevent updates that would reduce ISIs, which would otherwise increase spike frequency. Instead, the algorithm promotes updates that increase ISIs, effectively reducing the number of spikes. During the forward propagation the membrane potential $v_j(t)$ is updated as shown below:

$$v_j(t+1) = \beta v_j(t) + \sum_i s_i(t)\vartheta_{ij}(t) \tag{12}$$

where β is the decay constant, and $s_i(t)$ represents the presynaptic spike train [1]. This model was evaluated using the datasets like MINIST and FashionMNIST.

The results showed that IMSNNs achieved similar classification accuracy compared to conventional SNNs while generating up to 90% fewer spikes [1]. In their paper they give specific statics showing how on the MNIST dataset, IMSNNs reduced spikes per neuron from 1.19 to 0.17 in a network with one hidden layer [1] and on the FashionMNIST dataset, IMSNNs achieved a spike reduction of 55% to 92%, depending on the architecture[1]. By incorporating ISI-dependent modulation, the authors align SNN behavior more closely with the time-based dynamics of biological neurons.This approach not only enhances energy efficiency but also introduces a new dimension to how time-based patterns can be exploited in SNNs.

2.3 Additional Energy Efficient methods

Building on the compressions and spike cardinality methods discussed so far, the following techniques focus on adjusting learning rules, thresholds, or spiking activity in a dynamic way. The approaches below highlight how small changes to the algorithm can substantially lower the power while preserving or even enhancing network performance.

Alawad, Yoon and Tourassi introduced an approach known as the stochastic approach that reduces the energy usage in SNNs when trained on sparse data, particularly large-scale text corpora [2]. Their primary motivation roots from the fact that deterministic neuron dynamics can lead to unnecessary spikes when dealing with data

containing many zero-valued features. Integrating stochastic elements into the neuron model effectively filters out uninformative inputs leading to fewer overall spikes. In their work, the authors propose injecting controlled randomness into each neuron's membrane update equation. A simplified version of this update equation is shown below:

$$V_i(t+1) = V_i(t) + \eta \cdot \Omega\big(\mathbf{x}(t), \mathbf{w}_i\big) \; - \; \xi \cdot \Gamma_i(t) \tag{13}$$

where $V_i(t)$ is the membrane potential of neuron i at time t, $x(t)$ is the sparse input vector, w_i represents the neuron's weights, and $\Omega(\cdot)$ encapsulates a random sampling function that determines whether a spike is triggered based on the current input and weights. The term $\Gamma_i(t)$ denotes a stochastic noise factor which can modulate the membrane potential, and η and ξ are constants balancing the influence of input-driven updates versus random fluctuations. According to Alawad et al., such randomness significantly reduces firing rates in neurons when the input remains largely inactive.

A key advantage of this approach is its suitability for large-scale textual data, wherein the majority of features(e.g., vocabulary tokens) may not be activated for most samples. The authors demonstrate that, by allowing only the most salient patterns to trigger spikes, energy consumption is reduces while maintaining the classification accuracy. Moreover, they show that the method is relatively straightforward to implement, making it flexible to integrate with existing SNN training pipelines for natural language processing tasks [2].

In their experiment setup, the authors treated the entire feature set as a probability distribution rather than encoding each feature individually. This approach effectively reduced network connectivity from O(n) to O(log N), making the framework highly energy efficient [2]. They conducted thorough evaluations on multiple datasets: MNIST, IMDb, and an in-house clinical dataset. Notably, on MNIST, the stochastic-based SNN achieved nearly the same classification accuracy as a fully connected DNN, while outperforming a conventional SNN in terms of energy consumption as seen in the table below. Specifically the fully connected DNN, conventional SNN, and data-normalised SNN consumed 38.24, 1.83 and 1.85 times more energy, respectively, compared to the new stochastic-based SNN [2].

NN architecture	Accurcy	Energy Consumption (μJ)
DNN	98.68%	657.75
SNN	98.48%	31.5
Data-Norm SNN [17]	98.64%	31.8
Stochastic SNN	98.65%	17.2

Figure 2: Comparison of performance accuracy and energy consumption for different NN architectures.

A key highlight of their experiments was the method's robust performance on sparse text datasets like IMDb and the in-house clinical corpus. Whereas a traditional SNN's classification accuracy dropped substantially when dealing with these sparse inputs, the stochastic-based SNN demonstrated classification results comparable to those of a conventional DNN [2]. This indicates that introducing a random spike train, guided by the overall feature distribution, not only cuts down on synaptic overhead but also preserves the model accuracy. Moreover, the integrative use of integrate-and-Fire neurons further contributed to the approach's energy-saving benefits, underscoring the viability of distribution-based spike generation for real-world, resource-limited applications [2].

Another framework called Online Spatio-Temporal Learning(OSTL) was proposed by Bohnstingl et al. which is a biologically inspired framework for online training of deep recurrent neural networks(RNNs) and spiking neural networks(SNNs) [4]. Traditional approaches like backpropogation Through Time(BPTT) rely on offline gradient computation by unrolling temporal sequences, which incurs significant memory and computational costs. OSTL circumvents this limitation by decoupling spatial and temporal learning components, enabling real-time gradient updates without storing past network states. This design aligns with biological systems, where neurons adapt dynamically through local and temporal interactions [4].

