Temporal Misalignment in ANN-SNN Conversion and its Mitigation via Probabilistic Spiking Neurons

Velibor Bojković [* 1] **Xiaofeng Wu** [* 2] **Bin Gu** [3]

Abstract

Spiking Neural Networks (SNNs) offer a more energy-efficient alternative to Artificial Neural Networks (ANNs) by mimicking biological neural principles, establishing them as a promising approach to mitigate the increasing energy demands of large-scale neural models. However, fully harnessing the capabilities of SNNs remains challenging due to their discrete signal processing and temporal dynamics. ANN-SNN conversion has emerged as a practical approach, enabling SNNs to achieve competitive performance on complex machine learning tasks. In this work, we identify a phenomenon in the ANN-SNN conversion framework, termed *temporal misalignment*, in which random spike rearrangement across SNN layers leads to performance improvements. Based on this observation, we introduce biologically plausible two-phase probabilistic (TPP) spiking neurons, further enhancing the conversion process. We demonstrate the advantages of our proposed method both theoretically and empirically through comprehensive experiments on CIFAR-10/100, CIFAR10-DVS, and ImageNet across a variety of architectures, achieving state-of-the-art results. Codes are available at https://github.com/shizukanaskytree/TPP-SNN.

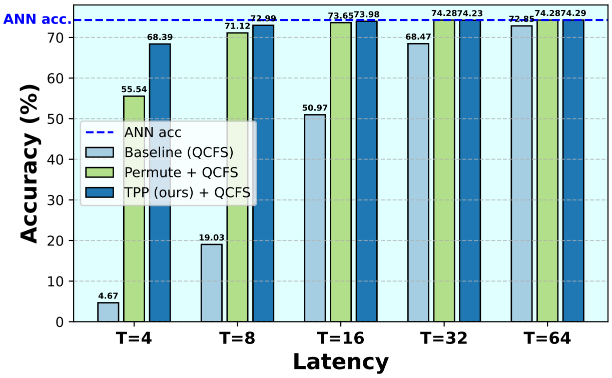


Figure 1. The initial experiment: After ANN-SNN conversion, we compared the accuracy of the baseline model QCFS (Bu et al., 2022c) with its "permuted" version and our proposed TPP neurons (setting is VGG-16 - ImageNet, ANN acc. 74.29%).

1. Introduction

Spiking neural networks (SNNs), often referred to as the third generation of neural networks (Maass, 1997), closely mimic biological neuronal communication through discrete spikes (McCulloch & Pitts, 1943; Hodgkin & Huxley, 1952; Izhikevich, 2003). While biological energy efficiency has historically influenced neural network design, SNNs uniquely achieve this through event-driven processing: weighted inputs integrate into membrane potentials, emitting binary spikes only upon crossing activation thresholds. This differs significantly from artificial neural networks (ANNs) (Braspenning et al., 1995), which are based on continuous floating-point operations that require energy-intensive and computationally costly multiplication operations (Roy et al., 2019). The spike-driven paradigm inherently circumvents these costly computations, suggesting that SNNs may offer a promising approach for energy-efficient AI. Recent developments in neuromorphic hardware (Pei et al., 2019; DeBole et al., 2019; loi; Ma et al., 2023) have enabled efficient SNN deployment. These specialized chips are inherently designed for spike-based computation, driving breakthroughs across domains: object detection (Kim et al., 2020b; Cheng et al., 2020), tracking (Yang et al., 2019), event-based vision (Zhu et al., 2022; Ren et al., 2024), speech recognition (Wang et al., 2023a), and generative AI through models like SpikingBERT (Bal & Sengupta, 2024) and SpikeGPT (Zhu et al., 2023; Wang et al., 2023b). These advancements underscore the potential of SNNs as a viable alternative to conventional ANNs.

Training SNNs is inherently challenging due to the same

[*] Equal contribution [1] Department of Machine Learning, Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates [2] Faculty of Data Science, City University of Macau, Macau, China [3] School of Artificial Intelligence, Jilin University, China. Correspondence to: Velibor Bojkovic <velibor.bojkovic@mbzuai.ac.ae>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

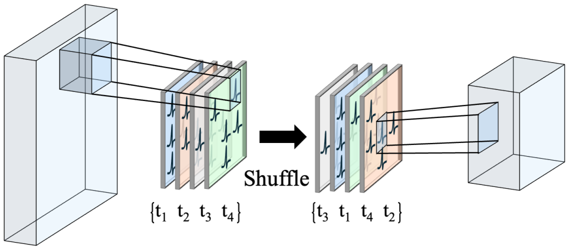


Figure 2. Spike train permutation: spikes at different time steps are shuffled to alter their temporal order.

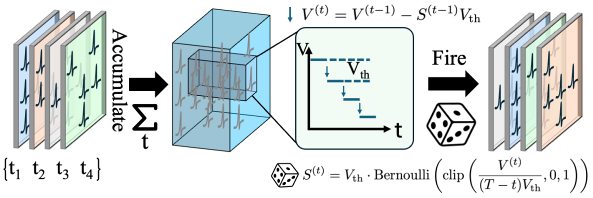

$V^{(t)} = V^{(t-1)} - S^{(t-1)}V_{\text{th}}$
Accumulate
$\sum_t$
V
V_{th}
t
Fire
$\{t_1\ t_2\ t_3\ t_4\}$
$S^{(t)} = V_{\text{th}} \cdot \text{Bernoulli}\left(\text{clip}\left(\frac{V^{(t)}}{(T-t)V_{\text{th}}}, 0, 1\right)\right)$

Figure 3. Two-Phase Probabilistic (TPP) spiking neuron mechanism operates in two phases: accumulation of inputs and probabilistic spiking based on membrane potential. The model uses a Bernoulli process for spike generation, with membrane potential updates over time steps.

characteristics that confer their advantages: their discrete processing of information. Unsupervised direct training, inspired by biological learning mechanisms, leverages local learning rules and spike timing to update weights (Diehl & Cook, 2015). While these methods are computationally efficient and can be executed on specialized hardware, SNNs trained in this manner often underperform compared to models trained with alternative approaches. Further research is needed to better understand and improve this method. In contrast, supervised training can be categorized into direct training via spatio-temporal backpropagation (e.g., surrogate gradient methods (Neftci et al., 2019)) and ANN-SNN conversion methods (Diehl et al., 2015; Cao et al., 2015). The present work focuses on the latter approach.

The core idea of ANN-SNN conversion is to leverage pretrained ANN models to train SNNs. This process begins by transferring the weights from the ANN to the SNN, which shares the same architecture, and initializing the spiking neuron parameters (e.g., thresholds, time constants) so that the spike rates approximate the activation values of the corresponding ANN layers. This method is advantageous because it typically requires no additional computation for training the SNN, thereby eliminating the need for gradient computation or limiting it to fine-tuning the SNN.

The initial experiment: Temporal information in SNNs after ANN-SNN conversion. We begin by identifying a counter-intuitive phenomenon that, to the best of our knowledge, has not been previously documented. Specifically, we investigate the extent to which individual spike timing affects the overall performance of the SNN model during ANN-SNN conversion. We explore this question in the context of various types of conversion-related issues described in the literature, such as *phase lag* (Li et al., 2022) and *unevenness of the spike input* (Bu et al., 2022c), where prior work has indicated that the timing of spike trains in SNNs obtained through ANN-SNN conversion may not be optimal and can lead to performance degradation under low-latency conditions.

In our initial experiment, we began by examining several widely-used ANN-SNN conversion methods and analyzing the spike outputs of spiking layers in the corresponding SNN models. To evaluate the importance of spike timing relative to firing rates, we randomly permuted the spike trains after each spiking layer. Specifically, when processing samples through the baseline model, we rearranged the temporal order of spikes within each spike train after each spiking layer (see Figure 2). The permuted spike trains were then passed to the subsequent layer, and this process was repeated for each layer until the output layer. The results of one of these initial experiments, comparing the performance of the "permuted" model with the baseline, are presented in Figure 1. For each latency, we conducted multiple trials with different permutations, and the "permuted" model consistently outperformed the baseline, achieving the original ANN accuracy at lower latencies. The impact of permutations on performance was particularly pronounced at lower latencies.

We refer to this phenomenon as "*temporal misalignment*" in ANN-SNN conversion and further investigate it by providing a conceptual interpretation in the form of bursting probabilistic spiking neurons, which are designed to mimic the effects of permutations in SNNs. The proposed neurons operate in two phases, as illustrated in Figure 3. In the first phase, they accumulate input, often surpassing the threshold, while in the second phase, they emit spikes probabilistically with varying temporal probabilities. Our proposed spiking neurons are characterized by two key properties: (1) the accumulation of membrane potential beyond the threshold, resulting in a firing phase, commonly referred to as bursting, and (2) probabilistic firing. Both properties exhibit biological plausibility and have been extensively studied in the neuroscience literature (see Section 3.3).

The main contributions of this paper are summarized as:

- We recognize and study the "temporal misalignment" phenomenon in ANN-SNN conversion, and we propose a framework for its exploitation in ANN-SNN conversion utilizing two-phase probabilistic spiking neurons. We provide theoretical insights into their functioning and superior performance, as well as support for their biological grounding.

- We present a comprehensive experimental validation demonstrating that our proposed method outperforms state-of-the-art conversion as well as the other training methods in terms of accuracy on CIFAR-10/100, CIFAR10-DVS, and ImageNet datasets.

2. Background and Related Work

The base model employed in this work is Integrate-and-Fire (IF) spiking neuron, whose internal dynamics, after discretization, are given by the equations

$$\begin{aligned}\mathbf{v}^{(l)}[t] &= \mathbf{v}^{(l)}[t-1] + \mathbf{W}^{(l)}\theta^{(l-1)} \cdot \mathbf{s}^{(l-1)}[t] - \theta^{(l)} \cdot \mathbf{s}[t-1], \\ \mathbf{s}^{(l)}[t] &= H(\mathbf{v}^{(l)}[t] - \theta^{(l)}).\end{aligned} \tag{1}$$

Here, $\theta^{(l)}$ is the threshold, $H(\cdot)$ is the Heaviside function, while the superscript l denotes the layer in the SNN. Later, we will modify these equations and use more advanced neuron models. By expanding the equations over $t = 1, \ldots, T$, and rearranging the terms, we obtain

$$\theta^{(l)} \frac{\sum_{t=1}^{T} \mathbf{s}^{(l)}[t]}{T} = \mathbf{W}^{(l)} V_{\text{th}}^{(l-1)} \frac{\sum_{t=1}^{T} \mathbf{s}^{(l-1)}[t]}{T} \tag{2}$$

$$+ \frac{\mathbf{v}^{(l)}[T] - \mathbf{v}^{(l)}[0]}{T}. \tag{3}$$

On the ANN side, the transformation between layers is given by

$$a^{(l)} = \mathcal{A}^{(l)}(\mathbf{W}^{(l)} a^{(l-1)}), \tag{4}$$

where $\mathcal{A}^{(l)}$ is the activation function. The ANN-SNN conversion process begins by transferring the weights (and biases) of a pre-trained ANN to an SNN with the same architecture. By comparing the equations for the ANN outputs (4) and the average output of the SNN (2) (Rueckauer et al., 2017a), the goal is to achieve a relation of the form

$$a_i^{(l)} \approx V_{\text{th}}^{(l)} \frac{\sum_{t=1}^{T} \mathbf{s}_i^{(l)}[t]}{T}. \tag{5}$$

The most commonly used activation function $\mathcal{A}$ is ReLU , due to its simplicity and non-negative output, which aligns well with the properties of IF neurons. It is important to note the importance of the three components in the conversion: 1) the threshold value θ, 2) the initialization $\mathbf{v}[0]$, 3) the ANN activation function $\mathcal{A}$.

