# RATE CODING OR DIRECT CODING: WHICH ONE IS BETTER FOR ACCURATE, ROBUST, AND ENERGY-EFFICIENT SPIKING NEURAL NETWORKS?

*Youngeun Kim*　*Hyoungseob Park*　*Abhishek Moitra*　*Abhiroop Bhattacharjee*
*Yeshwanth Venkatesha*　*Priyadarshini Panda*

Yale University, USA

## ABSTRACT

Recent Spiking Neural Networks (SNNs) works focus on an image classification task, therefore various coding techniques have been proposed to convert an image into temporal binary spikes. Among them, rate coding and direct coding are regarded as prospective candidates for building a practical SNN system as they show state-of-the-art performance on large-scale datasets. Despite their usage, there is little attention to comparing these two coding schemes in a fair manner. In this paper, we conduct a comprehensive analysis of the two codings from three perspectives: accuracy, adversarial robustness, and energy-efficiency. First, we compare the performance of two coding techniques with various architectures and datasets. Then, we measure the robustness of the coding techniques on two adversarial attack methods. Finally, we compare the energy-efficiency of two coding schemes on a digital hardware platform. Our results show that direct coding can achieve better accuracy especially for a small number of timesteps. In contrast, rate coding shows better robustness to adversarial attacks owing to the non-differentiable spike generation process. Rate coding also yields higher energy-efficiency than direct coding which requires multi-bit precision for the first layer. Our study explores the characteristics of two codings, which is an important design consideration for building SNNs[1].

***Index Terms*—** Spiking neural network, rate coding, direct coding, energy-efficiency, adversarial robustness

## 1. INTRODUCTION

Spiking Neural Networks (SNNs) [1, 2] have gained increasing attention as a promising paradigm for low-power intelligence. Inspired by biological neuronal functionality, SNNs process visual information with binary spikes over multiple timesteps. The majority of works on SNNs have so far focused on a static image classification problem [1] to develop an energy-efficient alternative to Artificial Neural Networks (ANNs). Recent works utilize a backpropagation rule for training and show that this yields state-of-the-art performance with fewer number of timesteps compared to other SNN optimization techniques [3].

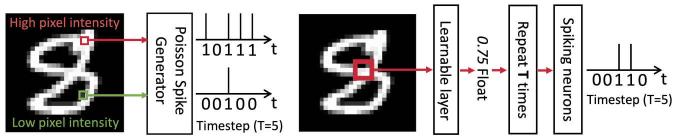
To convert a static image into binary spike trains, various coding schemes have been proposed for the image classification task [4, 5, 6]. Among them, rate coding and direct coding are prominent as these coding schemes enable SNNs to be trained on large-scale datasets [3, 7, 8, 9, 10]. The scalability of SNNs on large-scale datasets is important considering the growing demand for processing large-scale data on resource-constrained devices. Rate coding converts pixel intensity into a spike train where the number of spikes is proportional to the pixel intensity [11, 12, 10]. On the other hand, direct coding uses a trainable layer to generate float value for each timestep [3, 7, 8, 13]. Following the recent trend, we focus on the comparison between rate coding and direct coding even though they require more spikes than other coding techniques such as temporal coding [4, 14] (details discussed in Section 2).

In this paper, we objectively compare rate coding and direct coding in the same experimental settings from three different perspectives. Note, it is difficult to make a fair comparison between these coding schemes from previous works [3, 7, 8, 9, 15, 10] which use different architectures, neuron models, and hyperparameters. To this end, we first report the **accuracy** on various architectures (*i.e.*, multi-layer perceptron, VGG5, and VGG9) and datasets (*i.e.*, MNIST, CIFAR10, and CIFAR100). We present the change of accuracy with respect to the number of timesteps since achieving high performance in a low-latency regime is one of the important topics in SNN studies. **Adversarial robustness** [16] is recently highlighted as a new feature of SNNs [17]. We explore the adversarial robustness to provide a better understanding of the advantages and disadvantages of each coding scheme. Finally, **energy-efficiency** is a key evaluation metric for SNNs, therefore we estimate the energy cost of coding schemes on a digital hardware platform [18].