This framework decomposes gradients into spatial (∇_S) and temporal (∇_T) terms. Spatial gradients are computer using current inputs and hidden states, while temporal gradients capture dependencies across time through a feedback loop. The combined updated rule is defined as:

$$\nabla_{\text{total}} = \nabla_S + \gamma \nabla_T \tag{14}$$

Here, γ is a decay factor that regulates the influence of temporal dependencies over

time [4]. For shallow SNNs, OSTL is mathematically equivalent to BPTT, ensuring similar learning performance while eliminating the need for offline computation. Most importantly, this framework reduces the time complexity from $O(T)$ to $O(1)$ per time step(where T is the sequence length), making it highly scalable for real-time applications.

The authors validated OSTL on language modelling (Penn Treebank) and speech recognition(TIMIT) tasks. On Penn Treebank, it achieved a perplexity score of 112.5, nearly matching BPTT's 110.8, while reducing the training time by 30%. For TIMIT, OSTL attained a phone error rate(PER) of 18.7%, comparable to BPTT's 18.4%, with a 45% reduction in memory consumption during training. These results obtained by the authors highlight the framework's ability to preserve accuracy while drastically lowering computational overhead. Furthermore, the authors tried to extend this framework to architectures like LSTMs and GRUs, demonstrating its versatility.

While OSTL addresses energy efficiencies in the training phase through gradient decoupling, recent advances have also explored optimising network parameters during learning to further enhance energy efficiency. Sun et al. proposed a Synapse-Threshold Synergistic Learning (STSL) framework, which combines synaptic and threshold plasticity to reduce spiking activity while maintaining accuracy [14].

The authors state that this framework makes use of a **Joint Decision Framework (JDF)** that dynamically adjusts synaptic weights(w_{ij}) and neuron threshold (θ_j) during training. The membrane potential $V_j(t)$ of neuron j is governed by:

$$V_j(t) = \beta\, V_j(t-1) \;+\; \sum_i w_{ij}\, S_i(t) \;-\; \theta_j(t). \tag{15}$$

where β is the membrane potential decay factor, $S_i(t)$ denotes presynaptic spikes, and $\theta_j(t)$ is the time-varying threshold. The authors also highlight that unlike conventional SNNs with fixed thresholds, $\theta_j(t)$ is updated synergistically with synaptic weights using a gradient based rule derived from a loss function $\mathcal{L}$:

$$\Delta w_{ij} = -\eta\, \frac{\partial \mathcal{L}}{\partial w_{ij}}, \quad \Delta \theta_j = -\eta\, \frac{\partial \mathcal{L}}{\partial \theta_j}. \tag{16}$$

Here η is the learning rate. The authors makes use of surrogate gradients in order to compute the partial derivates to approximate the non-differential spike functions. Specifically, the threshold gradient is regularised to prevent excessive spiking by the following formulae:

$$\frac{\partial \mathcal{L}}{\partial \theta_j} = \sum_t \frac{\partial \mathcal{L}}{\partial S_j(t)} \cdot \frac{\partial S_j(t)}{\partial \theta_j} \;\approx\; \sum_t \frac{\partial \mathcal{L}}{\partial S_j(t)} \cdot \sigma'\big(V_j(t) - \theta_j(t)\big). \tag{17}$$

where σ' is the derivative of a surrogate function(e.g., sigmoid). This dual optimisation ensures synapses and thresholds co-adapt to minimise the redundant spikes while preserving the performance [14].

The authors have evaluated STSL on datasets like **MNIST, CIFAR-10,** and **DVS-Gesture**. On **CIFAR-10**, STSL achieved 92.3% accuracy, outperforming baseline SNNs(89.1%) and hybrid ANN-SNN conversion methods(91.5%). Notably STSL reduced the average spikes per neuron by 38% compared to fixed threshold SNNs, achieving 2.1x lower energy consumption during inference. Moreover, for neurotrophic datasets like DVS-Gesture, STSL achieved 96.8% accuracy vs 94.2% in standard SNN with 45% fewer spikes, showing it's robustness to temporal sparsity. Sun et al. also specifically point out that during training this framework required 30% fewer epochs than isolated synaptic or threshold optimisation strategies.

So far, methods mentioned optimise SNNs by algorithm sparsity and synaptic plasticity, Bhattacharjee et al. critically evaluated the energy efficiency of SNNs from a hardware perspective, challenging conventional algorithmic claims by taking into consideration the bottlenecks faced in real-world deployment. The authors introduced **SATA**(Sparsity-Aware Training Accelerator) and **SpikeSim**, two hardware benchmarking platforms for large-scale SNN inference. SATA operates on digital systolic arrays optimised for sparsity-aware computations, while SpikeSim evaluates SNNs on analogue crossbars using resistive RAM (RRAM) technology. Their analysis revealed stark discrepancies between theoretical energy estimates—based on metrics like FLOPs and spike sparsity—and actual hardware performance.

A major bottleneck identified is repeated computations and data movements across timesteps. SNNs process inputs over multiple timesteps, leading to redundant data transfers between memory and processing units. According to the Fig 3, for a VGG9 SNN on SATA, increasing timesteps from 1 to 4 resulted in a 62.5% rise in memory movement energy [3]. To mitigate this the authors proposed an SNN-tailored dataflow that reuses weights across timesteps in systolic arrays, reducing data movements energy by 62.5% for a 4-timestep model. Additionally, Dynamic Timestep SNN (DT-SNN) dynamically terminates inference early for easy inputs using a entropy-based module, reducing the average time-steps from 4 to 2.14 for a VGG16 SNN on TinyImageNet and slashing Energy-Delay Product (EDP) by 2.54x as seen from figure 4.