2.1. Related work

ANN-SNN Conversion. This methodology aligns ANNs and SNNs through activation-firing rate correspondence, as initially demonstrated in (Rueckauer et al., 2017a; Cao et al., 2015). Subsequent research has systematically improved conversion fidelity through four principal approaches: (i) weight normalization (Diehl et al., 2015), (ii) soft-reset mechanisms (Rueckauer et al., 2017b; Han et al., 2020), (iii) adaptive threshold configurations (Stöckl & Maass, 2021; Ho & Chang, 2021; Wu et al., 2023), and (iv) spike coding optimization (Kim et al., 2020a; Sengupta et al., 2018). Recent innovations focus on ANN activation function adaptations, including thresholded ReLU (Ding et al., 2021) and quantization strategies (Bu et al., 2022c; Liu et al., 2022; Hu et al., 2023; Shen et al., 2024). However, these approaches introduce inherent accuracy-compression tradeoffs. Parallel efforts modify integrate-and-fire neuron dynamics (Li & Zeng, 2022; Wang et al., 2022a; Liu et al., 2022), with (Liu et al., 2022) proposing a dual-phase mechanism resembling our approach.

Crucial for achieving high conversion efficacy, threshold initialization methodologies employ layer-wise activation maxima or percentile statistics (Rueckauer et al., 2017a; Deng & Gu, 2021; Li et al., 2021a; Wu et al., 2024), augmented with post-conversion weight calibration (Li et al., 2021a; Bojković et al., 2024). Contemporary strategies can be categorized into two paradigms: (1) ANN activation quantization for temporal efficiency at the cost of accuracy, and (2) SNN neuron modification preserving ANN expressiveness with extended temporal requirements. Our methodology adheres to the second paradigm.

Direct Training. This approach leverages spatio-temporal spike patterns through backpropagation-through-time with differentiable gradient approximations (O'Connor et al., 2018; Zenke & Ganguli, 2018; Wu et al., 2018; Bellec et al., 2018; Fang et al., 2021a;b; Zenke & Vogels, 2021; Mukhoty et al., 2024). Advancements encompass joint optimization of synaptic weights and neuronal parameters (threshold dynamics (Wei et al., 2023), leakage factors (Rathi & Roy, 2023)), novel loss formulations for spike distribution regularization (Zhu et al., 2024; Guo et al., 2022), and hybrid conversion-training pipelines (Wang et al., 2022b). State-of-the-art developments introduce ternary spike representations for enhanced information density (Guo et al., 2024) and reversible architectures for memory-efficient training (Zhang & Zhang, 2024).

3. Methodology

3.1. Motivation

In ANN-SNN conversion methodologies, constant and rate coding are commonly utilized in the resulting spiking neural network models, based on the principle that the expected input at each time step matches the original input to the ANN model. Notably, the encoding lacks temporal information, as spike timing does not convey additional information. In constant encoding, this is evident, while in rate encoding, for a fixed input channel, the spike probability remains constant across all time steps, with the channel value assumed to lie between 0 and 1.

The resulting SNN model is initialized to approximate the outputs of the original ANN model based on the principle that, for each spiking neuron, the expected number of spikes it generates should closely match the output of the corresponding ANN neuron. In particular, it is assumed that no temporal information is present throughout the SNN model; that is, the spike train outputs of each SNN layer should convey no additional temporal information beyond spike firing rates.

Previous studies examining conversion errors and their classifications (Li et al., 2022; Bu et al., 2022c; Ding et al., 2021; Bojković et al., 2024) suggest that SNNs obtained through ANN-SNN conversion may generate spikes that are suboptimally positioned in the temporal domain, leading to a degradation in model performance, particularly at low latencies. Our initial experiments (see Introduction and Figure 1) further validate this observation while also revealing a novel insight: random temporal displacements of spike trains after spiking layers significantly enhance model performance. This phenomenon, which we refer to as temporal misalignment – wherein the original spike trains exhibit temporal misalignment, thereby impairing model performance – serves as the foundation and motivation for our proposed method, which is elaborated upon in the next section. Additional experiments on permutations in the ANN-SNN context, along with an explanation of their impact on model performance, are provided in Appendix G.

3.2. From permutations to Bursting Probabilistic Spiking Neurons

This work aims to address the following question: How to incorporate the action of permutation of the output spike trains into the dynamics of the spiking neurons? We approach this problem in two steps.

Consider the scenario where spike trains from layer ℓ are to be permuted. A general approach involves introducing a "permutator" – a subsequent layer tasked with collecting all incoming spikes and re-emitting them in a permuted manner, as illustrated in Figure 2. This inherently implies the **two-phase** nature of the "permutator": specifically, during the first phase, incoming spikes are accumulated, and firing is deferred until the onset of the second phase, where spikes are emitted.

The second step focuses on the output mechanism of the "permutator". Specifically, it is desirable to design a spiking neuron mechanism that retains the stochastic component of the permutations. This consideration motivates the adoption of probabilistic firing in spiking neurons.

The final question we explore is whether a more compact approach can be achieved by employing probabilistic spiking neurons that aggregate weighted inputs from the previous layer, rather than directly processing spikes from a spiking layer.

TPP neurons: To address the aforementioned questions, we propose **two-phase probabilistic spiking neurons** (TPP) (see Figure 3). Specifically, in the first phase, the neurons will only accumulate the (weighted) input coming from the previous layer, while in the second phase, the neurons will spike. More precisely, suppose that at a particular layer ℓ the spiking neurons accumulate the whole output of the previous layer, without emitting spikes. Denote the accumulated membrane potential by $\mathbf{v}^{(l)}[0]$. The subsequent spiking phase is governed by the following equations:

$$\begin{aligned} \mathbf{s}^{(l)}[t] &= B\left(\frac{1}{\theta^{(l)}\cdot(T-t+1)}\cdot\mathbf{v}^{(l)}[t-1]\right), \\ \mathbf{v}^{(l)}[t] &= \mathbf{v}^{(l)}[t-1]-\theta^{(l)}\cdot\mathbf{s}^{(l)}[t], \end{aligned} \tag{6}$$

where $t=1,\ldots,T$. Here, $B(x)$ is a Bernoulli random variable with bias x, extended for $x\in\mathbb{R}$ in a natural way ($B(x)=B(\max(\min(x,1),0))$). If the weights of the SNN network are not normalized, the produced spikes will be scaled with the thresholds $\theta^{(l)}\cdot\mathbf{s}^{(l)}[t]$, before being sent to the next layer.

We observe that the presence of $T-t+1$ in the denominator of the bias in B demonstrates that the probability of spiking depends not only on the current membrane potential, but also on the temporal step: in the absence of spiking, for the same membrane potential, the probability of spiking increases through time. Figure 3 provides a visual representation of the functioning of TPP neurons.

The following theorem provides a comprehensive characterization of the functioning of TPP neurons and their applications in ANN-SNN conversion when approximating ANN outputs (here $\mathrm{ReLU}_\theta(x)=\min(\mathrm{ReLU}(x),\theta)$).

Theorem 1. *Let $X^{(l)}$ be the input of the ANN layer with* ReLU *activation and suppose that, during the accumulation phase, the corresponding SNN layer of TPP neurons accumulated $T\cdot X^{(l)}$ quantity of voltage.*
(a) Suppose that for some $t=1,\ldots,T$, the TPP layer produced $s^{(l)}[1],\ldots,s^{(l)}[t-1]$ vector spike trains for the first $t-1$ steps, and the residue voltage for neuron i is higher than zero. Then,

$$\frac{(T-t+1)\cdot\theta^{(l)}}{T}\cdot\mathbb{E}\left[s_i^{(l)}[t]\right]+\frac{\theta^{(l)}}{T}\cdot\sum_{i=1}^{t-1}s_i^{(l)}[i] = \mathrm{ReLU}_{\theta^{(l)}}(X_i^{(l)}). \tag{7}$$

(b) If $s^{(l)}[1],\ldots,s^{(l)}[T]$ are the output vectors of spike trains of the TPP neurons during T time steps, then

$$\frac{\theta^{(l)}}{T}\cdot\sum_{i=1}^{T}s_j^{(l)}[i]=\mathrm{ReLU}_{\theta^{(l)}}(X_j^{(l)}), \tag{8}$$

if $\mathrm{ReLU}_{\theta^{(l)}}(X_j^{(l)})$ *is a multiple of* $\frac{\theta^{(l)}}{T}$, *or*

$$\frac{\theta^{(l)}}{T} \cdot \sum_{i=1}^{T} s_j^{(l)}[i] = \frac{\theta^{(l)}}{T} \cdot \lfloor \frac{T}{\theta^{(l)}} \mathrm{ReLU}_{\theta^{(l)}}(X_j^{(l)}) \rfloor \tag{9}$$

$$\text{or } \frac{\theta^{(l)}}{T} \cdot \lfloor \frac{T}{\theta^{(l)}} \mathrm{ReLU}_{\theta^{(l)}}(X_j^{(l)}) \rfloor + \frac{\theta^{(l)}}{T}, \tag{10}$$

if $\mathrm{ReLU}_{\theta^{(l)}}(X_j^{(l)})$ *is not a multiple of* $\frac{\theta^{(l)}}{T}$.

(c) *Suppose that* $\max X^{(l)} \leq \theta$ *and that the same weights* $W^{(l+1)}$ *act on the outputs of layer* (l) *of ANN and SNN as above, and let* $X^{(l+1)}$ *(resp.* $T \cdot \tilde{X}^{(l+1)}$*) be the inputs to the* $(l+1)$*th ANN layer (resp. the accumulated voltage for the* $(l+1)$*th SNN layer of TPP neurons), Then*

$$||X^{(l+1)} - \tilde{X}^{(l+1)}||_\infty \leq ||W^{(l+1)}||_\infty \cdot \frac{\theta^{(l)}}{T}. \tag{11}$$

• We contrast the statement (a) with Theorem 2 of (Bu et al., 2022c). Specifically, the authors demonstrate that if the membrane potential is initialized at half of the threshold, the expectation of the conversion error (layerwise) is 0. However, this result in (Bu et al., 2022c) relies on the underlying assumption that the layerwise distribution of ANN activation values is **uniform**, which does not generally hold in practice (see, for example, (Bojković et al., 2024)). Our result (a) above shows that **after every** $t \leq T$ **time steps**, our expected spiking rate aligns well with the clipping of the ReLU activation by the threshold, as it should, without imposing any prior assumptions on the distribution of the ANN activation values.

Moreoveor, our result demonstrates that the activity of a TPP neuron **adapts to the output** it already produced. In particular, as long as the neuron is still active and contains residual membrane potential, the expectation of its output at the next time step takes into account the previously produced spikes and will yield the ANN counterpart.

