Think Fast: Time Control in Varying Paradigms of Spiking Neural Networks

Steven Nesbit*
Andrew O'Brien*
Jocelyn Rego*
Edward Kim
scn43@drexel.edu
ao543@drexel.edu
jr3548@drexel.edu
ek826@drexel.edu
Drexel University
Philadelphia, PA, USA

Yijing Watkins
Gavin Parpart*
Carlos González*
yijing.watkins@pnnl.gov
gavin.parpart@pnnl.gov
carlos.gonzalezrivera@pnnl.gov
Pacific Northwest National Laboratory
Richland, WA, USA

Terrence C. Stewart
terrence.stewart@nrc-cnrc.gc.ca
National Research Council of Canada
Canada

Garrett T. Kenyon
gkenyon@lanl.gov
Los Alamos National Laboratory
Los Alamos, NM, USA

ABSTRACT

The state-of-the-art in machine learning has been achieved primarily by deep learning artificial neural networks. These networks are powerful but biologically implausible and energy intensive. In parallel, a new paradigm of neural network is being researched that can alleviate some of the computational and energy issues. These networks, spiking neural networks (SNNs), have transformative potential if the community is able to bridge the gap between deep learning and SNNs. However, SNNs are notoriously difficult to train and lack precision in their communication. In an effort to overcome these limitations and retain the benefits of the learning process in deep learning, we investigate novel ways to translate between them. We construct several network designs with varying degrees of biological plausibility. We then test our designs on an image classification task and demonstrate our designs allow for a customized tradeoff between biological plausibility, power efficiency, inference time, and accuracy.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**; **Modeling methodologies**; • **Hardware** → **Neural systems**.

KEYWORDS

neuro-inspired artificial intelligence, machine learning, neuromorphic computing

1 INTRODUCTION

While artificial neural networks were originally inspired by the form and function of the brain, the state-of-the-art deep learning network is quite distinct from its biological counterpart. These deep learning networks are highly engineered to achieve state-of-the art results in a narrow range of tasks such as game play [33], image recognition [37], and natural language processing[6].

Despite these successes, deep neural networks have substantial drawbacks such as requiring large amounts of power consumption [36], biological implausibility [18], and slow inference times on edge devices [27]. For example, if we examine one of the latest language models, GPT-3, the amount of training required for this model would have produced CO2 emissions equivalent to driving a car to the Moon and back, approximately 480,000 miles. The power consumption of this model is equivalent to the consumption of 126 Danish homes for an entire year [3].

One of the most notable differences between machine learning models and biological neurons is their communication language. Biological neurons communicate via electrical impulses, i.e. action potentials or spikes in an efficient coding scheme. This means that a high majority of neurons are not being activated simultaneously. In fact, many neurons are silent most of the time, with a single percent to a few percent of neurons spiking at any one time. Given a specific stimulus, only a highly selective, small subset of neurons will activate [32].

Thus, spiking neural networks (SNN) running on neuromorphic hardware present an alternative learning paradigm that is more energy efficient and biologically plausible than deep neural networks [34]. Training SNNs is difficult, so traditionally, learned weights from an artificial neural network can be translated into spiking rates for the SNN.

In this paper we describe multiple novel approaches to the conversion from an ANN to an SNN. Specifically, we enable the replacement of deep learning, ReLU activated nodes with spiking nodes of varying precision, speed, efficiency and biological plausibility. Our framework allows for neuromorphic systems that must maintain strict biological plausibility, to hybrid systems that can introduce additional bit precision or payloads that lead to both faster inference and lower power consumption. In essence, our proposed architecture controls inference time allowing for a tradeoff between model speed and power usage and accuracy depending on the computing resources available to the model user.

*Authors contributed equally to this research.

2 BACKGROUND

2.1 Non-Spiking Neuron Models

Early development of artificial neural networks began in 1943 with the introduction of non-spiking neuron models by McColloch and Pitts [24]. Inspired by biology, this early research laid the groundwork for neural networks, with neuron models based on thresholding and weights. In 1958, psychologist Frank Rosenblatt introduced the first trainable neural network, the Perceptron [28]. This network computed a weighted sum of inputs, passed through the Heaviside step function. Importantly, the weights of Rosenblatt's Perceptron could be learned through training, unlike the early neuron model developed by McColloch and Pitts. The use of the Heaviside step function in the Perceptron, however, presented a problem, as it is non-differentiable. This model, thus, was limited in that it could only handle linearly separable problems [26]. After a period of stagnation in neural network and non-spiking neuron model research, the field was revitalized in 2012. Aided by the computational power of modern GPUs, the success of deep neural networks in both speech [16] and large-scale image recognition tasks [21] thrust these non-spiking artificial networks into the spotlight. Today, this kind of feedforward continuous-valued neural network, often using the ReLU activation function in its hidden layers, is considered state-of-the-art for many tasks. However, the community is facing daunting headwinds of diminishing returns as these networks continue to scale. Thus, new paradigms of network models are needed to achieve greater returns.