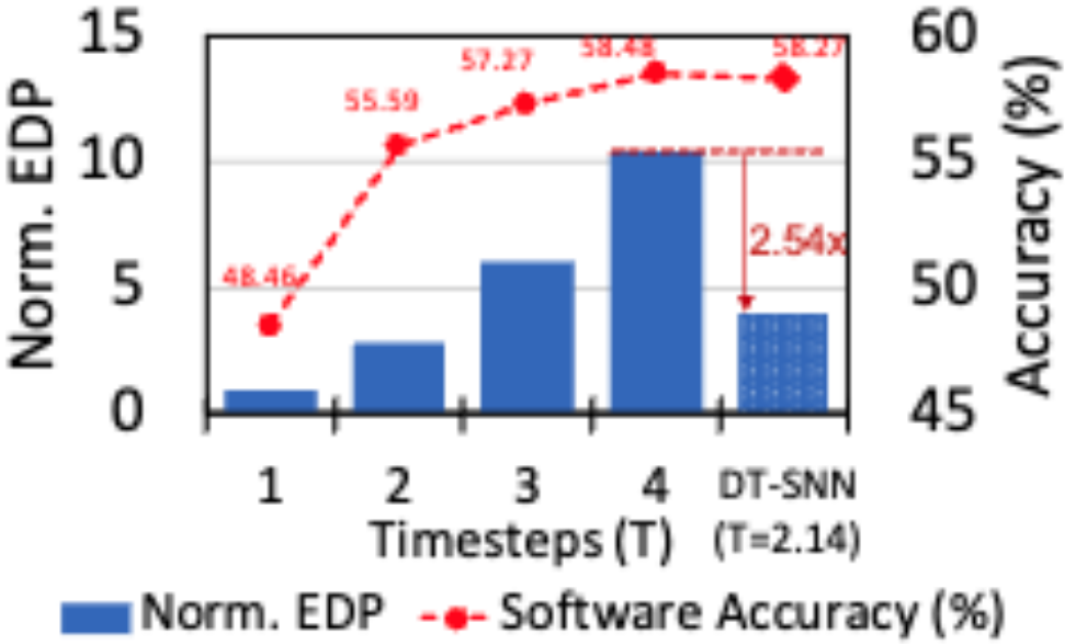

Figure 3: Impact of DT-SNN on SpikeSim using VGG16 SNN on the TinyImageNet dataset.

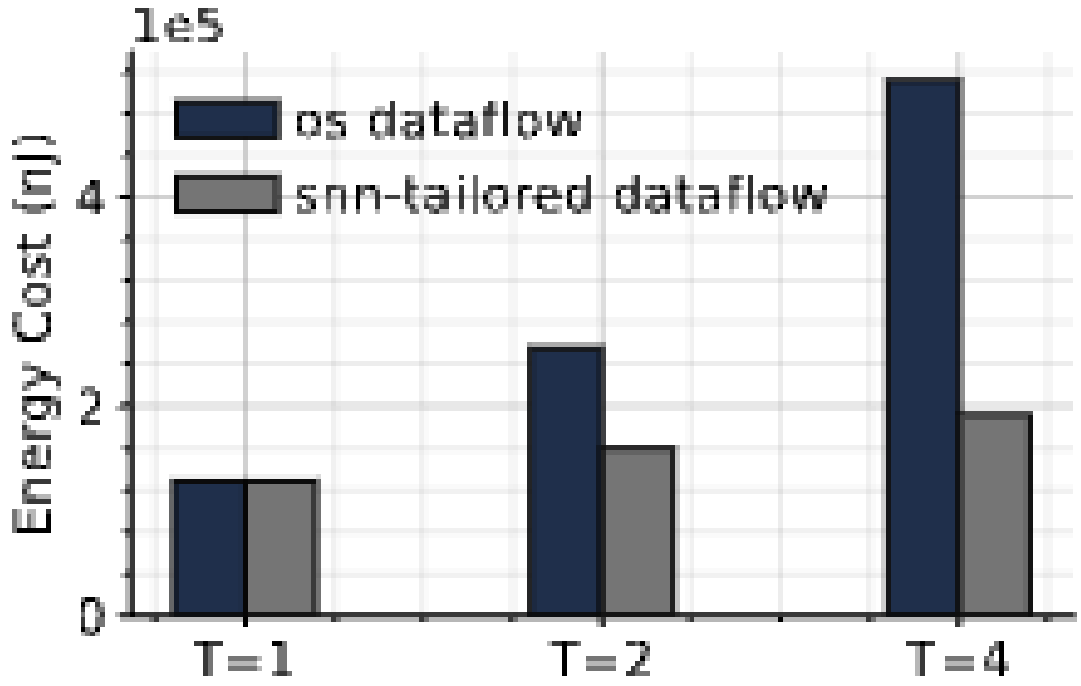

Figure 4: Data movement energy cost difference between a standard OS dataflow and the SNN-tailored dataflow for SATA using VGG9 SNN on Tiny ImageNet dataset.

Another crucial bottleneck is the LIF(Leaky-integrate-and-fire) neuronal overhead. LIF modules consume up to 61.6% of total power in digital accelerators due to membrane potential storage and reset operations [3]. The authors introduced LIF-sharing, where a single LIF neuron is shared across multiple channels, reducing LIF power by 75.1% on SATA (as shown in Fig.5b in [3]). The membrane potential quantisation further reduced SRAM(Static RAM) storage overhead, complementing LIF-sharing.

For analog crossbars, non-idealities like IR-drop and device variations degraded SNN accuracy by up to 40% as seen from figure 5 below, in order to deal with this the authors made use of non-ideality-aware weight encoding, which prioritises high-resistance synapses to reduce current leakage, and batchnorm adaption, which adjusts batchnorm parameters to noisy crossbar outputs. These strategies improved non-ideal accuracy by 40.13%, narrowing the gap between software and hardware performance.