Our work provides several noteworthy observations for SNN design. Direct coding yields better accuracy than rate coding. The advantage of direct coding increases for deeper architecture and larger datasets. In terms of robustness, rate coding shows better robustness with respect to Fast Gradient

The research was funded in part by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, and the National Science Foundation (Grant#1947826).

[1]The code is made available at https://github.com/Intelligent-Computing-Lab-Yale/Rate-vs-Direct



(a) Rate coding (b) Direct coding

**Fig. 1**. Illustration of rate coding and direct coding with total timesteps $T = 5$.

Sign Method (FGSM) [16] and Projected Gradient Descent (PGD) [19] attacks due to the non-differentiable Poisson spike generator. On the other hand, since the direct coding layer is differentiable, malicious backward gradients can degrade the accuracy significantly. Furthermore, rate coding achieves higher energy efficiency from binary input spikes than direct coding where multi-bit operation is required.

## 2. PRELIMINARIES: INPUT CODING SCHEMES

Various coding schemes has been proposed for image classification with SNN. Temporal coding [14, 4] generates one spike per neuron in which spike latency is inversely proportional to the pixel intensity. Phase coding encodes temporal information into spike patterns based on a global oscillator [20]. Burst coding transmits the burst of spikes in a small-time duration, increasing the reliability of synaptic communication between neurons [5]. With these coding schemes, most prior work successfully train shallow networks, however, they are difficult to be applied when the network and dataset size are scaled up. Rate coding can be applied on such large-scale settings, and therefore recent state-of-the-art methods utilize this coding [11, 12, 15, 10]. This coding scheme encodes the input by generating a spike train over $T$ timesteps, where the total number of spikes is proportional to the magnitude of input values. The spikes are sampled from a Poisson distribution. Fig. 1(a) shows the rate coding mechanism. Direct coding [3] uses the floating-point inputs directly in the first layer. As shown in Fig. 1(b), we pass the input image (or RGB pixel values) through the first convolution layer which generates floating point outputs. Note, the float output values are repeated for $T$ time-steps of SNN processing. These outputs are then processed through a layer of spiking neurons that generates binary spikes. Recently, Guo *et al.* [6] conducted a comprehensive analysis on various coding schemes. However, they use a shallow 2-layered network with Spike-Timing-Dependent Plasticity (STDP), which is difficult to apply to deeper models working on large scale datasets. Also, direct coding has not been compared to other coding schemes although they are leveraged in state-of-the-art works [3, 8].

## 3. RATE CODING VS. DIRECT CODING

In this section, we first present the Leaky Integrate-and-Fire (LIF) neuron model and training algorithm used in our comparison. After that, we analyze the strengths and weaknesses of two codings from three different perspectives.

### 3.1. Neuron model

We use LIF neuron for our comparison. The membrane potential $U_m$ of LIF neuron stores the temporal spike information. When an input signal $I(t)$ is given to the LIF neuron, the membrane potential is varied:

$$\tau_m \frac{dU_m}{dt} = -U_m + I(t), \tag{1}$$

where, $\tau_m$ is the time constant for the membrane potential decay. Since the voltage and current have continuous values, we convert the differential equation into a discrete version following the previous works [3, 9]. For each timestep $t$, we can formulate the membrane potential $u_i^t$ of a single neuron $i$ as:

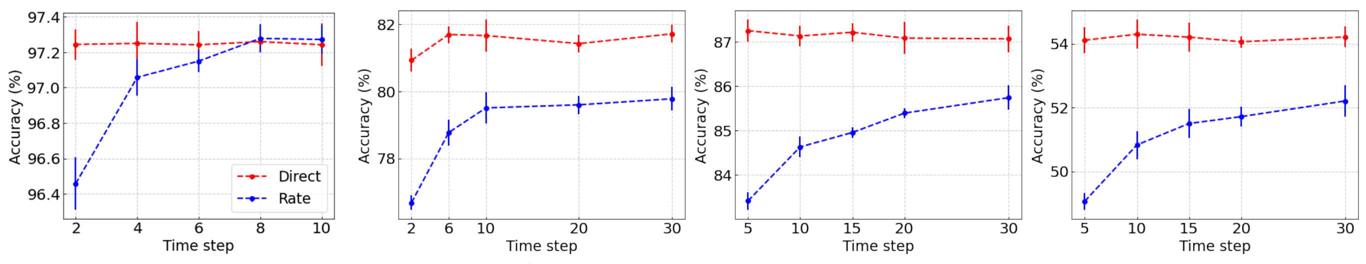
$$u_i^t = (1 - \frac{1}{\tau_m})u_i^{t-1} + \frac{1}{\tau_m}\sum_j w_{ij} o_j^t. \tag{2}$$

Here, the current membrane potential consists of the decayed membrane potential from previous timesteps and the weighted spike signal from the pre-synaptic neurons. The notation $w_{ij}$ is for weight connections between neuron $i$ and neuron $j$. The neuron $i$ accumulates voltage and generates a spike output $o_i^t$ whenever $u_i^t$ exceeds the firing threshold $\theta$. After the neuron fires, the membrane potential is lowered by the amount of the threshold.

### 3.2. Optimizing SNN using backpropagation

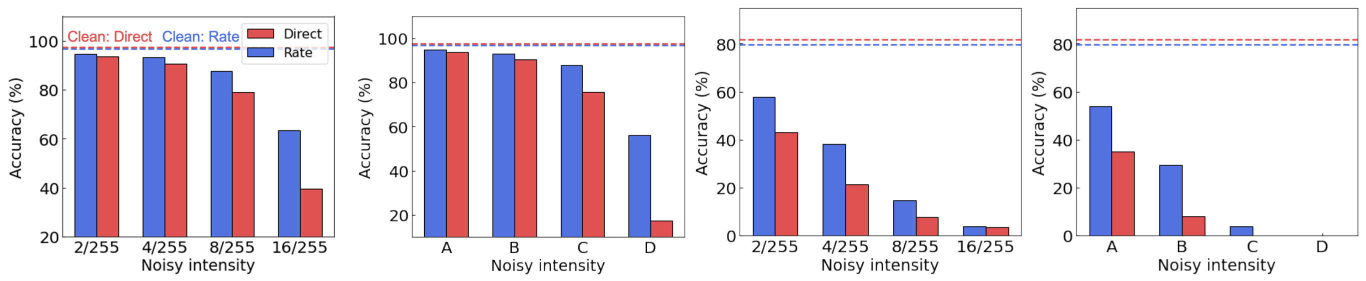
Various optimization techniques for SNNs have been proposed in the past decade. Among them, surrogate gradient learning has been mainly utilized in recent SNN works because it shows state-of-the-art performance in a low-latency regime [7, 8, 9, 21, 22]. Further, surrogate gradient based backpropagation learning can be implemented using well-established machine learning frameworks like PyTorch [23]. In our comparison, we use surrogate gradient learning as a baseline optimization technique.

Given input spikes, we train SNNs based on gradient optimization. Intermediate LIF neurons accumulate pre-synaptic spikes and generate output spikes (Eq. 2). Spike information is passed through all layers and stacked or accumulated at the output layer (*i.e.*, prediction layer). This enables the accumulated temporal spikes to be represented as probability distribution after the softmax function. From the accumulated membrane potential, we can compute the cross-entropy loss for SNNs. Based on the calculated loss, we compute the gradients of each layer $l$. Here, we use spatio-temporal back-propagation (STBP), which accumulates the gradients over all timesteps [24]. We can formulate the gradients at the layer $l$ by chain



(a) MLP-MNIST (b) VGG5-CIFAR10 (c) VGG9-CIFAR10 (d) VGG9-CIFAR100

**Fig. 2**. The accuracy change with respect to the number of timesteps. We run each configuration 5 times and report the mean and standard deviation (vertical line).