2.2 Spiking Neuron Models

Integrate-and-fire (IF) - The integrate-and-fire neuron model was developed in 1907 by Louis Lapicque, far before the mechanisms of neural communication through action potentials were fully understood [22]. IF neurons model the membrane potential and voltage behavior of a neuron as input is injected. Within this model, a neuron's voltage, V, accumulates as an input current, $I(t)$, is applied, charging up until it reaches a threshold value, V_{th}. Once this threshold is reached, the IF neuron, with membrane capacitance C, produces a spike and resets [7].

$$I(t) = C\frac{dV(t)}{dt} \tag{1}$$

Despite the simplicity of this model, it has been widely used in neuroscience and modern neuromorphic computing, particularly for rate-based ANN-SNN conversion. While sufficient for some modeling tasks, the IF neuron model is largely simplified and leaves out important features of biological neurons, like the leaky diffusion of ions through a neuron's membrane.

Leaky Integrate-and-fire (LIF) - The leaky integrate-and-fire model extends the simple, deterministic IF model, incorporating biologically critical features like leakage. Derived from the Hodgkin-Huxley model [17, 41], LIF neurons were developed to better model the behavior of biological neurons [20] with the addition of a leakage term, R.

$$I(t) - \frac{V}{R} = C\frac{dV(t)}{dt} \tag{2}$$

2.3 Hybrid Spiking Neuron Models

Accumulators for Discretization of Neuron Models - Introduced to train hybrid SNNs with backpropogation, accumulator neurons allow for a gradual transition from non-spiking to spiking regimes [38]. These neurons are employed to gradually train a network, converting non-spiking to spiking functions, through the choice of a time-step like parameter, $\omega > 0$. In the context of spiking neurons ω allows for multiple spikes to be emitted per timestep, with $\frac{1}{\omega}$ denoting a spike's height. In this context, multiple-spikes in one timestep may also be considered a single spike with magnitude. The most recent iteration of Intel's neuromorphic chip, Loihi 2, permits spikes to be communicated with magnitude as an integer payload [39]. This capability permits functions to be matched to an arbitrary degree of accuracy, dependent upon the parameter ω.

Our work takes inspiration from the recent development of this accumulator model, expanding it to more biologically plausible neuron models. We can vary the biological plausibility of the accumulator i.e. leaky integrate and fire, to ensemble LIF, to rate coding neurons. **Our work enables the implementation to work on generalized neuromorphic hardware, where accumulator data structures do not exist.**

2.4 Advantages of Spiking Neural Networks

Communication through the language of spikes, in addition to expansive parallel processing and hierarchical organization, grants the brain its incredible energy efficiency. The brain uses only slightly over 20 W on average, not only for direct task-related communication, but also for resting metabolic consumption and maintaining electrical homeostasis. The brain uses a remarkably low amount of energy to perform the general tasks essential to keep it functioning, along with more-specific tasks like near constant auditory processing, object recognition, and dictionary learning tasks [40]. In contrast to the brain's 20 W expenditure, a modern GPU will use around 400 W to learn to perform object recognition on 1,000 categories [25, 29]. Guided by the communication principles of the brain, spiking neural networks present a more energy efficient alternative to standard artificial neural networks. By transmitting information through single-bit spike events, rather than through continuous values, spiking neural networks are able to process information at a lower power cost than continuous-valued networks. Particularly when implemented on neuromorphic hardware optimized for spiking communication, spiking networks exhibit far greater energy efficiency than their GPU executed analog counterparts. This has resulted in work that attempts to convert traditional ANNs to SNNs.

Traditional ANN-SNN Conversion Continuous-valued artificial neural networks may be converted to spiking neural networks in order to take advantage of their improved power efficiency for inference. Rate-based schemes are perhaps the most widely employed method of ANN-SNN conversion. Rate-based conversion methods rely on the ability of SNN neurons to match the firing rate of the activation function used by an analog ANN. ANNs that employ ReLU activation functions are particularly well suited to be converted to SNNs composed of integrate-and-fire (IF) neurons. The foundational principle for rate-based conversion was introduced

by Cao et. al [8], who demonstrated the equivalency of ReLU activation and spiking rates of IF neurons. Significant constraints, such as the removal of bias neurons and batch normalization, limit the experimental relevancy of their method. Still, the equivalency of spiking rates of IF neurons and ReLU activation function is a significant building block for rate-based ANN-SNN conversion. Deihl et. al [13] improved upon this method, introducing model-based and data-based normalization schemes to scale ANN weights based on the maximum activation of neurons in each layer. This rescaling of weights maps to SNN threshold balancing and prevents approximation error caused by too much or too little firing in converted SNN neurons. Further work proposed a means for more robust normalization using 99.9% of activation, rather than the maximum activation values, to scale weights for more accurate ANN-SNN conversion [30]. Rueckauer et. al additionally presented a "soft-reset" mechanism to overcome the information loss caused by the more biologically realistic "hard-reset" employed by earlier ANN-SNN conversion methods [8, 13].

2.5 Neuromorphic Hardware

Neuromorphic hardware are brain-inspired computing architectures that are designed to have advantages over the Von Neumann architecture with respect to neural network performance. Schuman et al. provide a useful taxonomy of neuromorphic hardware [31]. They note neuromorphic hardware can be partitioned into analog, digital, or mixed implementations. The digital systems can be further partitioned into field programmable gate arrays (FPGAs) and full custom or application specific integrated circuit chips. FPGAs are useful to achieve speed improvements over software when running neuromorphic software simulations, but are less effective than custom chips for small and low-powered systems. Examples of FPGAs include [10] [15] [1]. Examples of the chips include Intel's Loihi chip [11], SpiNNaker [4], and IBM's TrueNorth chip [2]. The Loihi chip has been shown to exhibit substantial energy savings [11] and robust performance [9, 19] on certain benchmark machine learning tasks [12].