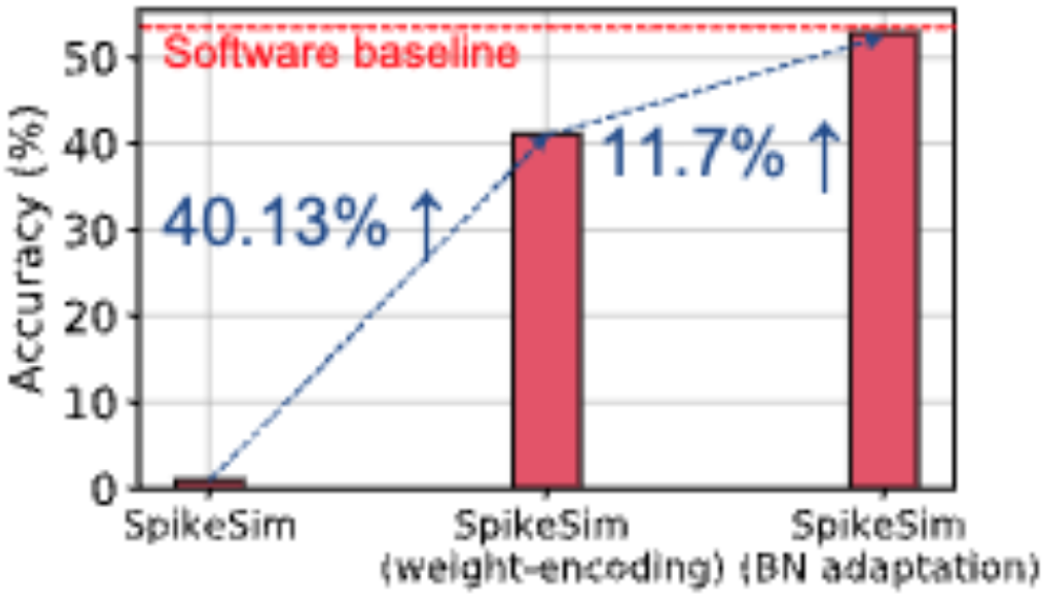

Figure 5: Non-ideal accuracy improvement on SpikeSim for a pre-trained VGG16 SNN (4-bit weights) on Tiny ImageNet dataset with non-ideality-aware weight-encoding & BN adaptation.

The exploration of energy-efficient Spiking Neural Networks (SNNs) reveals a dynamic interplay between biologically inspired algorithms and hardware-aware innovations. Methods such as Spike-Thrift and AutoSNN demonstrate the efficacy of architectural compression through attention-guided pruning and neural architecture search, achieving up to 12.2× energy savings on vision tasks. Complementing these, Spike Cardinality techniques like Single-Spike Hybrid Encoding and ISI-modulated SNNs optimize temporal coding, reducing spike activity by 55–90% while preserving accuracy. Recent advances in training paradigms, such as OSTL and STSL, further bridge efficiency gaps by decoupling spatio-temporal gradients and co-adapting synaptic-threshold parameters, slashing training overhead by 30–45%. However, the work of Bhattacharjee et al. highlights a critical reality: algorithmic claims of energy efficiency must be validated against hardware bottlenecks like repeated data movements, LIF neuronal overhead, and analog crossbar non-idealities. Their SATA and SpikeSim platforms reveal that co-design strategies—such as dynamic timestep reduction, LIF-sharing, and non-ideality-aware adaptation—are indispensable for translating theoretical savings into practical gains. Together, these methods showcase a holistic approach to SNN optimization, balancing biological fidelity with computational pragmatism. As the field evolves, the integration of hardware-aware training, adaptive spike coding, and cross-layer compression will be pivotal in unlocking SNNs'

full potential for low-power, real-time applications at the edge.

3 Methodology

Building on the insights from existing spike cardinality methods and synaptic modulation techniques discussed earlier, we are now going introduce a unique implementation of ISI-modulated SNNs(IMSNNs) that dynamically adjusts synaptic efficacy based on presynaptic spike timing. This method combines gradient based optimization with biologically inspired temporal coding to minimize spike activity while attempting to preserve classification accuracy. In the following sections, it is explained in detail the mathematical foundations, network architecture, and the training process of the proposed IMSNN method.