• The results (b) and (c) show that during the accumulation phase, the TPP neuron closely approximate ANN neurons with ReLU activation. In particular, the only remaining source of errors in layerwise approximation is the clipping error due to the threshold θ, and the quantization error due to the discrete outputs of the spiking neurons. We also note in equations (9) and (10) two possibilities for the output, which come from the probabilistic nature of spiking.

When it comes to the initial permutation effect on performance of the model, we connect with the *uniformization* of the output, as the next result suggests.

Theorem 2. *Suppose we have* N *spiking neurons that produced spike trains* $s_i[1], s_i[2], \ldots, s_i[T]$, $i = 1, \ldots, N$. *Furthermore, suppose that these spike trains are modulated with weights* $w_1, \ldots, w_N$, *respectively. For a given permutation* $\pi = (\pi_1, \ldots, \pi_N)$, *let* πs_i *denote the permutation of the spike train* s_i. *Then, for every* $t_1, t_2 \in \{1, 2, \ldots, T\}$,

$$E_\pi[\sum w_i \pi s_i[t_1]] = E_\pi[\sum w_i \pi s_i[t_2]].$$

Remarks: Theorem 1 is proved in the Appendix B while Theorem 2 is proved in the Appendix G.

3.3. Bio-plausibility and hardware implementation of TPP neurons

Our proposed neurons have two distinct properties: The two-phase regime and probabilistic spike firing. Both properties are biologically plausible and extensively studied in the neuroscience literature. For example, the two phase regime is related to firing after a delay of biological spiking neurons, where a neuron collects the input beyond the threshold value and fires after delay or after some condition is met. It could also be related to the bursting, when a biological neuron starts emitting bursts of spikes, after a certain condition is met, effectively dumping their accumulated potential (Izhikevich, 2007; Connors & Gutnick, 1990; Llinás & Jahnsen, 1982; Krahe & Gabbiani, 2004).

On the other side, stochastic firing of biological neurons has been well studied as well, and different aspects of noise introduction into firing have been proposed. We refer to (Shadlen & Newsome, 1994; Faisal et al., 2008; Softky & Koch, 1993; Maass & Natschläger, 1997; Pagliarini et al., 2019; Stein et al., 2005) for some examples.

Regarding the implementation of TPP neurons on neuromorphic hardware, two phase regime can be easily achieved on various modern neuromorphic chips that support programmable spiking neurons. Stochastic firing can be achieved through random sampling which, for example, is supported by IBM TrueNorth (Merolla et al., 2014), Intel Loihi (Davies et al., 2018), BrainScaleS-2 (Pehle et al., 2022), SpiNNaker (Furber et al., 2014) neuromorphic chips. For example, TrueNorth incorporates stochastic neuron models using on-chip pseudo-random number generators, enabling probabilistic firing patterns that mirror our approach. Similarly, Loihi (Gonzalez et al., 2024) supports stochastic operations by adding uniformly distributed pseudorandom noise to neuronal variables, facilitating the implementation of probabilistic spiking neurons.

To reduce the overall latency for processing inputs with our models, which yields linear dependence on the number of layers (implied by the two phase regime), we note that as soon as a particular layer has finished the firing phase, it can start receiving the input from the previous one: The process of classifying a dataset can be serialized. This has already been observed, for example in (Liu et al., 2022).

Neuromophic hardware implementation of this serialization has been proposed as well, see for example (Das, 2023; Song et al., 2021; Varshika et al., 2022).

4. Experiments

In this section, we verify the effectiveness and efficiency of our proposed methods. We compare it with state-of-the-art methods for image classification via converting ResNet-20, ResNet-34 (He et al., 2016), VGG-16 (Simonyan & Zisserman, 2015), RegNet (Radosavovic et al., 2020) on CIFAR-10 (LeCun et al., 1998; Krizhevsky et al., 2010), CIFAR-100 (Krizhevsky & Hinton, 2009), CIFAR10-DVS (Li et al., 2017) and ImageNet (Deng et al., 2009). Our experiments use PyTorch (Paszke et al., 2019), PyTorch vision models (maintainers & contributors, 2016), and the PyTorch Image Models (Timm) library (Wightman, 2019).

To demonstrate the wide applicability of the TPP neurons and the framework we propose, we combine them with three representative methods of ANN-SNN conversion from recent literature, each of which has their own particularities. These methods are: QCFS (Bu et al., 2022b), RTS (Deng & Gu, 2021), and SNNC (Li et al., 2021a). The particularity of QCFS method is that it uses step function instead of ReLU in ANN models during their training, in order to obtain higher accuracy in lower latency after the conversion. RTS method uses thresholded ReLU activation in ANN models during their training, so that the outliers are eliminated among the activation values, which helps to reduce the conversion error. Finally, SNNC uses standard ANN models with ReLU activation, and performs grid search on the activation values to find optimal initialization of the thresholds in the converted SNNs.

We initialize our SNNs following the standard ANN-SNN conversion process described in Section 3 (and detailed in A), starting with a pre-trained model given by the baseline, or with training an ANN model using default settings in QCFS (Bu et al., 2022b), RTS (Deng & Gu, 2021), and SNNC (Li et al., 2021a). ANN ReLU activations were replaced with layers of TPP neurons initialized properly. All experiments were conducted using NVIDIA RTX 4090 and Tesla A100 GPUs. For comprehensive details on all setups and configurations, see Appendix C.2.

4.1. Comparison with the State-of-the-art ANN-SNN Conversion methods

We evaluate our approach against previous state-of-the-art ANN-SNN conversion methods, including ReLU-Threshold-Shift (RTS) (Deng & Gu, 2021), SNN Calibration (SNNC-AP) (Li et al., 2021a), Quantization Clip-Floor-Shift activation function (QCFS) (Bu et al., 2022b), SNM (Wang et al., 2022a), Burst (Li & Zeng, 2022), OPI (Bu et al., 2022a), SRP (Hao et al., 2023a), DDI (Bojković et al., 2024) and FTBC (Wu et al., 2024).

ImageNet dataset: Table 1 compares the performance of our proposed methods with state-of-the-art ANN-SNN conversion methods on ImageNet. Our method outperforms the baselines across all simulation time steps for VGG-16, and RegNetX-4GF. For instance, on VGG-16 at $T = 32$, our method achieves 74.72% accuracy, surpassing other baselines even at $T = 128$. Moreover, at $T = 128$, our method nearly matches the original ANN performance with only a 0.12% drop in VGG-16 and a 0.14% drop in ResNet-34.

We see similar patterns in combining our methods with RTS and QCFS baselines, which use modified ReLU activations to reduce conversion errors. Table 1 shows these results. For instance, applying TPP with QCFS on ResNet-34 at $T = 16$ improves performance from 59.35% to 72.03%, a 12.68% increase. Similarly, for VGG-16 at $T = 16$, combining TPP with QCFS boosts performance from 50.97% to 73.98%, a 23.01% increase. Using TPP with RTS also shows significant improvements, such as a 12.82% increase for VGG-16 at $T = 16$. These results demonstrate the benefits of integrating TPP with other optimization approaches, solidifying its role as a comprehensive solution for ANN-SNN conversion challenges.

CIFAR dataset: We further evaluate the performance of our methods on CIFAR-100 dataset and present the results in Table 1. We observe similar patterns as with the ImageNet. When comparing our method with ANN-SNN conversion methods which use non-ReLU activations, e.g. QCFS and RTS, our method constantly outperforms RTS on ResNet-20 and VGG16. QCFS baseline suffers from necessity to train ANN models from scratch with custom activations, while our method is applicable to any ANN model with ReLU -like activation. Furthermore, custom activation functions sometimes sacrifice the ANN performance as can be seen from the corresponding ANN accuracies.

CIFAR10-DVS dataset: We evaluate our method on the event-based CIFAR10-DVS (Li et al., 2017) dataset, comparing it with state-of-the-art direct training and ANN-SNN conversion methods (Table 2). Our approach demonstrates superior performance, achieving 82.40% accuracy at just 8 timesteps and further improving to 83.20% at 64 timesteps. Notably, our method outperforms the direct training method Spikformer (Zhou et al., 2022) (80.90%) and the ANN-SNN conversion method AdaFire (Wang et al., 2025) (81.25%).

4.2. Comparison with other types of SNN training methods and models

We compare our approach with several state-of-the-art direct training and hybrid training methods as presented in Table 2. The comparison is founded on performance metrics

Table 1. Comparison between our method and the other ANN-SNN conversion methods on ImageNet and CIFAR-100. We provide the average accuracy and the associated standard deviation across 5 experiments.