Like digital systems, analog systems can be partitioned into programmable and custom chip designs [31]. The pros and cons of the various systems are roughly analogous to their digital counterparts. Mixed analog and digital systems have the biological plausibility of analog systems but can use digit components to compensate for analog weaknesses such as unreliability.

3 METHODOLOGY

Our objective is to utilize the strong learning capabilities of deep learning neural networks, while also augmenting the ability of spiking neural networks, see Figure 1 for a visualization of our methodology. To acheive this task, we constructed a feedforward neural network with one hidden layer for the task of MNIST image classification [23]. While some methods of ANN-SNN conversion transform analog input activations, such as pixel values, to Poisson firing rates [8, 13], we interpret input pixel values as constant input currents to avoid introducing further variability into the conversion [30]. This constant charge is added to the membrane potential of a neuron i at each timestep.

3.1 Basic ReLU-IF Conversion

Our continuous-valued ANN was implemented with the ReLU activation function. For a network with L layers and M^l units in each layer, the ReLU activation of a neuron can be denoted as:

$$a_i^l = max(0, \sum_{j=1}^{M^{l-1}} W_{ij}^l a_j^{l-1} + b_i^l) \tag{3}$$

where W^l is the weight matrix of between layers $l-1$ and l, with biases b^l, and a_j^{l-1} denotes the output activation value of neuron j in layer $l-1$. The dynamics of the ReLU activation can be mimicked by an IF neuron's spike rate. At each timestep, the weighted spike input is integrated into the neuron's membrane potential, the dynamics of which can be described by:

$$V_m(t) = V_m(t-1) + \sum_i w_i X_i(t) + b_i \tag{4}$$

where V_m denotes the membrane potential, X_i is the input spike train of IF neuron i, and w_i and b_i are the transferred weights and biases from the ANN [14].

3.2 Ensemble Neuron Models

Ensemble 1 - LIF Ensemble Neuron - Our LIF Ensemble Neuron (LEN) model is based on the premise that a collection of LIF neurons could be used to simulate the behavior of a ReLU neuron. The architecture of our LEN model is illustrated in Figure 2. The LEN takes the input and passes it to multiple LIF neurons, whose intercepts indicate the values at which each LIF neuron begins to fire. Our design allows for a variable number of LIF neurons. For our simulations, we used 9 LIF neurons with intercepts increasing in increments of .1 from .1 to .9 inclusive. The spike height of each LIF neuron is 1. Each neuron has a connection to a node computing a max function. The weight of the connection is identical to the node's intercept. At each timestep, the output of the max node is returned.

To implement our LEN model, we used LIF neurons from the Nengo software package [5]. Nengo uses the Neural Engineering Framework to construct neural models [35]. In the framework, a vector $\mathbf{a}$ of neurons is used to represent a stimulus vector $\mathbf{x}$ in a distributed way. Each neuron element i of $\mathbf{a}$ produces the following output: $a_i = G(\alpha_i \, \mathbf{e_i} \cdot \mathbf{x} + \beta_i)$ where G is a neuronal model, $\mathbf{e_i}$ is an orientation vector that encodes neuron i's preferred firing direction, α_i is a gain parameter, and β_i is a background bias current. For our LEN model, all G neuronal models are chosen to be the LIF neuron model.

For each a_i, its intercept can be set by adjusting its orientation, gain, and bias parameters in the following way. Its orientation parameter is set to a vector of 1s. The input to LIF neuron i can now be written as function J_i of the raw stimulus $\mathbf{x}$ as well as the gain and the bias, $J_i(\mathbf{x}) = \alpha_i \mathbf{x} + \beta_i$. In Nengo, the threshold for the LIF neuron is fixed at 1, so the firing rate for neuron i can be written as the following function of the stimulus:

$$r_i(\mathbf{x}) = \frac{1}{\tau_i^{ref} - \tau_i^{RC}(1 - \frac{1}{J_i(\mathbf{x})})} \tag{5}$$