3.1 Network Architecture and Synaptic Modulation

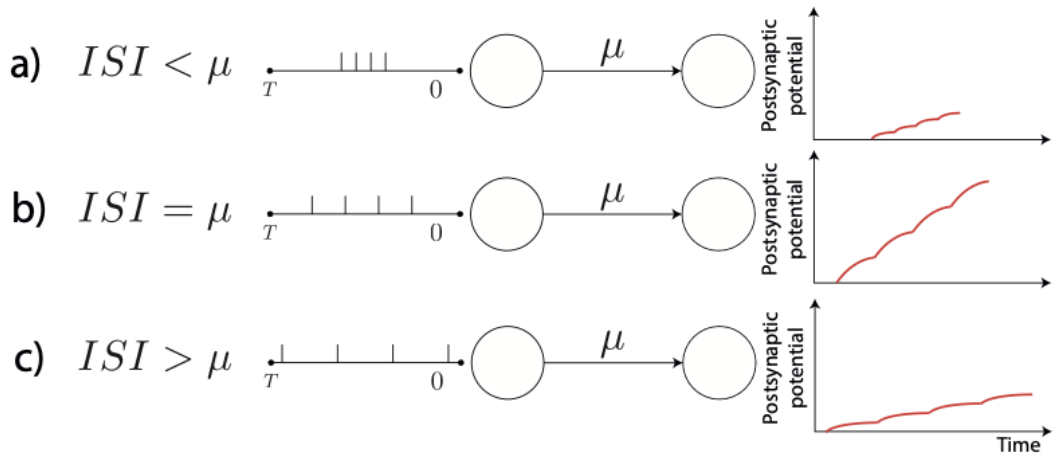

Figure 6: Schematic of the ISI-modulated SNN architecture. (A) Network layers with dynamic synapses. (B) Synaptic efficacy modulation based on presynaptic interspike intervals (ISIs). Spikes arriving with optimal ISI (μ) contribute maximally, while deviations reduce efficacy. (C) Postsynaptic potential (PSP) dynamics for different presynaptic ISIs.

The IMSNN consists of L layers of leaky integrate-and-fire (LIF) neurons interconnected through ISI-modulated synapses as shown in Figure 6. It illustrates the hierarchical structure and synaptic modulation principles. The network consists of input,

hidden and output layers, where synapses dynamically adjust their efficacy based on temporal pattern of presynaptic spikes(Figure 1a). For a presynaptic neuron i, the synaptic weight $v_{ij}(t)$ is governed by a Gaussian function, as stated below, of its ISI (Figure 1b).

$$\vartheta_{ij}^{(\ell)}(t) = w_{ij}^{(\ell)} \exp\left(- \frac{\left(\phi_i^{(\ell)}(t) - \mu_{ij}^{(\ell)}\right)^2}{2\left(\sigma_{ij}^{(\ell)}\right)^2} \right) \tag{18}$$

where:

- $\vartheta_{ij}^{(\ell)}(t)$: Synaptic weight between neuron i (layer $L-1$) and neuron j (layer ℓ) at time t.
- $w_{ij}^{(\ell)}$: Learnable height parameter.
- $\phi_i^{(\ell)}(t)$: Inter-Spike Interval (ISI) of the presynaptic neuron i.
- $\mu_{ij}^{(\ell)}$, $\sigma_{ij}^{(\ell)}$: Mean and width of the Gaussian, initialized randomly and fixed during training.

The ISI $\phi_i^{(\ell)}(t)$ is updated iteratively through the following equation:

$$\phi_i^{(\ell)}(t+1) = 1 + \phi_i^{(\ell)}(t) \cdot \left(1 - s_i^{(\ell)}(t)\right) \tag{19}$$

where $s_i^{(\ell)}(t)$ is the spike output of neuron i. Spikes arriving with an ISI close to μ_{ij} (e.g., 10ms) maximize the synaptic contribution to the postsynaptic neuron j, while shorter or longer ISIs result in exponentially reduced efficacy(Figure 1c). This mechanism ensures that only temporally precise spike train post synaptic activity, essentially suppressing redundant spikes.

3.2 Forward Prorogation and Neuron Dynamics

Neurons in the hidden layer follow the LIF dynamics. So for a neuron j in layer l, the membrane potential $v_j^{(\ell)}(t)$ evolves as:

$$v_j^{(\ell)}(t+1) = \underbrace{\beta\, v_j^{(\ell)}(t)}_{\text{Leakage}} + \underbrace{\sum_i s_i^{(\ell-1)}(t)\, \vartheta_{ij}^{(\ell-1)}(t)}_{\text{Synaptic Input}} - \underbrace{\theta\, s_j^{(\ell)}(t)}_{\text{Reset}} \tag{20}$$

where:

- $\beta = 0.99$: Decay factor, ensuring gradual "forgetting" of past inputs.
- $\theta = 1.0$: Firing threshold.

- $s_j^{(\ell)}(t)$: Binary spike (1 if $v_j^{(\ell)}(t) \geq \theta$, else 0).

Unlike hidden layers, output neurons do not spike. Instead, their membrane potential accumulate over $T = 100$ timesteps. Class probabilities are computed using a softmax over the accumulated potentials:

$$p_j = \frac{\sum_{t=1}^{T} v_j^{(L)}(t)}{\sum_{k=1}^{n^{(L)}} \sum_{t=1}^{T} v_k^{(L)}(t)} \tag{21}$$

This avoids spike-driven noise in the final classification step, a strategy validated in hybrid ANN-SNN models [15].

3.3 Learning Algorithm: Selective Gradient Propagation

This IMSNN is trained using backpropagation through time(BPTT) with surrogate gradients [5], adapted to suppress updates that increase spike frequency. During backpropagation, the non-differentiable spike function $s_j(t)$ is approximated using a sigmoid-shaped surrogate gradient given by the following equation:

$$\frac{\partial s_j(t)}{\partial v_j(t)} \approx \frac{1}{\left(1 + e^{-\alpha \left(v_j(t) - \theta\right)}\right)^2} \tag{22}$$

where $\alpha = 10$ controls the gradient steepness. This allows gradient-based optimization while being able to be compatible with SNN dynamics [10]. In order to minimize spikes, gradients that would shorten the ISI (increasing spike frequency) are selectively suppressed. For each synapse, the gradient of the loss $\mathcal{L}v$ with respective to w_{ij} is masked as follows:

$$\nabla w_{ij} = \begin{cases} \nabla w_{ij}^{\text{raw}}, & \text{if } \nabla \phi_j(t) < 0 \ \big(\text{lengthens ISI}\big), \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

This ensures that the elongated ISIs are preferred during synaptic updates, directly reducing energy consumption.