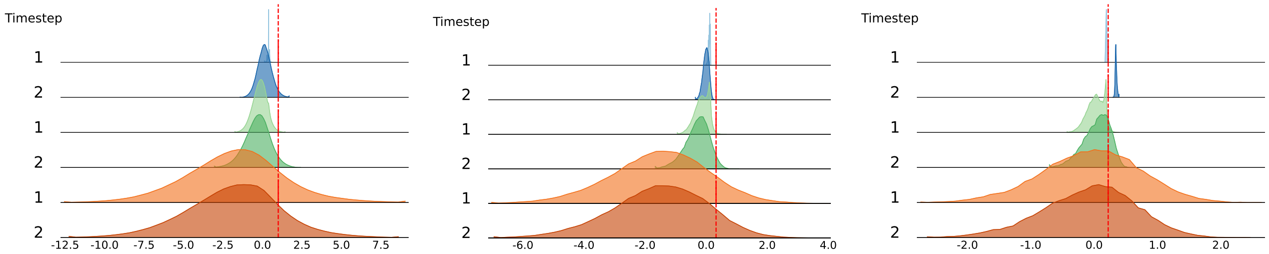
Dataset	Arch.	Method	ANN	T=4	T=8	T=16	T=32	T=64	T=128
ImageNet	ResNet-34	RTS (Deng & Gu, 2021)$^{\text{ICLR}}$	75.66	–	–	–	33.01	59.52	67.54
		SNNC-AP*(Li et al., 2021a)$^{\text{ICML}}$	75.66	–	–	–	64.54	71.12	73.45
		QCFS (Bu et al., 2022b)$^{\text{ICLR}}$	74.32	–	–	59.35	69.37	72.35	73.15
		SRP (Hao et al., 2023a)$^{\text{AAAI}}$	74.32	66.71	67.62	68.02	68.40	68.61	–
		FTBC(+QCFS) (Wu et al., 2024)$^{\text{ECCV}}$	74.32	49.94	65.28	71.66	73.57	74.07	74.23
		Ours (TPP) + QCFS	74.32	37.23 (0.07)	67.32 (0.06)	72.03 (0.02)	72.97 (0.03)	73.24 (0.02)	73.30 (0.02)
		Ours (TPP)*+ SNNC w/o Cali.	75.65	2.69 (0.03)	49.24 (0.23)	69.97 (0.10)	74.07 (0.06)	75.23 (0.03)	75.51 (0.05)
	VGG-16	SNNC-AP*(Li et al., 2021a)$^{\text{ICML}}$	75.36	–	–	–	63.64	70.69	73.32
		SNM*(Wang et al., 2022a)$^{\text{IJCAI}}$	73.18	–	–	–	64.78	71.50	72.86
		RTS (Deng & Gu, 2021)$^{\text{ICLR}}$	72.16	–	–	55.80	67.73	70.97	71.89
		QCFS (Bu et al., 2022b)$^{\text{ICLR}}$	74.29	–	–	50.97	68.47	72.85	73.97
		Burst (Li & Zeng, 2022)$^{\text{IJCAI}}$	74.27	–	–	–	70.61	73.32	73.00
		OPI*(Bu et al., 2022a)$^{\text{AAAI}}$	74.85	–	6.25	36.02	64.70	72.47	74.24
		SRP (Hao et al., 2023a)$^{\text{AAAI}}$	74.29	66.47	68.37	69.13	69.35	69.43	–
		FTBC(+QCFS) (Wu et al., 2024)$^{\text{ECCV}}$	73.91	58.83	69.31	72.98	74.05	74.16	74.21
		Ours (TPP) + RTS	72.16	30.50 (1.19)	56.69(0.67)	67.34 (0.25)	70.63 (0.11)	71.75 (0.05)	72.05 (0.03)
		Ours (TPP) + QCFS	74.22	68.39 (0.08)	72.99 (0.05)	73.98 (0.07)	74.23 (0.03)	74.29 (0.00)	74.33 (0.01)
		Ours (TPP)*+ SNNC w/o Cali.	75.37	54.14 (0.59)	69.75 (0.27)	73.44 (0.02)	74.72 (0.06)	75.14 (0.02)	75.25 (0.03)
	RegNetX-4GF	RTS (Deng & Gu, 2021)$^{\text{ICLR}}$	80.02	–	–	–	0.218	3.542	48.60
		SNNC-AP*(Li et al., 2021a)$^{\text{ICML}}$	80.02	–	–	–	55.70	70.96	75.78
		Ours (TPP)*+ SNNC w/o Cali.	78.45	–	–	22.71 (2.98)	66.51 (0.44)	75.54 (0.07)	77.83 (0.04)
CIFAR-100	ResNet-20	TSC*(Han & Roy, 2020)$^{\text{ECCV}}$	68.72	–	–	–	–	–	58.42
		RMP*(Han et al., 2020)$^{\text{CVPR}}$	68.72	–	–	–	27.64	46.91	57.69
		SNNC-AP*(Li et al., 2021a)$^{\text{ICML}}$	77.16	–	–	76.32	77.29	77.73	77.63
		RTS (Deng & Gu, 2021)$^{\text{ICLR}}$	67.08	–	–	63.73	68.40	69.27	69.49
		OPI*(Bu et al., 2022a)$^{\text{AAAI}}$	70.43	–	23.09	52.34	67.18	69.96	70.51
		QCFS^{+}(Bu et al., 2022b)$^{\text{ICLR}}$	67.09	27.87	49.53	63.61	67.04	67.87	67.86
		Burst*(Li & Zeng, 2022)$^{\text{IJCAI}}$	80.69	–	–	–	76.39	79.83	80.52
		Ours (TPP) + QCFS	67.10	46.88 (0.40)	64.77 (0.20)	67.25 (0.12)	67.74 (0.06)	67.77 (0.05)	67.79 (0.04)
		Ours (TPP)*+ SNNC w/o Cali.	81.89	39.67 (0.99)	71.05 (0.68)	78.97 (0.24)	81.06 (0.05)	81.61 (0.08)	81.62 (0.05)
	VGG-16	TSC*(Han & Roy, 2020)$^{\text{ECCV}}$	71.22	–	–	–	–	–	69.86
		SNM*(Wang et al., 2022a)$^{\text{ICLR}}$	74.13	–	–	–	71.80	73.69	73.95
		SNNC-AP*(Li et al., 2021a)$^{\text{ICML}}$	77.89	–	–	–	73.55	77.10	77.86
		RTS$^{\circ}$(Deng & Gu, 2021)$^{\text{ICLR}}$	76.13	23.76	43.81	56.23	67.61	73.45	75.23
		OPI*(Bu et al., 2022a)$^{\text{AAAI}}$	76.31	–	60.49	70.72	74.82	75.97	76.25
		QCFS^{+}(Bu et al., 2022b)$^{\text{ICLR}}$	76.21	69.29	73.89	75.98	76.53	76.54	76.60
		DDI (Bojković et al., 2024)$^{\text{AISTATS}}$	70.44	51.21	53.65	57.12	61.61	70.44	73.82
		FTBC(+QCFS) (Wu et al., 2024)$^{\text{ECCV}}$	76.21	71.47	75.12	76.22	76.48	76.48	76.48
		Ours (TPP) + RTS	76.13	37.88 (0.35)	65.81 (0.27)	73.05 (0.12)	75.17 (0.17)	75.64 (0.12)	75.9 (0.08)
		Ours (TPP) + QCFS	76.21	73.93 (0.22)	76.03 (0.23)	76.43 (0.07)	76.55 (0.03)	76.55 (0.07)	76.52 (0.04)
		Ours (TPP)*+ SNNC w/o Cali.	77.87	59.23 (0.65)	73.16 (0.17)	76.05 (0.26)	77.16 (0.09)	77.56 (0.13)	77.64 (0.04)

*: Without modification to ReLU of ANNs. +: Using authors' provided models and code. ○: Self implemented.

like accuracy and the number of timesteps utilized during inference on the CIFAR-100 and ImageNet datasets. We benchmark our method against prominent approaches such as LM-H (Hao et al., 2023b), SEENN (Li et al., 2023), Dual-Phase (Wang et al., 2022b), TTS (Guo et al., 2024), RMP-Loss (Guo et al., 2023), RecDis-SNN (Guo et al., 2022), SpikeConv (Liu et al., 2022), and GAC-SNN (Qiu et al., 2024). We showcase the best accuracy comparable to state-of-the-art methods achieved by our approach with minimal timesteps. We prioritize accuracy, but direct training and hybrid training opt for a lower number of timesteps and sacrifice accuracy. We outperform LM-H (Hao et al., 2023b) and Dual-Phase (Wang et al., 2022b) for VGG-16 on CIFAR-100. For ResNet-20 on CIFAR-100, we have higher accuracy but longer timesteps. Additionally, for ResNet-34 on the ImageNet dataset, the accuracy of our method with QCFS with 16 timesteps is higher than that of SpikeConv (Liu et al., 2022) with the same number of timesteps. We also achieve higher accuracy with longer timesteps as expected. Overall, our approach demonstrates promising performance and competitiveness in comparison with the existing SNN training methods.

4.3. Spiking activity

The event driven nature of various neuromorphic chips implies that the energy consumption is directly proportional to the spiking activity, i.e., the number of spikes produced throughout the network: the energy is consumed in the presence of spikes. To this end, we tested our proposed method (TPP) for the spike activity and compared with the baselines. For a given model, we counted the average number of spikes produced after each layer, per sample, for both



(a) Layer 4 (b) Layer 8 (c) Layer 12

Figure 4. The membrane potential distributions of the first channel (randomly selected) across three modes (baseline, shuffle, and probabilistic) in VGG-16 on CIFAR-100. For comparative analysis, the first two timesteps ($t = 1$, $t = 2$) from a total of eight timesteps ($T = 8$) are selected for each mode. The baseline mode (blue) attains an accuracy of 24.22%, while the shuffle mode (light green) enhances accuracy to 70.54%, and the probabilistic mode (dark orange) further improves accuracy to 73.42%. The distributions are presented prior to neuronal firing, with the red dashed line indicating the threshold voltage (Vth) for the respective layer (see Appendix F).

Table 2. Comparison of direct and hybrid training methods for SNNs on CIFAR-100, ImageNet and CIFAR10-DVS. Baseline results include the highest reported accuracy and corresponding latency.

Method	Category	Timesteps	Accuracy
VGG-16 [CIFAR-100]			
LM-H (Hao et al., 2023b)^ICLR	Hybrid Training	4	73.11
SEENN-II *(Li et al., 2023)^NeurIPS	Direct Training	1.15*	72.76
Dual-Phase (Wang et al., 2022b)	Hybrid Training	4 / 8	70.08 / 75.06
Ours (TPP) + QCFS	ANN-SNN	4 / 8	73.93 / 76.03
ResNet-20 [CIFAR-100]			
LM-H (Hao et al., 2023b)^ICLR	Hybrid Training	4	57.12
TTS (Guo et al., 2024)^AAAI	Direct Training	4	74.02
Ours (TPP) + SNNC w/o Cali.	ANN-SNN	16	78.97
ResNet-34 [ImageNet]			
SEENN-I (Li et al., 2023)^NeurIPS	Direct Training	3.38 *	64.66
RMP-Loss (Guo et al., 2023)^ICCV	Direct Training	4	65.17
RecDis-SNN (Guo et al., 2022)^CVPR	Direct Training	6	67.33
SpikeConv (Liu et al., 2022)^AAAI	Hybrid Training	16	70.57
GAC-SNN (Qiu et al., 2024)^AAAI	Direct Training	6	70.42
TTS (Guo et al., 2024)^AAAI	Direct Training	4	70.74
SEENN-I (Li et al., 2023)^NeurIPS	ANN-SNN	29.53 *	71.84
Ours (TPP) + QCFS	ANN-SNN	16	72.03
Ours (TPP)+ SNNC w/o Cali.	ANN-SNN	32	74.07
ResNet-18 [CIFAR10-DVS]			
TA-SNN (Yao et al., 2021)^ICCV	Direct Training	10	72.00
PLIF (Fang et al., 2021c)^ICCV	Direct Training	20	74.80
Dspkie (Li et al., 2021b)^NeurIPS	Direct Training	10	75.40
DSR (Meng et al., 2022)^CVPR	Direct Training	10	77.30
Spikformer (Zhou et al., 2022)^ICLR	Direct Training	10	80.90
AdaFire (Wang et al., 2025)^AAAI	ANN-SNN	8	81.25
Ours (TPP) + SNNC w/o Cali.	ANN-SNN	4	77.90
		8	82.40
		16	82.80
		32	83.00
		64	83.20

* The average number of timesteps during inference on the test dataset.

the baseline and our method. Figure 5 shows the example of RTS and RTS + TPP. Both the baseline and our method exhibit similar spike counts. In particular, our method constantly outperforms the baselines, and possibly in doing so it needs longer average latency per sample (T). However, the energy consumed is approximately the same as that for the baseline in time T. The complete tables are present in Appendix E.4, where we provide more detailed picture of spike activities.

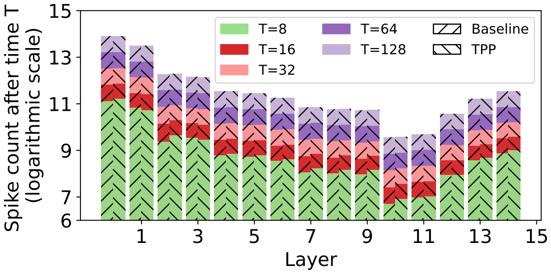


Figure 5. Spike counts of VGG-16 on CIFAR-100 of RTS baseline compared with RTS+TPP. Note: The bar height from bottom indicates the spike counts after each timestep T, and the color of longer Ts is overlaid by shorter Ts (see Appendix E.4)

4.4. Membrane potential distribution in early time steps

In Figure 4 we compare the membrane potential distributions for baseline models and two methods that we studied in the paper, the permutations applied on spike trains and TPP method. Once again, it can be seen another reason for performance degradation of baseline models in low latency, as the membrane potential is not variable enough to produce spike informative spike trains, which is particularly visible in deeper layers. On the other side "permuted" and TPP models show sufficient variability throughout the layers.