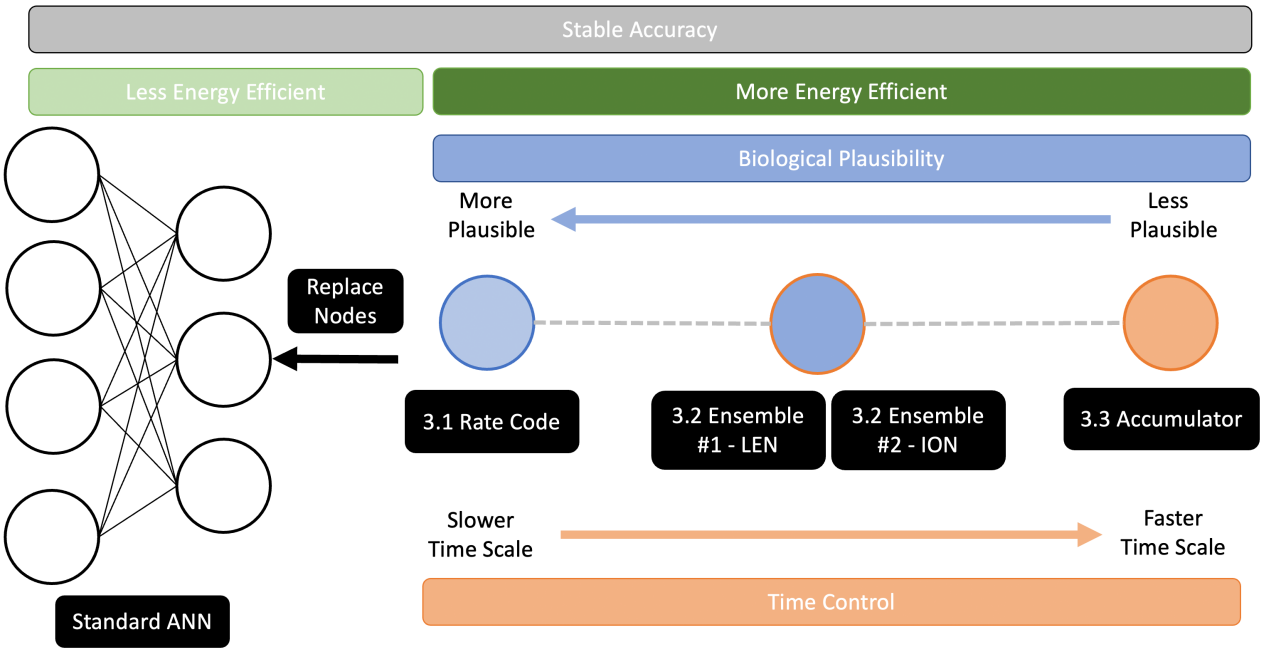

Figure 1: General overview of our methodology. As a first step, we obtain trained weights from an offline artificial neural network. These weights are then used in our framework, where we can replace the standard ANN nodes with our spiking nodes and control for time and biological plausibility. At one end of the framework, we use a simple rate coded leaky integrate and fire network that requires more time to achieve stable accuracy. On the other end, we can use accumulator integrate and fire neurons to emulate the performance of an ANN with ReLU activation with nearly instantaneous time scales. In this research effort, we describe this spectrum and provide implementation and experimental details for these states and everything inbetween. An important note is that our accuracy is stable across all methods.

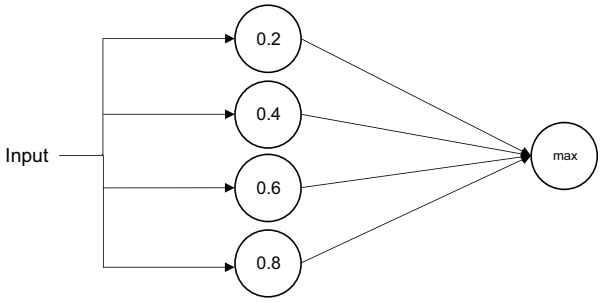

Figure 2: LIF Ensemble Neuron architecture: 4 LIF neurons, with intercepts increasing in increments of 0.2 from 0.2 to 0.8.

Where τ_i^{ref} and τ_i^{RC} are neuron i's refractory and time constants, respectively. A plot of the firing rate as a function of the stimulus vector is called a tuning curve. The plotted curves are characterized by the intercept indicating where the neuron begins to fire and the max-rate which is the maximum firing rate the neuron reaches on the input interval. The input interval for the raw stimulus is assumed to be in the range [-1,1]. Figure 3 shows tuning curves for 4 LIF neurons of intercepts 0.2, 0.4, 0.6, and 0.8 composing a hypothetical LEN model.

Using J_i and r_i, the parameters that result in the desired intercept can be found by solving a system of 2 equations. Since we know spiking begins when the neuron input J exceeds the threshold of 1, Equation 6 is one constraint that must hold for a desired intercept value $x_{intercept}$

$$1 = \alpha_i x_{intercept} + \beta_i \tag{6}$$

For an LIF neuron, the spiking frequency is an increasing function of the stimulus x and the the largest value in the domain is 1, so the max rate occurs at $x = 1$. Therefore, Equation 7 must hold for a desired maximum rate.

$$max_{rate} = \alpha_i + \beta_i \tag{7}$$

Solving the two equations yields unique values for α_i and β_i.

Ensemble 2 - Interfering Offset Neurons - The Interfering Offset Neurons (ION) model relies on multiple biased ternary IF neurons, as illustrated in Figure 4. Let $IF_t(x)$ be a single IF neuron with input x and threshold t, and $Acc_t(x)$ be an accumulator with spike height t. Note $IF_t(x) \approx Acc_t(x)$ for input from $-t$ to t. Additionally, if we introduce a bias to the input of a neuron $b = 2ti, i \in \mathbb{Z}$, then $IF_t(x - b) + b \approx Acc_t(x)$ from $-t + b$ to $t - b$. Since an IF neuron with threshold t saturates for inputs outside $-t$ to t, its output is given by Equation 8:

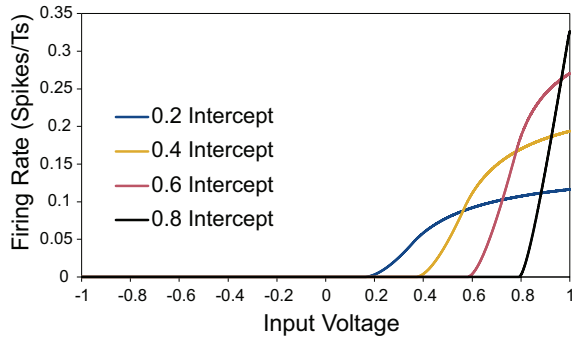

Figure 3: Tuning curves for a 4-neuron LEN model. Firing rates as functions of input voltage are shown for neurons with intercepts of 0.2, 0.4, 0.6, and 0.8.