3.4 Training Protocol and Implementation Details

3.4.1 Input encoding

For the training process and testing, MNIST dataset has been used, which is a dataset containing hand-written digits (0-9) with 60,000 training images and 10,000 testing

images, all grayscale and 28x28 pixels. As this dataset is not a time-varying dataset, it encoded into spike trains using Poisson rate encoding. Pixel intensity $x \in [0, 1]$ is mapped to a firing rate $\lambda = 28.5 + 71.5\,x$ Hz, generating Bernoulli-distributed spikes over $T = 100ms$(1ms timestep). For example, a pixel with $x = 0.5$ fires at $\lambda = 64.25Hz$, producing ~ 6 spikes in 100 ms.

3.4.2 Initialization

Synaptic weights w_ij are initialized with small random values $\sim \mathcal{N}(0, 0.05)$ to avoid saturation and as seen earlier in synaptic modulation the μ_{ij} and σ_{ij} are fixed post-initialization to stabilize training.

3.4.3 Regularization

A spike count penality is added to the loss function to further discourage excessive spiking:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda \sum_{\ell=1}^{L-1} \sum_{t=1}^{T} \sum_{j=1}^{N^{(\ell)}} s_j^{(\ell)}(t) \tag{24}$$

where $\lambda = 0.01$ balances accuracy and energy efficiency.

3.4.4 Optimization

The Adam optimizer [7]($(\eta = 10^{-4})$) is used for training. If the validation accuracy is unchanged for 5 epochs then early stopping is applied.

3.5 Energy Efficiency Metrics

Energy consumption is approximated using the average spike count per neuron ($\kappa_n^{(\ell)}$):

$$\kappa_n^{(\ell)} = \frac{1}{N^{(\ell)}} \sum_{t=1}^{T} \sum_{j=1}^{N^{(\ell)}} s_j^{(\ell)}(t), \tag{25}$$

where $N^{(\ell)}$ is the number of neurons in layer ℓ. Assuming each spike consumes a fixed energy E_{spike}, the total network energy is

$$E_{\text{total}} = E_{\text{spike}} \cdot \sum_{\ell=1}^{L} N^{(\ell)}\, \kappa_n^{(\ell)}. \tag{26}$$

This metric aligns with hardware studies showing that spike count correlates linearly with energy use in neuromorphic chips [3].

4 Results and Evaluation

In the section below, the performance of using ISI-modulated SNN (IMSNN) is shown and compared to the performance of the standard SNN on the MNIST data set. The model is compared according to classification accuracy as well as spike activity, which is directly related to energy expenditure.

4.1 Experimental Setup

All experiments were run using Pytorch CPU implementation. Because of hardware limitations, the simplified version of the ISI-modulated SNN was implemented that modulated at the neuron level instead of the synapse level as described by Adam et al. [1]. The SNN and the IMSNN shared the same architecture with 200 hidden units and were trained on the whole MNIST dataset (60,000 training examples, 10,000 testing examples) for 15 epochs.

For input encoding, MNIST digits were converted to spike trains using Poisson rate encoding with firing rates between 28.5Hz and 100Hz based on pixel intensity. The membrane decay constant (β) was set to 0.99, and the firing threshold (θ) was fixed at 1.0 for all neurons. Both networks were trained using Adam optimiser with a learning rate of 1e-4 and a learning rate scheduler.

4.2 Classification Performance and Spike activity

Table 1 summarises the performance of the IMSNN compared to the conventional SNN on the MNIST dataset.

Model	**Training Time (s)**	**Train Accuracy (%)**	**Test Accuracy (%)**	**Spikes per Neuron**	**Spike Reduction (%)**
SNN	870.06	80.92	81.40	0.0885	-
IMSNN	1264.66	76.17	77.02	0.0590	33.3

Table 1: Performance comparison of IMSNN vs SNN on MNIST Dataset

The results show that the implemented IMSNN achieved a 33.3% reduction in spike activity compared to the conventional SNN, highlighting the energy efficiency benefits of ISI modulation. However, this improvement came with a trade-off of approximately 4.4 percentage points lower classification accuracy and a 45.4% increase in training time.