By increasing the latency, the baseline models can recover some of the variability and spike production, as can be seen in Figure 5. But, due to the misplacement of spike trains through temporal dimension, they are still not able to pick up on the ANN performance.

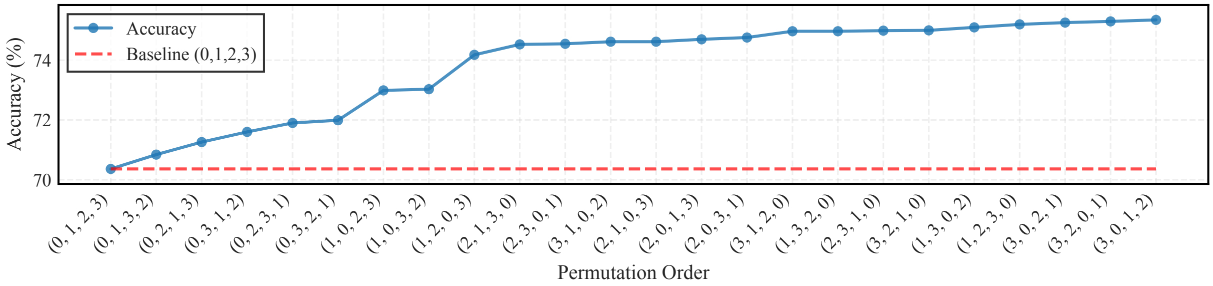


Figure 6. Accuracy comparison for all $T!$ permutations of input order over $T = 4$ time steps using QCFS with VGG-16 on CIFAR-100. Results of permuted orders outperform the original, non-permuted order $(0, 1, 2, 3)$. Baseline accuracy is 69.31%, The ANN accuracy is 76.21%.

4.5. Energy Efficiency

Following prior work such as (Rathi & Roy, 2023; Yao et al., 2023), we computed the energy cost for both ANNs and SNNs using 45nm CMOS technology. The energy consumption of a 32-bit MAC operation in ANNs (4.6 pJ) is approximately 5.1× higher than that of an addition operation in SNNs (0.9 pJ) (Horowitz, 2014). The results are reported in Table 3. We refer to Appendix E.5 for details.

Table 3. Energy consumption and Top-1 accuracy evaluated on ImageNet with a default input resolution of 3x224x224. The baseline model is used for comparison, and the results for the proposed method (TPP) are highlighted in bold.

Arch	Method	T	Acc.(%)	Energy(mJ)
VGG-16	QCFS	32	68.47	61.67
	TPP(ours)	4	68.39	**7.98**
	QCFS	64	72.85	123.02
	TPP(ours)	8	72.99	**15.66**
ResNet-34	SNNC	32	63.64	13.27
	TPP(ours)	8	69.75	**5.69**
	SNNC	64	70.69	23.51
	TPP(ours)	16	73.44	**12.16**

4.6. Effect of Temporal Reordering

To validate the effectiveness of temporal permutations on SNN performance, we exhaustively evaluate all $T! = 24$ possible orderings of the spike trains over $T = 4$ time steps using the QCFS with VGG-16 on CIFAR-100. As shown in Figure 6, all 23 permuted spike orders outperform the default chronological order $(0, 1, 2, 3)$ without any reordering. The accuracy of the default-order SNN is **70.36%**, while the ANN reference achieves **76.21%**.

The best-performing permutation $(3, 0, 1, 2)$ achieves **75.35%**—only 0.86% below the ANN accuracy, demonstrating that reordering consistently and significantly improves accuracy.

We also refer to the Appendix G for further experiments concerning the effect of permutations in ANN-SNN conversion.

5. Conclusions and future work

This work identified the phenomenon of “temporal misalignment” in ANN-SNN conversion, where random spike rearrangement enhances performance. We introduced two-phase probabilistic (TPP) spiking neurons, designed to intrinsically perform the effect of spike permutations. We show biological plausibility of such neurons as well as the hardware friendlines of the underlying mechanisms. We demonstrate their effectiveness through exhaustive experiments on large scale datasets, showing their competing performance compared to SOTA ANN-SNN conversion and direct training methods.

In the future work, we aim to study the effect of permutations and probabilistic spiking in combination with directly trained SNN models.

Acknowledgements

We extend our sincere gratitude to the anonymous reviewers for their valuable comments on this paper. The author X. Wu was supported by the Science and Technology Development Fund (FDCT) of Macau Special Administrative Region (0055/2023/ITP2, 0071/2024/ITP2).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Taking Neuromorphic Computing to the Next Level with Loihi 2. https://download.intel.com/newsroom/2021/new-technologies/neuromorphic-computing-loihi-2-brief.pdf. Accessed: 16-05-2023.

Bal, M. and Sengupta, A. Spikingbert: Distilling bert to train spiking language models using implicit differentiation. In *Proceedings of the AAAI conference on artificial intelligence*, 2024.

Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., and Maass, W. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in neural information processing systems*, 2018.

Bojković, V., Anumasa, S., De Masi, G., Gu, B., and Xiong, H. Data driven threshold and potential initialization for spiking neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2024.

Braspenning, P. J., Thuijsman, F., and Weijters, A. J. M. M. *Artificial neural networks: an introduction to ANN theory and practice*, volume 931. Springer Science & Business Media, 1995.

Bu, T., Ding, J., Yu, Z., and Huang, T. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022a.

Bu, T., Fang, W., Ding, J., Dai, P., Yu, Z., and Huang, T. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022b.

Bu, T., Fang, W., Ding, J., Dai, P., Yu, Z., and Huang, T. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2022c.

Cao, Y., Chen, Y., and Khosla, D. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1): 54–66, 2015.

Cheng, X., Hao, Y., Xu, J., and Xu, B. Lisnn: Improving spiking neural networks with lateral interactions for robust object recognition. In *IJCAI*, pp. 1519–1525, 2020.

Connors, B. W. and Gutnick, M. J. Intrinsic firing patterns of diverse neocortical neurons. *Trends in neurosciences*, 13(3):99–104, 1990.

Das, A. A design flow for scheduling spiking deep convolutional neural networks on heterogeneous neuromorphic system-on-chip. *ACM Transactions on Embedded Computing Systems*, 2023.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

DeBole, M. V., Taba, B., Amir, A., Akopyan, F., Andreopoulos, A., Risk, W. P., Kusnitz, J., Otero, C. O., Nayak, T. K., Appuswamy, R., et al. Truenorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 52(5):20–29, 2019.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Deng, S. and Gu, S. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Diehl, P. U. and Cook, M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., and Pfeiffer, M. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. ieee, 2015.

Ding, J., Yu, Z., Tian, Y., and Huang, T. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. *In International Joint Conference on Artificial Intelligence*, pp. 2328–2336, 2021.

Faisal, A. A., Selen, L. P., and Wolpert, D. M. Noise in the nervous system. *Nature reviews neuroscience*, 9(4): 292–303, 2008.

Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., and Tian, Y. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021a.

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., and Tian, Y. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2021b.

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., and Tian, Y. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2661–2671, 2021c.

Furber, S. B., Galluppi, F., Temple, S., and Plana, L. A. The spinnaker project. *Proceedings of the IEEE*, 102(5): 652–665, 2014.

Gonzalez, H. A., Huang, J., Kelber, F., Nazeer, K. K., Langer, T., Liu, C., Lohrmann, M., Rostami, A., Schöne, M., Vogginger, B., et al. Spinnaker2: A large-scale neuromorphic system for event-based and asynchronous machine learning. *arXiv preprint arXiv:2401.04491*, 2024.

Guo, Y., Tong, X., Chen, Y., Zhang, L., Liu, X., Ma, Z., and Huang, X. Recdis-snn: Rectifying membrane potential distribution for directly training spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Guo, Y., Liu, X., Chen, Y., Zhang, L., Peng, W., Zhang, Y., Huang, X., and Ma, Z. Rmp-loss: Regularizing membrane potential distribution for spiking neural networks. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, 2023.

Guo, Y., Chen, Y., Liu, X., Peng, W., Zhang, Y., Huang, X., and Ma, Z. Ternary spike: Learning ternary spikes for spiking neural networks. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, 2024.

Han, B. and Roy, K. Deep spiking neural network: Energy efficiency through time based coding. In *European Conference on Computer Vision*. Springer, 2020.

Han, B., Srinivasan, G., and Roy, K. RMP-SNN: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13558–13567, 2020.

Hao, Z., Bu, T., Ding, J., Huang, T., and Yu, Z. Reducing ANN-SNN conversion error through residual membrane potential. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, 2023a.

Hao, Z., Shi, X., Huang, Z., Bu, T., Yu, Z., and Huang, T. A progressive training framework for spiking neural networks with learnable multi-hierarchical model. In *The Twelfth International Conference on Learning Representations*, 2023b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Ho, N.-D. and Chang, I.-J. Tcl: an ann-to-snn conversion with trainable clipping layers. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021.

Hodgkin, A. L. and Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117 (4):500, 1952.

Horowitz, M. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pp. 10–14. IEEE, 2014.

Hu, Y., Zheng, Q., Jiang, X., and Pan, G. Fast-snn: fast spiking neural network by converting quantized ann. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Izhikevich, E. M. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

Izhikevich, E. M. *Dynamical systems in neuroscience*. MIT press, 2007.

Kim, J., Kim, K., and Kim, J. Unifying activation- and timing-based learning rules for spiking neural networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a.

Kim, S., Park, S., Na, B., and Yoon, S. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 11270–11277, 2020b.

Krahe, R. and Gabbiani, F. Burst firing in sensory systems. *Nature Reviews Neuroscience*, 5(1):13–23, 2004.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *https://www.cs.toronto.edu/ kriz/cifar.html*, 2009.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 2010.

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

Li, H., Liu, H., Ji, X., Li, G., and Shi, L. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017.

Li, Y. and Zeng, Y. Efficient and accurate conversion of spiking neural network with burst spikes. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 2022.

Li, Y., Deng, S., Dong, X., Gong, R., and Gu, S. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, pp. 6316–6325. PMLR, 2021a.

Li, Y., Guo, Y., Zhang, S., Deng, S., Hai, Y., and Gu, S. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34:23426–23439, 2021b.

Li, Y., Zhao, D., and Zeng, Y. Bsnn: Towards faster and better conversion of artificial neural networks to spiking neural networks with bistable neurons. *Frontiers in Neuroscience*, 16, 2022. ISSN 1662-453X. doi: 10.3389/fnins.2022.991851. URL https://www.frontiersin.org/articles/10.3389/fnins.2022.991851.

Li, Y., Geller, T., Kim, Y., and Panda, P. SEENN: towards temporal spiking early exit neural networks. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Liu, F., Zhao, W., Chen, Y., Wang, Z., and Jiang, L. Spike-converter: An efficient conversion framework zipping the gap between artificial neural networks and spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 1692–1701, 2022.

Llinás, R. and Jahnsen, H. Electrophysiology of mammalian thalamic neurones in vitro. *Nature*, 297(5865):406–408, 1982.

Ma, D., Jin, X., Sun, S., Li, Y., Wu, X., Hu, Y., Yang, F., Tang, H., Zhu, X., Lin, P., and Pan, G. Darwin3: A large-scale neuromorphic chip with a novel ISA and on-chip learning. *CoRR*, 2023.