$$IF_t(x-b) = \begin{cases} t & x > b+t \\ -t & x < b-t \\ Acc_t(x) - b & otherwise \end{cases} \tag{8}$$

For ION, we strategically bias the IF neurons so that the sum of their output approximates an accumulator. For ION with $2n+1$ neurons we bias the neurons from $-2tn$ to $2tn$ in multiples of $2t$. Note that with this spacing, for any input x there is only 1 neuron such that $b-t <= x <= b+t$. There are $n - \frac{b}{2t}$ neurons biased more than b, each outputting $-t$ for a total output of $-nt + \frac{b}{2}$. Additionally there are $n + \frac{b}{2t}$ opposing neurons biased less than b, each outputting t for a total output of $n + \frac{b}{2}$. The sum of all IF neuron outputs is then given by Equation 9:

$$ION(x) = nt + \frac{b}{2} + -nt + \frac{b}{2} + Acc_t(x) - b \tag{9}$$

While all but one of the IF neurons fire at every timestep, the summation of the neuron outputs can be performed locally, allowing for implementation on neuromorphic hardware.

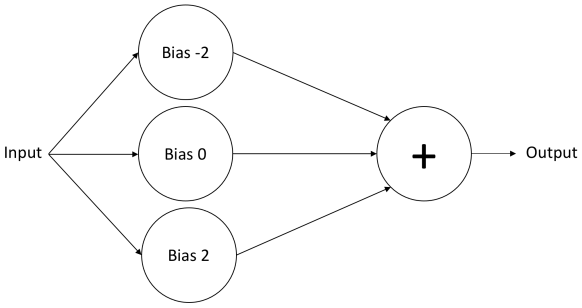

Figure 4: Interfering Offset Neuron model architecture with 3 IF neurons.

3.3 Accumulator Neurons

Accumulator neurons essentially represent a digital sampling of analog signaling. Therefore, better approximation can be achieved through more precise sampling. In the case of accumulator neurons, sampling precision is determined by the maximum number of spikes per timestep allowed. Figure 5 shows the performance of the accumulator models as a function of maximum spikes per timestep allotted. The non-leaky accumulator outperformed the leaky accumulator – with a rate constant of 10 – for all values in this comparison. It also achieved near peak performance far quicker than the leaky accumulator. These results essentially show how well these models can approximate a ReLU function, given that weights were transferred directly from a ReLU ANN to these models. Therefore, the non-leaky accumulator is able to approximate the ReLU better at peak performance, which it achieves with less spikes per timestep than the leaky accumulator.

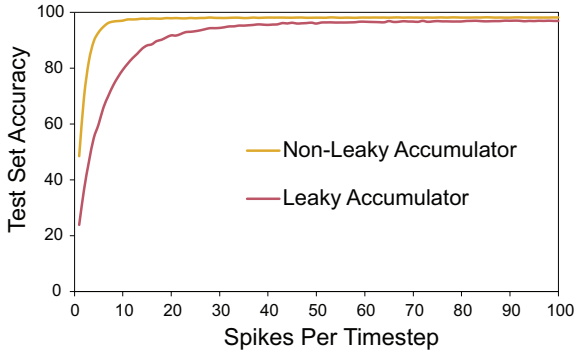

Figure 5: MNIST classification performance of SNNs with leaky and non-leaky accumulator nodes.

3.4 Custom Gradients

In addition to transferring the weights from the trained ANN with ReLU activation function, we trained each SNN model individually using their own gradients. The ION and LEN models approximated the ReLU functions with their outputs, so their gradients were the same as that of ReLU:

$$\frac{\partial V_{out}}{\partial V_{in}} = \begin{cases} 1 & V_{in} \geq V_{th} \\ 0 & V_{in} < V_{th} \end{cases} \tag{10}$$

Where $\frac{\partial V_{out}}{\partial V_{in}}$ represents the change in output voltage as a function of input voltage, V_{in} represents input voltage, V_{out} represents the output voltage, and V_{th} represents the threshold. For the other models, the firing rate of an LIF neuron is given by:

$$\frac{\partial V_{out}}{\partial t} = \begin{cases} \frac{V_{in}}{R} t & V_{in} \geq V_{th} \\ 0 & V_{in} < V_{th} \end{cases} \tag{11}$$

Where $\frac{\partial V_{out}}{\partial t}$ represents the firing rate given an input voltage, R represents the leak rate constant, and t represents a given timestep. A non-leaky IF neuron would be one with a rate constant of 1.

Differentiating Equation 11 with respect to input voltage gives the gradient of LIF neuron firing rate:

$$\frac{\partial^2 V_{out}}{\partial V_{in} \partial t} = \begin{cases} \frac{t}{R} & V_{in} \geq V_{th} \\ 0 & V_{in} < V_{th} \end{cases} \tag{12}$$

Because the accumulator models output multiple spikes per timestep, setting the value of t in Equation 12 to 1, treating it as a constant, and integrating with respect to input voltage gives the gradient for the accumulator models:

$$\frac{\partial V_{out}}{\partial V_{in}} = \begin{cases} \frac{V_{in}}{R} & V_{in} \geq V_{th} \\ 0 & V_{in} < V_{th} \end{cases} \tag{13}$$