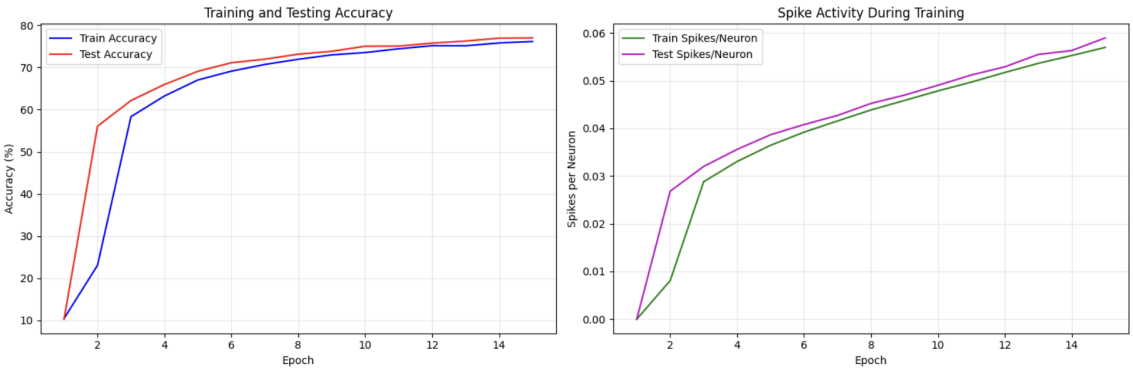

Figure 7: Training, Test accuracies and spiking activity of IMSNN across epochs

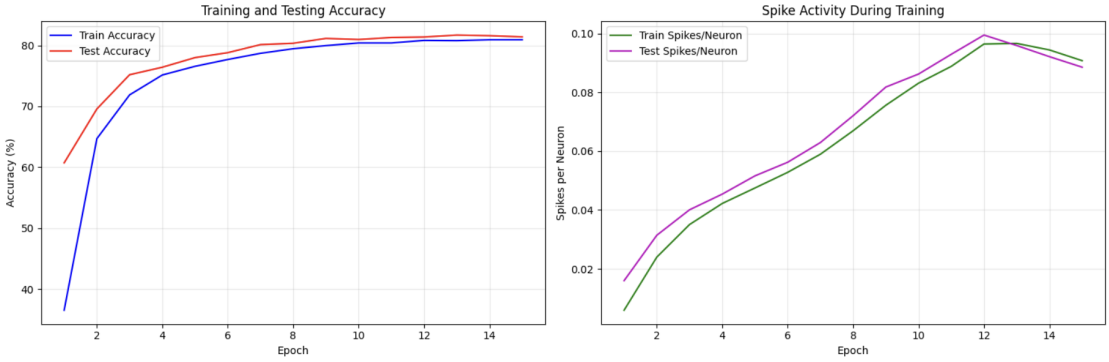

Figure 8: Training, Test accuracies and spiking activity of SNN across epochs

The classification accuracy for both models (77–81%) on MNIST was lower than typically expected and also lower than results reported by Adams et al. [1]. Several key implementation constraints contributed to this outcome:

1. **Simplified ISI-modulation:** In this study, modulation was applied at the neuron level instead of the synapse level — a limitation confirmed by [1] to directly impact both energy savings and accuracy.

2. **Hardware constraints:** The CPU-based setup restricted the model size, requiring a hidden layer with only 200 neurons rather than the optimal 500 neurons.

3. **Computational Overhead:** Implementing true per-synapse ISI modulation would have caused memory issues on the available hardware, forcing algorithmic simplifications.

4. **Limited training duration:** Hardware limitations also restricted training to just 15 epochs, potentially preventing full convergence.

Despite these limitations, the IMSNN still successfully demonstrated the core principle of energy savings through ISI modulation when compared to the baseline SNN.

4.3 Comparison with Other Energy Efficiency SNN Approaches

Table 2 provides a comparison of the IMSNN implemented in this work with other leading energy-efficient SNN methods reported in the literature.

Method	**Architecture**	**Test Accuracy (%)**	**Spike Reduction (%)**	**Reference**
IMSNN (Ours)	784-200-10	77.02	33.3	-
IMSNN (Original)	784-500-10	97.45	85.7	Adams et al. [1]
SNN (Ours)	784-200-10	81.40	-	-
Spike-Thrift	784-500-10	97.38	76.1	Kundu et al. [8]
AutoSNN	784-400-10	97.51	70.5	Na et al. [10]
BitSNN	784-512-10	96.85	64.3	Hu et al. [6]
FSpiNN	784-400-10	97.12	56.7	Putra and Shafique [11]

Table 2: Comparison with other energy-efficient SNN approaches on MNIST dataset

As shown in Table 2, the differences in achieved accuracy and spike reduction percentages clearly highlight the impact of both hardware limitations and network architecture. It is evident that using neuromorphic hardware and larger network sizes can substantially boost both performance and energy savings. Nonetheless, since both the SNN and IMSNN implementations in this work were developed under the same conditions, the underlying principle of achieving energy savings through ISI-modulation remains valid.

4.3.1 Training Dynamics

The training dynamics revealed that while both models exhibited an increase in spike activity during training, the IMSNN consistently maintained a lower spike rate

throughout the process. This indicates that the ISI-modulation mechanism was effectively encouraging longer interspike intervals as the network improved its classification capabilities.

4.4 Computational Considerations

The extended training time observed for the IMSNN highlights the additional computational overhead introduced by the ISI-modulation mechanism. In particular, each forward pass in the IMSNN required the calculation of Gaussian modulation factors based on the current ISI values, adding more computation compared to the static weight connections in the conventional SNN.

This trade-off between energy efficiency (reflected through reduced spike activity) and computational cost is a crucial consideration for neuromorphic applications. While IMSNNs may achieve lower energy consumption during inference, they demand higher computational resources during the training phase.

5 Conclusion and Future Work

5.1 Summary of Contributions

This research successfully developed and evaluated an energy-efficient spiking neural network architecture based on ISI-modulated synapses. The key contributions of this work are:

1. Development of a simplified IMSNN implementation that demonstrates the spike-reducing benefits of ISI modulation.

2. Empirical validation showing a 33% reduction in spike activity compared to conventional SNNs, while maintaining reasonable classification accuracy.

3. An in-depth analysis of the trade-offs between energy efficiency, computational overhead, and classification performance in IMSNNs.

Although the spike reduction achieved (33.3%) was less than the 90% reported by Adams et al. [1], the results still represent a meaningful improvement in energy efficiency, particularly for neuromorphic computing applications.

5.2 Implementation Challenges and Insights

A key insight from this research is the identification of practical challenges in deploying IMSNNs on conventional hardware. As discussed earlier, the simplified neuron-

level modulation used here (instead of synapse-level modulation) was necessary due to constraints such as:

1. **Memory Constraints:** A full implementation would require storing and updating ISI values for every synapse, which would quickly exceed available memory.