Maass, W. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(97)00011-7. URL https://www.sciencedirect.com/science/article/pii/S0893608097000117.

Maass, W. and Natschläger, T. Networks of spiking neurons can emulate arbitrary hopfield nets in temporal coding. *Network: Computation in Neural Systems*, 8(4):355–371, 1997.

maintainers, T. and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016.

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

Meng, Q., Xiao, M., Yan, S., Wang, Y., Lin, Z., and Luo, Z.-Q. Training High-Performance Low-Latency Spiking Neural Networks by Differentiation on Spike Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12444–12453, 2022.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C., Nakamura, Y., et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

Mukhoty, B., Bojković, V., de Vazelhes, W., Zhao, X., De Masi, G., Xiong, H., and Gu, B. Direct training of snn using local zeroth order method. *Advances in Neural Information Processing Systems*, 36, 2024.

Neftci, E. O., Mostafa, H., and Zenke, F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

O'Connor, P., Gavves, E., Reisser, M., and Welling, M. Temporally efficient deep learning with spikes. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

Pagliarini, S. N., Bhuin, S., Isgenc, M. M., Biswas, A. K., and Pileggi, L. A probabilistic synapse with strained mtjs for spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(4):1113–1123, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Pehle, C., Billaudelle, S., Cramer, B., Kaiser, J., Schreiber, K., Stradmann, Y., Weis, J., Leibfried, A., Müller, E., and Schemmel, J. The brainscales-2 accelerated neuromorphic system with hybrid plasticity. *Frontiers in Neuroscience*, 16:795876, 2022.

Pei, J., Deng, L., Song, S., Zhao, M., Zhang, Y., Wu, S., Wang, G., Zou, Z., Wu, Z., He, W., et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

Qiu, X., Zhu, R., Chou, Y., Wang, Z., Deng, L., and Li, G. Gated attention coding for training high-performance and efficient spiking neural networks. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, 2024.

Radosavovic, I., Kosaraju, R. P., Girshick, R. B., He, K., and Dollár, P. Designing network design spaces. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, 2020.

Rathi, N. and Roy, K. DIET-SNN: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Trans. Neural Networks Learn. Syst.*, 2023.

Ren, H., Zhou, Y., Huang, Y., Fu, H., Lin, X., Song, J., and Cheng, B. Spikepoint: An efficient point-based spiking neural network for event cameras action recognition. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Austria, May 6-11, 2024*, 2024.

Roy, K., Jaiswal, A., and Panda, P. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11, 2017a. ISSN 1662-453X. doi: 10.3389/fnins.2017.00682. URL https://www.frontiersin.org/articles/10.3389/fnins.2017.00682.

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017b.

Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroence*, 2018.

Shadlen, M. N. and Newsome, W. T. Noise, neural codes and cortical organization. *Current opinion in neurobiology*, 4 (4):569–579, 1994.

Shen, G., Zhao, D., Li, T., Li, J., and Zeng, Y. Are conventional snns really efficient? a perspective from network quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27538–27547, 2024.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition, 2015.

Softky, W. R. and Koch, C. The highly irregular firing of cortical cells is inconsistent with temporal integration of random epsps. *Journal of neuroscience*, 13(1):334–350, 1993.

Song, S., Varshika, M. L., Das, A., and Kandasamy, N. A design flow for mapping spiking neural networks to many-core neuromorphic hardware. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9. IEEE, 2021.

Stein, R. B., Gossen, E. R., and Jones, K. E. Neuronal variability: noise or part of the signal? *Nature Reviews Neuroscience*, 6(5):389–397, 2005.

Stöckl, C. and Maass, W. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nature Machine Intelligence*, 3(3): 230–238, 2021.

Varshika, M. L., Balaji, A., Corradi, F., Das, A., Stuijt, J., and Catthoor, F. Design of many-core big little μbrains for energy-efficient embedded neuromorphic computing. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1011–1016. IEEE, 2022.

Wang, Q., Zhang, T., Han, M., Wang, Y., Zhang, D., and Xu, B. Complex dynamic neurons improved spiking transformer network for efficient automatic speech recognition. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, 2023a.

Wang, Y., Zhang, M., Chen, Y., and Qu, H. Signed neuron with memory: Towards simple, accurate and high-efficient ann-snn conversion. In Raedt, L. D. (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 2501–2508. International Joint Conferences on Artificial Intelligence Organization, 7 2022a. doi: 10.24963/ijcai.2022/347. URL https://doi.org/10.24963/ijcai.2022/347. Main Track.

Wang, Z., Lian, S., Zhang, Y., Cui, X., Yan, R., and Tang, H. Towards lossless ANN-SNN conversion under ultra-low latency with dual-phase optimization. *CoRR*, 2022b.

Wang, Z., Fang, Y., Cao, J., Zhang, Q., Wang, Z., and Xu, R. Masked spiking transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023b.

Wang, Z., Fang, Y., Cao, J., Ren, H., and Xu, R. Adaptive calibration: A unified conversion framework of spiking neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

Wei, W., Zhang, M., Qu, H., Belatreche, A., Zhang, J., and Chen, H. Temporal-coded spiking neural networks with dynamic firing threshold: Learning with event-driven backpropagation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Wightman, R. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Wu, J., Chua, Y., Zhang, M., Li, G., Li, H., and Tan, K. C. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 2023.

Wu, X., Bojković, V., Gu, B., Suo, K., and Zou, K. FTBC: Forward temporal bias correction for optimizing ann-snn conversion. *ECCV*, 2024.

Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12: 331, 2018.

Yang, Z., Wu, Y., Wang, G., Yang, Y., Li, G., Deng, L., Zhu, J., and Shi, L. Dashnet: a hybrid artificial and spiking neural network for high-speed object tracking. *arXiv preprint arXiv:1909.12942*, 2019.

Yao, M., Gao, H., Zhao, G., Wang, D., Lin, Y., Yang, Z., and Li, G. Temporal-wise attention spiking neural networks for event streams classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10221–10230, 2021.

Yao, M., Hu, J., Zhou, Z., Yuan, L., Tian, Y., Xu, B., and Li, G. Spike-driven transformer. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Zenke, F. and Ganguli, S. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 2018.

Zenke, F. and Vogels, T. P. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, 33(4): 899–925, 2021.

Zhang, H. and Zhang, Y. Memory-efficient reversible spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, 2024.

Zhou, Z., Zhu, Y., He, C., Wang, Y., Yan, S., Tian, Y., and Yuan, L. Spikformer: When Spiking Neural Network Meets Transformer. *arXiv preprint arXiv:2209.15425*, 2022.

Zhu, L., Wang, X., Chang, Y., Li, J., Huang, T., and Tian, Y. Event-based video reconstruction via potential-assisted spiking neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3594–3604, 2022.

Zhu, R., Zhao, Q., and Eshraghian, J. K. Spikegpt: Generative pre-trained language model with spiking neural networks. *CoRR*, 2023.

Zhu, Y., Fang, W., Xie, X., Huang, T., and Yu, Z. Exploring loss functions for time-based training strategy in spiking neural networks. In *Advances in Neural Information Processing Systems*, 2024.

A. Conversion steps

Copying ANN architecture and weights. ANN-SNN conversion process starts with a pre-trained ANN model, whose weights (and biases) will be copied to an SNN model following the same architecture. In this process, one considers ANN models whose non-activation layers become linear during the inference. In particular, these include fully connected, convolutional, batch normalization and average pooling layers.

Approximating ANN activation functions. The second step of the process considers the activation layers and their activation functions in ANN. Here, the idea is to initialize the spiking neurons in the corresponding SNN layer in such a way that their average spiking rate approximates the values of the corresponding activation functions. For the ReLU (or ReLU -like such as quantized or thresholded ReLU) activations, this process is rather well understood. The spiking neuron threshold is usually set to correspond to the maximum activation ANN channel or layerwise, or to be some percentile of it. If we denote by f the ANN actiavtion, then ideally, after setting the thresholds, one would like to have

$$f(\mathbf{v}[T]) \approx \frac{\theta}{T} \cdot \sum_{t=1}^{T} \mathbf{s}[t]. \tag{12}$$

If we recall the equations for the IF neuron (equations (1) in the article)

$$\mathbf{v}^{(l)}[t] = \mathbf{v}^{(l)}[t-1] + \mathbf{W}^{(l)}\theta^{(l-1)} \cdot \mathbf{s}^{(l-1)}[t] - \theta^{(l)} \cdot \mathbf{s}[t-1], \tag{13}$$

$$\mathbf{s}^{(l)}[t] = H(\mathbf{v}^{(l)}[t] - \theta^{(l)}), \tag{14}$$

we see that the value with which we are comparing the membrane potential (threshold) is the same as the value with which we are scaling the output spikes. In particular, as soon as our membrane potential has reached θ, it will produce the value θ. This can be loosely described as, whatever the input is, the output will be approximately that value (or zero, if the input is negative), which is exactly what ReLU does.

Absorbing thresholds. Finally, we notice that, once we produce a spike $\mathbf{s}^{(l)}[t]$, the value $\theta^{(l)} \cdot \mathbf{s}^{(l)}[t]$ will be sent to the next layer, and will further be weighted with weights $W^{(l+1)}$ and the bias $b^{(l+1)}$ will be applied. As we want SNNs to operate only using ones and zeros (to avoid multiplication due to energy efficiency), the values $\theta^{(l)}$ will be absorbed into $W^{(l+1)}$, i.e. $W^{(l+1)} \leftarrow \theta^{(l)} W^{(l+1)}$.

B. Proof of Theorem 1

We prove the main theorem from the article, which we restate here.

Theorem 1. *Let $X^{(l)}$ be the input of the ANN layer with* ReLU *activation and suppose that, during the accumulation phase, the corresponding SNN layer of TPP neurons accumulated $T \cdot X^{(l)}$ quantity of voltage.*
(a) Suppose that for some $t = 1, \ldots, T$, the TPP layer produced $s^{(l)}[1], \ldots, s^{(l)}[t-1]$ vector spike trains for the first $t-1$ steps, and the residue voltage for neuron i is higher than zero. Then,

$$\begin{aligned} &\frac{(T-t+1)\cdot\theta^{(l)}}{T} \cdot \mathbb{E}\left[s_i^{(l)}[t]\right] + \frac{\theta^{(l)}}{T} \cdot \sum_{i=1}^{t-1} s_i^{(l)}[i] \\ &= \mathrm{ReLU}_{\theta^{(l)}}(X_i^{(l)}). \end{aligned} \tag{7}$$

(b) If $s^{(l)}[1], \ldots, s^{(l)}[T]$ are the output vectors of spike trains of the TPP neurons during T time steps, then

$$\frac{\theta^{(l)}}{T} \cdot \sum_{i=1}^{T} s_j^{(l)}[i] = \mathrm{ReLU}_{\theta^{(l)}}(X_j^{(l)}), \tag{8}$$

if $\mathrm{ReLU}_{\theta^{(l)}}(X_j^{(l)})$ *is a multiple of* $\frac{\theta^{(l)}}{T}$, *or*

$$\frac{\theta^{(l)}}{T} \cdot \sum_{i=1}^{T} s_j^{(l)}[i] = \frac{\theta^{(l)}}{T} \cdot \lfloor \frac{T}{\theta^{(l)}} \mathrm{ReLU}_{\theta^{(l)}}(X_j^{(l)}) \rfloor \tag{9}$$

$$\textit{or } \frac{\theta^{(l)}}{T} \cdot \lfloor \frac{T}{\theta^{(l)}} \mathrm{ReLU}_{\theta^{(l)}}(X_j^{(l)}) \rfloor + \frac{\theta^{(l)}}{T}, \tag{10}$$

if $\mathrm{ReLU}_{\theta^{(l)}}(X_j^{(l)})$ *is not a multiple of* $\frac{\theta^{(l)}}{T}$.