As with the rate code, a leak rate constant of 1 would make the accumulator non-leaky, and a rate constant greater than 1 would produce a leaky accumulator. Integrating Equation 13 with respect to V_{in} gives the output of the accumulator neuron in terms of spikes per timestep for a given input:

$$V_{out} = \begin{cases} \frac{V_{in}^2}{2R} & V_{in} \geq V_{th} \\ 0 & V_{in} < V_{th} \end{cases} \tag{14}$$

4 RESULTS

4.1 MNIST Classifier Performance

Training the ReLU 2-layer ANN MNIST classifier with stochastic gradient descent with momentum produced test set accuracies ranging from 98.08% to 98.25% over 10 iterations. Therefore, accuracies that are within 0.17% of each other would be considered within the margin of error. For reproducible results, we seeded the random number generator to 0, resulting in a ReLU ANN MNIST classification accuracy of 98.09%. The weights from this training regime were then transferred to the other models and their accuracies are shown in Table 1. The non-leaky models generally gave better performance in this regime, with the non-leaky accumulator at 100 spikes per timestep achieving the best accuracy of 98.19%, the ION model over 100 timesteps classifying with 96.93% accuracy, and the non-leaky rate code over 100 timesteps performing at 97.03% accuracy. The leaky accumulator with a rate constant of 10 at 100 spikes per timestep performed the best of the leaky models with 96.86% accuracy, followed by the LEN model with a rate constant of 10 run over 100 timesteps at 95.42% accuracy, with the leaky rate code with a rate constant of 10 run over 100 timesteps performing the worst at 80.22%.

4.2 Time Control

Each network was evaluated to see how many timesteps it would need to be run to match the performance of the leaky rate code of 80.22% at 100 timesteps. The non-leaky models again generally gave better results in this measure, having the shortest inference times. The non-leaky accumulator was run for 2 timesteps, the ION model was run for 8 timesteps, and the non-leaky rate code was run for 17 timesteps. For the leaky models, each with a rate constant of 10, the leaky accumulator required 13 timesteps, the LEN model 20 timesteps, and finally the leaky rate code was run for 100 timesteps.

In addition, each non-accumulator model was compared for performance as a function of number of timesteps run, as shown in Figure 6. These models all showed accuracies at the level of chance when run for less than 10 timesteps, so Figure 6 shows performance for the range of 10 to 100 timesteps. The accumulator models are not included in this comparison because they were able to achieve their peak performance when run for less than 10 timesteps, and their performance was more dependent on the maximum number of spikes per timestep allotted. Figure 6 shows that non-leaky models showed a much faster increase in performance at the lower timestep values than the leaky models, which both have rate constants of 10 in this comparison. In addition, the ensemble models showed quicker performance increases than the rate code models, with the leaky rate code showing a very slow increase in performance, even up to 100 timesteps.

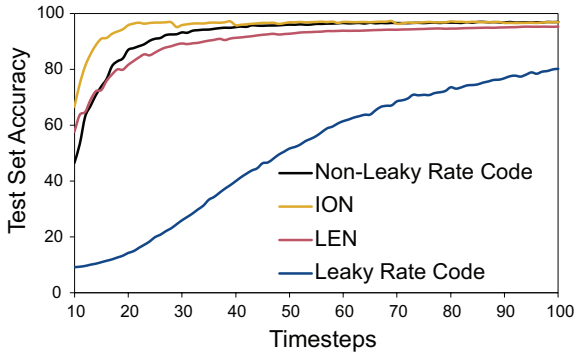

Figure 6: MNIST classification performance of SNNs with rate-coded nodes and ensemble nodes.

4.3 Power Efficiency

Relative power consumption was determined by the number of spikes per timestep required to achieve the same accuracy as mentioned in the previous section – 80.22%. This was done under the assumption that each spike would require roughly the same amount of energy to produce, with the accumulator output being represented by single spikes with integer payloads. Better power efficiency was generally achieved by leaky models. The leaky accumulator, leaky rate code, and LEN models showed power consumptions of 0.08, 0.44, and 5.05 spikes per timestep, respectively. The non-leaky accumulator, non-leaky rate code, and ION models showed power consumptions of 0.50, 0.41, and 6.88 spikes per timestep, respectively. The ensemble models were able to achieve power values with multiple spikes per timestep because they consist of multiple neurons.

4.4 Custom Training

After analyzing the model performances from transferring ANN weights to SNN models, each model was trained on its own to analyze performance with model-specific weights. Again, each model was trained with a random seed of 0 to achieve reproducible results, but results within 0.17% of each other were considered to be similar, based on our trainings of the ReLU ANN ranging from 98.08% to 98.25% accuracy. All models tested were within the range of accuracy of the ReLU ANN. Therefore, each of the SNN models

Model	Weight Transfer (Accuracy %)	Custom Training (Accuracy %)	Inference Time (Timesteps)	Power Consumption (Spikes/Timestep)
ReLU-ANN	98.09	98.08-98.25	1	1
Non-leaky rate code	97.03	98.12	17	0.41
Leaky rate code	80.22	98.03	100	0.44
Non-leaky accumulator	**98.19**	98.15	**2**	0.50
Leaky accumulator	96.86	98.07	13	**0.08**
Interfering Offset Neurons	96.93	98.16	8	6.88
LIF Ensemble Neuron	95.42	**98.25**	20	5.05

Table 1: Classification accuracy, inference time, and power consumption of neural networks on MNIST.

presented here are capable of matching the accuracy of a ReLU ANN. The only significant difference found from the custom training was that the LEN model achieved an accuracy of 98.25%, which is greater than that achieved by the leaky rate code of 98.03%.