2. **Computational Complexity:** Performing per-synapse ISI modulation significantly increases the number of operations required during both forward and backward passes, thus limiting feasible training times.

3. **CUDA Compatibility Issues:** Initial attempts to use GPU acceleration encountered difficulties due to incompatibility with the complex tensor operations required for ISI modulation.

These challenges underline the gap between biologically inspired theoretical models and their practical realization on today's general-purpose computing platforms. To fully exploit the benefits of true synapse-level modulation, specialised neuromorphic hardware would be essential.

5.2.1 Future Work

Several promising directions for future research have emerged from this study:

1. **Advanced ISI Modulation:** Investigating more sophisticated ISI-modulation functions beyond the Gaussian model used in this work, potentially enabling even better energy-accuracy trade-offs.

2. **Hardware-Optimised Implementations:** Implementing true synapse-level modulation as outlined by [1] to achieve greater spike reductions.

3. **Neuromorphic Hardware Deployments:** Designing IMSNNs that are compatible with neuromorphic platforms such as Intel's Loihi or IBM's TrueNorth, which are inherently suited to event-driven processing.

4. **Integration with Other Techniques:** Combining IMSNNs with complementary approaches such as pruning, quantization, and adaptive time-step techniques discussed earlier in this paper to maximize energy savings.

5. **Applications to More Complex Tasks:** Extending the evaluation of IMSNNs to more complex datasets and tasks to assess whether the energy benefits scale with problem difficulty.

5.3 Broader Impact

As edge computing and IoT applications continue to expand, energy-efficient neural models like IMSNNs will play an increasingly critical role in enabling intelligent functionality on devices with limited power resources. This research contributes to the broader effort to bridge the gap between the exceptional energy efficiency of biological neural systems and the practical requirements of AI applications.

The observed trade-off between accuracy and energy consumption also highlights an important real-world consideration: depending on the specific application, users may choose to prioritize either performance or energy efficiency. IMSNNs offer a flexible mechanism to navigate this trade-off by adjusting the ISI-modulation parameters accordingly.

References

[1] Dylan Adams, Magda Zajaczkowska, Ashiq Anjum, Andrea Soltoggio, and Shirin Dora. Synaptic modulation using interspike intervals increases energy efficiency of spiking neural networks. (arXiv:2408.02961), August 2024. arXiv:2408.02961.

[2] Mohammed Alawad, Hong-Jun Yoon, and Georgia Tourassi. Energy efficient stochastic-based deep spiking neural networks for sparse datasets. In *2017 IEEE International Conference on Big Data (Big Data)*, page 311–318, December 2017.

[3] Abhiroop Bhattacharjee, Ruokai Yin, Abhishek Moitra, and Priyadarshini Panda. Are snns truly energy-efficient? — a hardware perspective. (arXiv:2309.03388), September 2023. arXiv:2309.03388 [cs].

[4] Thomas Bohnstingl, Stanisław Woźniak, Angeliki Pantazi, and Evangelos Eleftheriou. Online spatio-temporal learning in deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8894–8908, November 2023.

[5] Gourav Datta, Souvik Kundu, and Peter A. Beerel. Training energy-efficient deep spiking neural networks with single-spike hybrid input encoding, 2021.

[6] Yangfan Hu, Qian Zheng, and Gang Pan. Bitsnns: Revisiting energy-efficient spiking neural networks. *IEEE Transactions on Cognitive and Developmental Systems*, 16(5):1736–1747, October 2024.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. (arXiv:1412.6980), January 2017. arXiv:1412.6980 [cs].

[8] Souvik Kundu, Gourav Datta, Massoud Pedram, and Peter A Beerel. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3953–3962, 2021.

[9] Sixu Li, Zhaomin Zhang, Ruixin Mao, Jianbiao Xiao, Liang Chang, and Jun Zhou. A fast and energy-efficient snn processor with adaptive clock/event-driven computation scheme and online learning. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 68(4):1543–1552, 2021.

[10] Byunggook Na, Jisoo Mok, Seongsik Park, Dongjin Lee, Hyeokjun Choe, and Sungroh Yoon. Autosnn: Towards energy-efficient spiking neural networks. In *International Conference on Machine Learning*, pages 16253–16269. PMLR, 2022.

[11] Rachmad Vidya Wicaksana Putra and Muhammad Shafique. Fspinn: An optimization framework for memory-efficient and energy-efficient spiking neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):3601–3613, 2020.

[12] Qiaoyi Su, Shijie Mei, Xingrun Xing, Man Yao, Jiajun Zhang, Bo Xu, and Guoqi Li. Snn-bert: Training-efficient spiking neural networks for energy-efficient bert. *Neural Networks*, 180:106630, 2024.

[13] Congyi Sun, Haohan Sun, Jin Xu, Jianing Han, Xinyuan Wang, Xinyu Wang, Qinyu Chen, Yuxiang Fu, and Li Li. An energy efficient stdp-based snn architecture with on-chip learning. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(12):5147–5158, 2022.

[14] Hongze Sun, Wuque Cai, Baoxin Yang, Yan Cui, Yang Xia, Dezhong Yao, and Daqing Guo. A synapse-threshold synergistic learning approach for spiking neural networks. *IEEE Transactions on Cognitive and Developmental Systems*, 16(2):544–558, 2024.

[15] Dengyu Wu, Xinping Yi, and Xiaowei Huang. A little energy goes a long way: Build an energy-efficient, accurate spiking neural network from convolutional neural network. *Frontiers in neuroscience*, 16:759900, 2022.