(c) *Suppose that* $\max X^{(l)} \le \theta$ *and that the same weights* $W^{(l+1)}$ *act on the outputs of layer* (l) *of ANN and SNN as above, and let* $X^{(l+1)}$ *(resp.* $T \cdot \tilde{X}^{(l+1)}$*) be the inputs to the* $(l+1)$*th ANN layer (resp. the accumulated voltage for the* $(l+1)$*th SNN layer of TPP neurons), Then*

$$||X^{(l+1)} - \tilde{X}^{(l+1)}||_\infty \le ||W^{(l+1)}||_\infty \cdot \frac{\theta^{(l)}}{T}. \tag{11}$$

Proof. We start by rewriting equation (6) as

$$\begin{aligned}
\mathbf{s}^{(l)}[t] &= B\left(\frac{1}{\theta^{(l)} \cdot (T-t+1)} \cdot \mathbf{v}^{(l)}[t-1]\right) && (15)\\
&= B\left(\frac{1}{\theta^{(l)} \cdot (T-t+1)} \cdot \left(T \cdot X^{(l)} - \theta^{(l)} \cdot \sum_{i=1}^{t-1} s^{(l)}[i]\right)\right) && (16)\\
&= B\left(\frac{1}{T-t+1} \cdot \left(T \cdot \frac{X^{(l)}}{\theta^{(l)}} - \sum_{i=1}^{t-1} s^{(l)}[i]\right)\right), && (17)
\end{aligned}$$

obtained by unrolling through time the expression for the membrane potential.

We next consider various settings of the theorem. For the first three statements, we can assume that the vector $X^{(l)}$ is one dimensional, i.e. $X^{(l)} \in \mathbb{R}$. We start by first considering the situation where $X^{(l)} > \theta^{(l)}$, so that the output of the ANN is $\mathrm{ReLU}_{\theta^{(l)}}(X^{(l)}) = \theta^{(l)}$. In that case, we notice that the spiking neuron will fire at every time step $t = 1, \ldots, T$, because the bias for the Bernoulli random variable will always be bigger than or equal to 1, as follows from the previous equations. Similarly, if $X^{(l)} \le 0$, the bias will always be non-positive, and both the output of ANN and of SNN will be 0. Consequently, the first three statements follow directly for both of these cases.

Suppose now that $0 \le X^{(l)} < \theta^{(l)}$ (or, equivalently, $0 \le \frac{X^{(l)}}{\theta^{(l)}} < 1$) and let a be minimal non-negative integer such that $a \cdot \theta^{(l)} \le T \cdot X^{(l)} < (a+1) \cdot \theta$, i.e. $a := \lfloor \frac{T \cdot X^{(l)}}{\theta^{(l)}} \rfloor$. In particular, $0 \le a < T$. We proceed to prove statement (b) above. We note that in general, $\sum_{i=1}^{t-1} s^{(l)}[i] \le a+1$, because after at most $a+1$ spikes, the residue membrane potential would become negative. Also, due to the condition of the statement that the residue membrane potential is still non-negative, we may assume that $\sum_{i=1}^{t-1} s^{(l)}[i] \le a$. Then, it becomes clear that the bias in the last equation (17) is a non-negative number smaller than 1, and it follows that

$$\mathbb{E}\left[s_i^{(l)}[t]\right] = \frac{1}{T-t+1} \cdot \left(T \cdot \frac{X^{(l)}}{\theta^{(l)}} - \sum_{i=1}^{t-1} s^{(l)}[i]\right), \tag{18}$$

so that we finally have

$$\begin{aligned}
&\frac{\theta^{(l)} \cdot (T-t+1)}{T} \cdot \mathbb{E}\left[s_i^{(l)}[t]\right] + \frac{\theta^{(l)}}{T} \cdot \sum_{i=1}^{t-1} s_i^{(l)}[i] && (19)\\
&\quad = \frac{\theta^{(l)} \cdot (T-t+1)}{T} \cdot \frac{1}{T-t+1} \cdot \left(T \cdot \frac{X^{(l)}}{\theta^{(l)}} - \sum_{i=1}^{t-1} s^{(l)}[i]\right) + \frac{\theta^{(l)}}{T} \cdot \sum_{i=1}^{t-1} s_i^{(l)}[i] && (20)\\
&\quad = X^{(l)} = \mathrm{ReLU}_{\theta^{(l)}}(X^{(l)}). && (21)
\end{aligned}$$

Statement (a) follows from (b), while for the statement (c) for the first case, we notice that after every spike, the residue membrane potential is a multiple of $\theta^{(l)}$, hence we will have exactly a spikes (notation above), while the second case follows from statement (b). Finally, statement (d) is a direct consequence of the above discussion. □

C. Experiments Details

C.1. Datasets

CIFAR-10: The CIFAR-10 dataset (Krizhevsky et al., 2010) contains 60,000 color images of 32x32 pixels each, divided into 10 distinct classes (e.g., airplanes, cars, birds), with each class containing 6,000 images. The dataset is split into 50,000 training images and 10,000 test images.

CIFAR-100: The CIFAR-100 dataset (Krizhevsky et al., 2010) consists of 60,000 color images of 32x32 pixels, distributed across 100 classes, with each class having 600 images. Similar to CIFAR-10, it is divided into 50,000 training images and 10,000 test images.

CIFAR10-DVS: The CIFAR10-DVS (Li et al., 2017) dataset is a neuromorphic vision benchmark derived from the CIFAR-10 dataset, converting 10,000 static images (1,000 per class across 10 categories like airplanes and cars) into event streams using a Dynamic Vision Sensor (DVS). Generated via Repeated Closed-Loop Smooth (RCLS) movement to simulate realistic motion, the dataset captures spatio-temporal intensity changes as asynchronous ON/OFF events (128×128 resolution) with inherent noise (e.g., 60Hz LCD artifacts, later corrected). Designed for moderate complexity between MNIST-DVS and N-Caltech101, it challenges event-driven algorithms, achieving initial low accuracies (22–29%) with methods like spiking neural networks and SVM, highlighting its utility for advancing neuromorphic research. Publicly available on Figshare[1], CIFAR10-DVS bridges traditional and event-based vision, fostering innovation in brain-inspired computing and real-world dynamic scene analysis.

ImageNet: The ImageNet dataset (Deng et al., 2009) comprises 1,281,167 images spanning 1,000 classes in the training set, with a validation set and a test set containing 50,000 and 100,000 images, respectively. Unlike the CIFAR datasets, ImageNet images vary in size and resolution. The validation set is frequently used as the test set in various applications.

C.2. Configuration and Setups

C.2.1. OURS + QCFS

CIFAR: We followed the original paper's training configurations to train ResNet-20 and VGG-16 on CIFAR-100. The Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9 was used. The initial learning rate was set to 0.02, with a weight decay of 5×10^{-4}. A cosine decay scheduler adjusted the learning rate over 300 training epochs. The quantization steps L were set to 8 for ResNet-20 and 4 for VGG-16. All models were trained for 300 epochs.

ImageNet: We utilized checkpoints for ResNet-34 and VGG-16 from the original paper's GitHub repository. For ImageNet, L was set to 8 and 16 for ResNet-34 and VGG-16, respectively.

C.2.2. OURS + RTS

CIFAR: We trained models using the recommended settings from the original paper.

ImageNet: We used pre-trained checkpoints for ResNet-34 and VGG-16 from the original paper's GitHub repository. Subsequently, all ReLU layers were replaced with spiking neuron layers.

For all datasets, we initialize TPP membrane potential to zero, while in the baselines we do as they propose.

C.2.3. OURS + SNNC W/O CALIBRATION

CIFAR: We adhered to the original paper's configurations to train ResNet-20 and VGG-16 on CIFAR-100. The SGD optimizer with a momentum of 0.9 was used. The initial learning rate was set to 0.01, with a weight decay of 5×10^{-4} for models with batch normalization. A cosine decay scheduler adjusted the learning rate over 300 training epochs. All models were trained for 300 epochs with a batch size of 128.

ImageNet: We used pre-trained checkpoints for ResNet-34 and VGG-16 from the original paper's GitHub repository. Subsequently, all ReLU layers were replaced with our proposed spiking neuron layers.

[1] https://figshare.com/articles/dataset/CIFAR10-DVS_New/4724671

D. Algorithms

The baseline SNN neuron forward function (Algorithm 1) initializes the membrane potential to zero and iteratively updates it by adding the layer output at each timestep. Spikes are generated when the membrane potential exceeds a defined threshold, θ, and the potential is reset accordingly. This function captures the core dynamics of spiking neurons. The Shuffle Mode (Algorithm 2) is an extension of the baseline forward function. After generating the spikes across the simulation length, this mode shuffles the spike train.

The TPP Mode (Algorithm 3) introduces a probabilistic component to the spike generation process. Instead of a deterministic threshold-based spike generation, it uses a Bernoulli process where the probability of spiking is determined by the current membrane potential relative to the threshold adjusted for the remaining timesteps.

Algorithm 1 SNN Neuron Forward Function and Additional Modes

```
Require: SNN Layer $\ell$; Input tensor $\mathbf{x}$; Threshold $\theta$; Simulation length $T$.
function BaselineSNN($\ell, \mathbf{x}, \theta, T$)
    $\mathbf{v} \leftarrow 0$ {Initialize membrane potential}
    for $t = 1$ to $T$ do
        $\mathbf{v} \leftarrow \mathbf{v} + \ell(\mathbf{x}(t))$
        $\mathbf{s} \leftarrow (\mathbf{v} \geq \theta) \times \theta$
        $\mathbf{v} \leftarrow \mathbf{v} - \mathbf{s}$
        Store $\mathbf{s}(t)$
    end for
    return $\mathbf{s}$
end function
```

Algorithm 2 SNN Neuron Forward Function of Shuffle Mode

```
Require: SNN Layer $\ell$; Input tensor $\mathbf{x}$; Threshold $\theta$; Simulation length $T$.
function ShuffleMode($\ell, \mathbf{x}, \theta, T$)
    $\mathbf{v} \leftarrow 0$ {Initialize membrane potential}
    for $t = 1$ to $T$ do
        $\mathbf{v} \leftarrow \mathbf{v} + \ell(\mathbf{x}(t))$
        $\mathbf{s} \leftarrow (\mathbf{v} \geq \theta) \times \theta$
        $\mathbf{v} \leftarrow \mathbf{v} - \mathbf{s}$
        Store $\mathbf{s}(t)$
    end for
    Shuffle the stored spikes $\mathbf{s}(1), \mathbf{s}(2), \ldots, \mathbf{s}(T)$
    return shuffled $\mathbf{s}$
end function
```

Algorithm 3 SNN Neuron Forward Function of TPP Mode

```
Require: SNN Layer $\ell$; Input tensor $\mathbf{x}$; Threshold $\theta$; Simulation length $T$.
function TPPMode($\ell, \mathbf{x}, \theta, T$)
    $\mathbf{v} \leftarrow \sum_{t=1}^{T} \mathbf{x}(t)$ {Initialize membrane potential with the sum of inputs}
    for $t = 1$ to $T$ do
        $\mathbf{p} \leftarrow \text{Clamp}(\mathbf{v}/(\theta \times (T - t + 1)), 0, 1)$
        $\mathbf{s} \leftarrow \text{Bernoulli}(\mathbf{p}) \times \theta$
        $\mathbf{v} \leftarrow \mathbf{v} - \mathbf{s}$
        Store $\mathbf{s}(t)$
    end for
    return $\mathbf{s}$
end function
```

E. Additional Experiments

E.1. SNNC

We show extra experiment results about the comparison among permutation method and two-phase probabilistic method. We validated ResNet-20 and VGG-16 on the CIFAR-10/100 dataset , and ResNet-34, VGG-16 and RegNetX-4GF on ImageNet with batch and channel-wise normalization enabled. Using a batch size of 128, the experiment was run five times with different random seeds to ensure reliable and reproducible results.