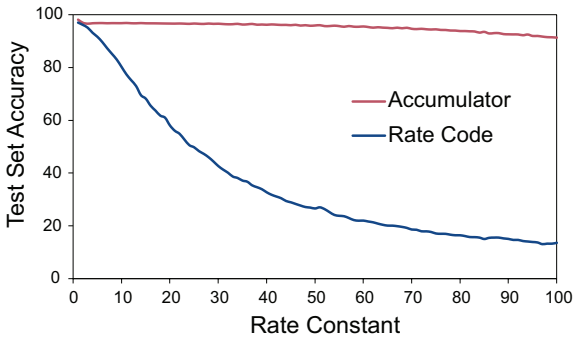

Figure 7: MNIST classification performance of accumulator and rate coded models as a function of leak rate constant.

4.5 Leakiness vs Performance

Figure 7 shows the MNIST classification performance of accumulator and rate coded models as a function of leak rate constant; the larger the rate constant, the leakier the model, with a rate constant of 1 representing a non-leaky model. The accumulator model was run with a maximum of 100 spikes per timestep and the rate coded model was run for 100 timesteps. The rate coded model shows a sharp decline in performance from the non-leaky model as the leak rate constant is increased. In contrast, the accumulator model shows a far more gradual decline in performance as leak rate constant is increased. Its initial decrease from a rate constant of 1 to 2 is similar to that seen in the rate coded model, however after this the accumulator performance decays very slowly. As seen in Table 1, increasing the accumulator rate constant from 1 (non-leaky) to 10 significantly increased the power efficiency from 0.50 to 0.08 spikes per timestep. Therefore, Figure 7 shows how the accumulator model power efficiency can be adjusted significantly by changing the rate constant without sacrificing much performance.

5 CONCLUSION

In this paper, we put forward multiple methods for converting continuous valued artificial neural networks to spiking neural networks. Taking inspiration from the brain, we implemented networks consisting of spiking nodes with varying degrees of biological realism, from rate-coded neurons to accumulator neurons to ensembles of leaky integrate and fire neurons. We compared the performances of these approaches in accuracy, speed, and power consumption to the standard ReLU activated ANN. Along with evaluating how these spiking architectures performed with weights transferred from the ReLU activated ANN, we developed a novel, differentiable activation function shown in Equation 14. Thus, we were able to directly train SNN architectures consisting of our nodes with back-propogation and similarly compare power consumption, speed, and accuracy.

Our results display a trade-off in accuracy, speed, and power efficiency across different leaky and non-leaky neuron models. While non-leaky models outperform leaky models in both accuracy, with ANN transferred weights, and time efficiency, this is to be expected. Since these non-leaky models have no membrane leakage, they are able to fire more rapidly and, thus, approximate functions more quickly and accurately. These non-leaky models, however, are less biologically plausible than our leaky implementations. The more biologically plausible leaky accumulator implementation far outperforms all other models in power efficiency. It is unsurprising that our ensemble LEN and ION models require more power, as they are comprised of sets of many neurons. Training with our model-specific gradients yielded results within the standard ReLU activated ANN's margin of error for all neuron models. The LEN model performs exceptionally well in this case and is of particular interest for future work. For future work, we would like to use our LEN implementation for on-chip unsupervised dictionary learning based image classification tasks on neuromorphic hardware that permits only single-bit spikes. We would also like to explore how changing the number of neurons and leak rate constant in the LEN implementation affects performance.

Taken in sum, our results display how our biologically plausible frameworks, implementable on neuromorphic hardware, allow for a user controllable trade-off between accuracy, power usage, and speed.

REFERENCES

[1] Byungik Ahn. 2014. Computation of deep belief networks using special-purpose hardware architecture. In *2014 International Joint Conference on Neural Networks (IJCNN)*. 141–148. https://doi.org/10.1109/IJCNN.2014.6889903

[2] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, Brian Taba, Michael Beakes, Bernard Brezzo, Jente B. Kuang, Rajit Manohar, William P. Risk, Bryan Jackson, and Dharmendra S. Modha. 2015. TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 10 (2015), 1537–1557. https://doi.org/10.1109/TCAD.2015.2474396

[3] Lasse F Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051* (2020).

[4] Rui Araújo, Nicolai Waniek, and Jorg Conradt. 2014. Development of a Dynamically Extendable SpiNNaker Chip Computing Module. 821–828. https://doi.org/10.1007/978-3-319-11179-7_103

[5] Trevor Bekolay, James Bergstra, Eric Hunsberger, Travis DeWolf, Terrence Stewart, Daniel Rasmussen, Xuan Choo, Aaron Voelker, and Chris Eliasmith. 2014. Nengo: a Python tool for building large-scale functional brain models. *Frontiers in Neuroinformatics* 7, 48 (2014), 1–13. https://doi.org/10.3389/fninf.2013.00048

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[7] Anthony N. Burkitt. 2006. A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input. *Biological cybernetics* 95, 1 (2006), 1–19. https://doi.org/10.1007/s00422-006-0068-6

[8] Yongqiang Cao, Yang Chen, and Deepak Khosla. 2015. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision* 113, 1 (2015), 54–66.

[9] John Carter, Jocelyn Rego, Daniel Schwartz, Vikas Bhandawat, and Edward Kim. 2020. Learning Spiking Neural Network Models of Drosophila Olfaction. In *International Conference on Neuromorphic Systems 2020*. 1–5.