(a) MLP-FGSM (b) MLP-PGD (c) VGG5-FGSM (d) VGG5-PGD

**Fig. 3**. The adversarial robustness with respect to FGSM attack and PGD attack. We present the clean accuracy using dotted lines. In the PGD attack experiments, A-PGD[2/255, 1/255,10], B-PGD[4/255, 1/255,10], C-PGD[8/255, 4/255,10], D-PGD[16/255, 4/255,10]. We use timestep 10 for all experiments.

rule as:

$$\frac{\partial L}{\partial W_l} = \begin{cases} \sum_t \left(\frac{\partial L}{\partial O_l^t}\frac{\partial O_l^t}{\partial U_l^t} + \frac{\partial L}{\partial U_l^{t+1}}\frac{\partial U_l^{t+1}}{\partial U_l^t}\right)\frac{\partial U_l^t}{\partial W_l}, & \text{if } l : \text{hidden} \\ \frac{\partial L}{\partial U_l^T}\frac{\partial U_l^T}{\partial W_l}. & \text{if } l : \text{output} \end{cases} \tag{3}$$

Here, $O_l^t$ and $U_l^t$ are output spikes and membrane potential at timestep $t$ for layer $l$, respectively. LIF neurons in hidden layers generate spike output only if the membrane potential $u_i^t$ exceeds the firing threshold, leading to non-differentiability. To deal with this problem, we use an approximate gradient:

$$\frac{\partial o_i^t}{\partial u_i^t} = \max\{0, 1 - |\frac{u_i^t - \theta}{\theta}|\}. \tag{4}$$

Overall, network parameters at the layer $l$ are updated based on the gradient value (Eq. 3).

## 3.3. Experimental settings

In our experiments, we use three architectures on (*i.e.*, MLP 784-800-10, VGG5, and VGG9) three public datasets (*i.e.*, MNIST, CIFAR10, and CIFAR100). MNIST [25] contains gray-scale images of size 28 × 28. CIFAR10 [26] consists of 60,000 RGB color images of size 32 × 32. (50,000 for training / 10,000 for testing) with 10 categories. CIFAR100 has the same configuration as CIFAR10, except it contains 100 categories. For all datasets, we use random horizontal flip for data augmentation. Our implementation is based on PyTorch framework [23]. We set the total number of epochs to 60, 100, and 100 for MNIST, CIFAR10, and CIFAR100, respectively. During training, we utilize step-wise learning rate scheduling with a decay factor of 10 at 50% and 75% of the total epochs. We train the networks with Adam optimizer with an initial learning rate $1e-4$. For LIF neuron, we set time constant $\tau_m$ and threshold $\theta$ to 2 and 1, respectively.

## 3.4. Accuracy comparison

Fig. 2 shows the accuracy of rate-encoded SNN and direct-encoded SNN with respect to the number of timesteps. From the experimental results, we observe the following: (1) In general, direct coding brings higher accuracy than rate coding especially with small number of timesteps. (2) As the rate-coded SNN is trained with larger number of timesteps, the performance gap between the two coding decreases. (3) As the dataset and network architecture gets more complicated, the performance gap between the two coding increases. The reason for higher performance of direct-coded SNNs is a trainable input coding layer (Fig. 1(b)). The weights of the coding layer are trained for minimizing cross-entropy loss, providing optimal float amplitude.

## 3.5. Robustness studies

We compare the robustness of rate-coded and direct-coded SNNs against adversarial attacks. For this work, we consider two adversarial attacks. FGSM attack [16] is a single-step attack based on backward gradients. For the input image $x$, adversarial image $x_{adv}$ is generated by adding the sign of gradients scaled by $\epsilon$:

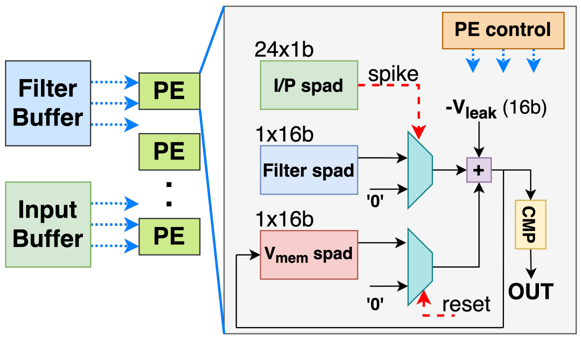


**Fig. 4**. A pictorial representation of a PE in a 16-bit *Eyeriss* platform for SNN evaluation.

$$x_{adv} = x + \epsilon \; sign(\nabla_x(\mathcal{L}(x, y_{true}))), \tag{5}$$

where, $y_{true}$ stands for the ground-truth label. Projected Gradient Descent (PGD) attack [19] is an iterative adversarial attack characterized by parameters, such as, maximum perturbation $\epsilon$, perturbation step size $\alpha$ and the number of iterations $n$. Higher values of $\epsilon$, $\alpha$ and $n$ denote stronger PGD attacks. In our experiments, we represent the configuration of PGD as [$\epsilon$, $\alpha$, $n$]. In Fig. 3, we compare the robustness of SNNs with rate coding and direct coding on two architectures (*i.e.*, MLP and VGG5). Note, for generating adversarial examples for SNNs with rate coding, we use the method proposed by [17]. Interestingly, we find that rate-coded SNNs have higher adversarial robustness approximately up to 20% higher accuracy compared to direct-coded SNNs. This is because the Poisson generator function is non-differentiable, therefore the gradient has to be approximated, making the attack ineffective.

### 3.6. Energy-efficiency of two coding schemes

To assess the energy costs incurred by the two coding schemes, we evaluate our MLP and VGG5 SNNs on a 16-bit *Eyeriss* platform (65 nm CMOS technology) using an *output stationary* dataflow. *Eyeriss* follows a von-Neumann mode of neural computation widely adopted in modern accelerators [18, 27]. It enables us to optimize over different design choices such as, type of dataflow, computation reuse and skipping zero computations. A representation of a processing element (PE) for the evaluation of SNNs on *Eyeriss* has been depicted in Fig. 4. Rate-encoded SNNs also involve a digital circuit for achieving the Poisson rate coding that is also accounted for in our energy evaluation [6]. Table 1 shows the normalized energy values per image for the SNNs, wherein we find that rate coding yields less energy cost than direct coding by $\sim 50\%$. Here, all the measured energy values are normalized with respect to a 16-bit Multiply-and-Accumulate (MAC) operation. Primarily, it is due to the fact that the first layer of the direct-encoded SNNs has 16-bit data precision and thus, incurs greater computational costs with respect to the rate-encoded SNNs having binary spike trains as inputs. Note, the first layer mainly contributes to the total energy cost as deep layers yield highly sparse spike activation. Moreover, in the standard *Eyeriss* architecture, direct coding requires the same full-precision inputs and weights for the first layer to be repeatedly fetched $T$ times to perform corresponding MAC operations (where, $T$ is total number of timesteps) that add significantly to the energy expenditure. On the other hand, rate-encoded SNNs with a sparse distribution of input spikes help reduce the computations and hence, energy expenditure on the *Eyeriss* platform. In order to circumvent the repetition of the full-precision first layer computations in direct-encoded SNNs, we need to modify the standard *Eyeriss* PE to facilitate digital shift operations for the partial sums generated during MAC. The modification will alleviate repeated fetches of the same inputs and weights from the spads (or registers) as well as the cost of multiply operation for multiple time-steps.