[10] Nimet Dahasert, İsmail Öztürk, and Recai Kiliç. 2012. Implementation of Izhikevich neuron model with field programmable devices. In *2012 20th Signal Processing and Communications Applications Conference (SIU)*. 1–4. https://doi.org/10.1109/SIU.2012.6204544

[11] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 38, 1 (2018), 82–99. https://doi.org/10.1109/MM.2018.112130359

[12] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A. Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R. Risbud. 2021. Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook. *Proc. IEEE* 109, 5 (2021), 911–934. https://doi.org/10.1109/JPROC.2021.3067593

[13] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. 2015. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International joint conference on neural networks (IJCNN)*. IEEE, 1–8.

[14] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. 2020. RMP-SNN: Residual Membrane Potential Neuron for Enabling Deeper High-Accuracy and Low-Latency Spiking Neural Network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (June 2020).

[15] H. Hikawa. 2003. A digital hardware pulse-mode neuron with piecewise linear activation function. *IEEE Transactions on Neural Networks* 14, 5 (2003), 1028–1037. https://doi.org/10.1109/TNN.2003.816058

[16] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. 2012. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine* 29, 6 (2012), 82–97. https://doi.org/10.1109/MSP.2012.2205597

[17] A. L. Hodgkin and A. F. Huxley. 1990. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Bltn Mathcal Biology* 52 (1990), 25–71. https://doi.org/10.1007/BF02459568

[18] Bernd Illing, Wulfram Gerstner, and Johanni Brea. 2019. Biologically plausible deep learning — But how far can we go with shallow networks? *Neural Networks* 118 (2019), 90–101. https://doi.org/10.1016/j.neunet.2019.06.001

[19] Edward Kim, Jessica Yarnall, Priya Shah, and Garrett T Kenyon. 2019. A Neuromorphic Sparse Coding Defense to Adversarial Images. In *Proceedings of the International Conference on Neuromorphic Systems*. 1–8.

[20] C. Koch and I. Segev. 1998. Methods in Neuronal Modeling: from Ions to Networks, 2nd Edition. (1998).

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, Vol. 25. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[22] Louis Lapicque. 1907. Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *Journal of Physiol Pathol Générale* 9 (1907), 620–635.

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791

[24] Warren S. McCulloch and Walter Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5, 4 (1943), 115–133.

[25] Maxim Milakov. 2014. Deep Learning With GPUs. nvidia.co.uk/docs/IO/147844/Deep-Learning-With-GPUs-MaximMilakov-NVIDIA.pdf

[26] Marvin Minsky and Seymour Papert. 1969. Perceptrons. (1969).

[27] Evgeny Ponomarev, Sergey Matveev, Ivan Oseledets, and Valery Glukhov. 2021. Latency Estimation Tool and Investigation of Neural Networks Inference on Mobile GPU. *Computers* 10 (08 2021), 104. https://doi.org/10.3390/computers10080104

[28] Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* 65, 6 (1958), 386.

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

[30] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. 2017. Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification. *Frontiers in Neuroscience* 11 (2017). https://doi.org/10.3389/fnins.2017.00682

[31] Catherine D. Schuman, Thomas E. Potok, Robert M. Patton, J. Douglas Birdwell, Mark E. Dean, Garrett S. Rose, and James S. Plank. 2017. A Survey of Neuromorphic Computing and Neural Networks in Hardware. https://doi.org/10.48550/ARXIV.1705.06963

[32] Shy Shoham, Daniel H O'Connor, and Ronen Segev. 2006. How silent is the brain: is there a "dark matter" problem in neuroscience? *Journal of Comparative Physiology A* 192, 8 (2006), 777–784.

[33] 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. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (2016), 484–489. https://doi.org/10.1038/nature16961

[34] Martino Sorbaro, Qian Liu, Massimo Bortone, and Sadique Sheik. 2020. Optimizing the Energy Consumption of Spiking Neural Networks for Neuromorphic Applications. *Frontiers in Neuroscience* 14 (2020). https://doi.org/10.3389/fnins.2020.00662

[35] Terrence C. Stewart. 2012. *A Technical Overview of the Neural Engineering Framework*. Technical Report. Centre for Theoretical Neuroscience.

[36] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and Policy Considerations for Modern Deep Learning Research. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 09 (Apr. 2020), 13693–13696. https://doi.org/10.1609/aaai.v34i09.7123

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1–9. https://doi.org/10.1109/CVPR.2015.7298594

[38] Aaron R. Voelker, Daniel Rasmussen, and Chris Eliasmith. 2020. A Spike in Performance: Training Hybrid-Spiking Neural Networks with Quantized Activation Functions. *CoRR* abs/2002.03553 (2020).

[39] Sally Ward-Foxton. 2021. Intel offers Loihi 2 neuromorphic chip and software framework. embedded.com/intel-offers-loihi-2-neuromorphic-chip-and-software-framework/

[40] Yijing Watkins, Edward Kim, Andrew Sornborger, and Garrett T Kenyon. 2020. Using sinusoidally-modulated noise as a surrogate for slow-wave sleep to accomplish stable unsupervised dictionary learning in a spike-based sparse coding model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 360–361.

[41] Zhou S. Zhi T. Du Z. Zhang, L. and Y. Chen. 2019. TDSNN: From Deep Neural Networks to Deep Spike Neural Networks with Temporal-Coding. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 1, 1319–1326. https://doi.org/10.1609/aaai.v33i01.33011319