**Table 1**. Energy expended by rate-encoded and direct-encoded SNNs on an *Eyeriss* platform normalized with respect to the energy of 1 MAC operation. We use 10 timesteps for both architectures.

| | Normalized energy/image | |
|---|---|---|
| Scheme | MLP/MNIST | VGG5/CIFAR10 |
| Rate coding | 8.58E+06 | 2.73E+07 |
| Direct coding | 2.26E+07 | 4.22E+07 |

## 4. CONCLUSION AND DISCUSSION

This study is motivated by the question: *Which coding scheme is better for building accurate, robust, and energy-efficient SNN, rate coding or direct coding?* Note that we focus on rate coding and direct coding because they enable SNNs to be trained on large-scale datasets with deep architectures. This question is timely as neuromorphic researchers are finding a way of scaling up the SNN models like their ANN counterparts. To explore this, we conduct a comprehensive analysis on accuracy, robustness, and energy-efficiency. Direct coding trains the weights of the coding layer, therefore the input image can be converted into the optimal spikes with float amplitude for minimizing cross-entropy loss. Consequently, direct coding achieves better accuracy than rate coding where the input spikes are generated based on pixel intensity. Although the learnable layer in direct coding improves performance, such layer is vulnerable to adversarial attack as it conveys malicious backward gradients to the input image. On the other hand, Poisson spike generator of the rate coding is non-differentiable and stochastic, resulting in less performance degradation in case of adversarial attack. We also find that rate-encoded SNNs achieve higher energy-efficiency on *Eyeriss* architecture than their direct-encoded counterparts. Overall, our extensive experiments provide a fundamental understanding of the two coding schemes. We hope this enables researchers to utilize proper coding schemes according to their specific applications.

## 5. REFERENCES

[1] Kaushik Roy et al., "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[2] Dennis V Christensen et al., "2022 roadmap on neuromorphic computing and engineering," *Neuromorphic Computing and Engineering*, 2022.

[3] Yujie Wu et al., "Direct training for spiking neural networks: Faster, larger, better," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 1311–1318.

[4] Iulia M Comsa et al., "Temporal coding in spiking neural networks with alpha synaptic function," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8529–8533.

[5] Seongsik Park et al., "Fast and efficient information transmission with burst spikes in deep spiking neural networks," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[6] Wenzhe Guo et al., "Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems," *Frontiers in Neuroscience*, vol. 15, pp. 212, 2021.

[7] Hanle Zheng et al., "Going deeper with directly-trained larger spiking neural networks," *arXiv preprint arXiv:2011.05280*, 2020.

[8] Wenrui Zhang and Peng Li, "Temporal spike sequence learning via backpropagation for deep spiking neural networks," *arXiv preprint arXiv:2002.10085*, 2020.

[9] Wei Fang et al., "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," *arXiv preprint arXiv:2007.05785*, 2020.

[10] Chankyu Lee et al., "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in Neuroscience*, vol. 14, 2020.

[11] Peter U Diehl and Matthew Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 9, pp. 99, 2015.

[12] Jun Haeng Lee et al., "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, pp. 508, 2016.

[13] Youngeun Kim and Priyadarshini Panda, "Visual explanations from spiking neural networks using interspike intervals," *arXiv preprint arXiv:2103.14441*, 2021.

[14] Hesham Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 7, pp. 3227–3235, 2017.

[15] Nitin Rathi et al., "Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation," *arXiv preprint arXiv:2005.01807*, 2020.

[16] Ian J Goodfellow et al., "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[17] Saima Sharmin et al., "Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations," *arXiv preprint arXiv:2003.10399*, 2020.

[18] Yu-Hsin Chen et al., "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[19] Aleksander Madry et al., "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[20] Marcelo A Montemurro et al., "Phase-of-firing coding of natural visual stimuli in primary visual cortex," *Current biology*, vol. 18, no. 5, pp. 375–380, 2008.

[21] Youngeun Kim and Priyadarshini Panda, "Revisiting batch normalization for training low-latency deep spiking neural networks from scratch," *arXiv preprint arXiv:2010.01729*, 2020.

[22] Youngeun Kim and Priyadarshini Panda, "Optimizing deeper spiking neural networks for dynamic vision sensing," *Neural Networks*, vol. 144, pp. 686–698, 2021.

[23] Adam Paszke et al., "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[24] Yujie aand others Wu, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in neuroscience*, vol. 12, pp. 331, 2018.

[25] Yann LeCun et al., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," 2009.

[27] Surya Narayanan et al., "Spinalflow: an architecture and dataflow tailored for spiking neural networks," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 349–362.