Table 4. Comparison between our proposed methods and ANN-SNN conversion SNNC method on **CIFAR-10**. The average accuracy and standard deviation of the TPP method are reported over 5 experiments.

Architecture	Method	ANN	T=1	T=2	T=4	T=8	T=16	T=32	T=64
ResNet-20	SNNC-AP (Li et al., 2021a)	96.95	51.20	66.07	83.60	92.79	95.62	96.58	96.85
	Ours (Permute)	96.95	34.05	61.46	90.54	95.05	96.12	96.62	96.77
	Ours (TPP)	96.95	10.05 (0.02)	17.30 (0.52)	79.19 (0.67)	93.72 (0.05)	95.87 (0.09)	96.67 (0.04)	96.80 (0.01)
VGG-16	SNNC-AP (Li et al., 2021a)	95.69	60.72	75.82	82.18	91.93	93.27	94.97	95.40
	Ours (Permute)	95.69	38.01	64.40	84.65	92.24	92.80	93.33	94.10
	Ours (TPP)	95.69	11.46 (0.35)	32.24 (1.40)	86.85 (0.42)	94.34 (0.12)	94.86 (0.06)	95.48 (0.03)	95.60 (0.04)

Table 5. Comparison between our proposed methods and ANN-SNN conversion SNNC method on **CIFAR-100**. The average accuracy and standard deviation of the TPP method are reported over 5 experiments.

Architecture	Method	ANN	T=1	T=2	T=4	T=8	T=16	T=32	T=64
ResNet-20	SNNC-AP (Li et al., 2021a)	81.89	17.91	34.08	54.78	72.28	78.57	81.20	81.95
	Ours (Permute)	81.89	5.64	19.54	52.46	75.21	79.76	81.12	81.52
	Ours (TPP)	81.89	1.94 (0.11)	5.15 (0.44)	39.67 (0.99)	71.05 (0.68)	78.97 (0.24)	81.06 (0.05)	81.61 (0.08)
VGG-16	SNNC-AP (Li et al., 2021a)	77.87	28.64	34.87	50.95	64.30	71.93	75.39	77.05
	Ours (Permute)	77.87	12.50	34.98	60.81	69.42	72.78	73.50	75.14
	Ours (TPP)	77.87	2.05 (0.27)	15.90 (0.71)	59.23 (0.65)	73.16 (0.17)	76.05 (0.26)	77.16 (0.09)	77.56 (0.13)

Table 6. Comparison between our proposed methods and ANN-SNN conversion SNNC method on ImageNet. The average accuracy and standard deviation of the TPP method are reported over 5 experiments.

Architecture	Method	ANN	T=4	T=8	T=16	T=32	T=64	T=128
ResNet-34	SNNC-AP (Li et al., 2021a)	75.65	–	–	–	64.54	71.12	73.45
	Ours (Permute)	75.65	10.51	57.57	70.94	74.00	75.06	75.47
	Ours (TPP)	75.65	2.69 (0.03)	49.24 (0.23)	69.97 (0.10)	74.07 (0.06)	75.23 (0.03)	75.51 (0.05)
VGG-16	SNNC-AP (Li et al., 2021a)	75.37	–	–	–	63.64	70.69	73.32
	Ours (Permute)	75.37	38.61	67.29	73.35	74.34	74.82	75.11
	Ours (TPP)	75.37	54.14 (0.59)	69.75 (0.27)	73.44 (0.02)	74.72 (0.06)	75.14 (0.02)	75.25 (0.03)
RegNetX-4GF	SNNC-AP (Li et al., 2021a)	80.02	–	–	–	55.70	70.96	75.78
	Ours (Permute)	78.45	–	–	43.45	68.12	75.63	77.63
	Ours (TPP)	78.45	–	–	22.71 (2.98)	66.51 (0.44)	75.54 (0.07)	77.83 (0.04)

E.2. RTS

Table 7. Comparison between our proposed methods and ANN-SNN conversion RTS method on CIFAR-10/100 and ImageNet. The average accuracy and standard deviation of the TPP method are reported over 5 experiments.

Dataset	Architecture	Method	ANN	T=4	T=8	T=16	T=32	T=64	T=128
CIFAR-10	VGG-16	RTS*(Deng & Gu, 2021)	94.99	88.64	91.67	93.64	94.50	94.76	94.91
		Ours (Permute)	94.99	91.22	93.70	94.50	94.86	94.88	94.97
		Ours (TPP)	94.99	91.49 (0.21)	94.11 (0.09)	94.72 (0.08)	94.84 (0.06)	94.91 (0.02)	94.98 (0.02)
	ResNet-20	RTS*(Deng & Gu, 2021)	91.07	27.08	40.88	65.13	84.75	90.12	90.76
		Ours (Permute)	91.07	68.18	86.57	90.20	90.81	91.04	90.99
		Ours (TPP)	91.07	72.87 (0.22)	88.27 (0.14)	90.44 (0.08)	90.86 (0.14)	90.94 (0.04)	91.01 (0.03)
CIFAR-100	VGG-16	RTS$^{\circ}$(Deng & Gu, 2021)	76.13	23.76	43.81	56.23	67.61	73.45	75.23
		Ours (Permute)	76.13	35.31	62.84	71.20	74.34	75.53	75.92
		Ours (TPP) + RTS	76.13	37.88 (0.35)	65.81 (0.27)	73.05 (0.12)	75.17 (0.17)	75.64 (0.12)	75.90 (0.08)
ImageNet	VGG-16	RTS (Deng & Gu, 2021)	72.16	–	–	55.80	67.73	70.97	71.89
		Ours (Permute)	72.16	33.77	58.31	67.80	70.89	71.65	71.95
		Ours (TPP)	72.16	30.50 (1.19)	56.69(0.67)	67.34 (0.25)	70.63 (0.11)	71.75 (0.05)	72.05 (0.03)

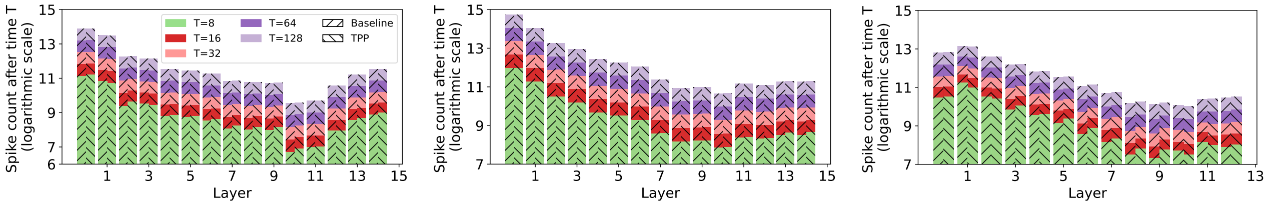
E.3. QCFS

Table 8. Comparison between our proposed methods and ANN-SNN conversion QCFS method on CIFAR-10/100 and ImageNet. The average accuracy and standard deviation of the TPP method are reported over 5 experiments.

Dataset	Architecture	Method	ANN	T=4	T=8	T=16	T=32	T=64
CIFAR-10	VGG-16	QCFS*(Bu et al., 2022c)	95.76	94.33	95.21	95.65	95.87	95.99
		Ours (Permute)	95.76	95.15	95.58	95.83	95.95	95.97
		Ours (TPP)	95.76	95.28(0.09)	95.84(0.1)	95.95(0.05)	95.98(0.06)	95.97 (0.03)
	ResNet-20	QCFS (Bu et al., 2022c)	92.43	79.45	88.56	91.94	92.79	92.82
		Ours (Permute)	92.43	84.85	91.24	92.67	92.82	92.85
		Ours (TPP)	92.43	86.24(0.18)	92.08(0.11)	92.70(0.1)	92.78(0.04	92.68(0.06)
CIFAR-100	VGG-16	QCFS$^{\circ}$(Bu et al., 2022c)	76.3	69.29	73.89	75.98	76.52	76.54
		Ours (Permute)	76.3	74.28	75.97	76.54	76.60	76.64
		Ours (TPP)	76.3	74.0(0.15)	76.06(0.08)	76.37(0.1)	76.55(0.09)	76.51(0.07)
	ResNet-20	QCFS (Bu et al., 2022c)	67.0	27.44	49.35	63.12	66.84	67.77
		Ours (Permute)	67.0	45.33	62.81	66.93	67.85	67.96
		Ours (TPP)	67.0	47.0(0.2)	64.66(0.25)	67.28(0.12)	67.61(0.1)	67.77(0.06)
ImageNet	VGG-16	QCFS (Bu et al., 2022c)	74.29	–	–	50.97	68.47	72.85
		Ours (Permute)	73.89	55.54	71.12	73.65	74.28	74.28
		Ours (TPP)	74.22	68.39 (0.08)	72.99 (0.05)	73.98 (0.07)	74.23 (0.03)	74.29 (0.00)

E.4. Spiking activity

The percentage difference between the baseline and our method in TPP mode is calculated as follows: Percentage Difference $= \frac{\text{Ours}-\text{Baseline}}{\text{Baseline}} \times 100$.



(a) RTS

(b) QCFS

(c) SNNC

Figure 7. Spike counts of VGG-16 on CIFAR-100 after different timesteps (T). Note: The bar height from bottom indicates the spike counts after each timestep T, and the color of longer Ts is overlaid by shorter Ts.

Table 9. Comparison of firing counts percentage difference between the baseline and our proposed TPP method for VGG-16 on CIFAR-100 using QCFS.

Layer	T=4	T=8	T=16	T=32	T=64	T=128
1	1.073	0.528	0.261	0.136	0.065	0.033
2	2.629	1.022	0.438	0.206	0.102	0.050
3	0.049	0.230	0.185	0.109	0.056	0.028
4	-0.867	-0.664	-0.419	-0.228	-0.118	-0.060
5	0.073	0.515	0.350	0.182	0.090	0.044
6	0.701	0.010	-0.098	-0.074	-0.041	-0.021
7	-1.071	-0.865	-0.470	-0.246	-0.122	-0.063
8	1.009	1.193	0.731	0.385	0.196	0.096
9	0.504	0.417	0.205	0.108	0.051	0.024
10	-0.112	0.842	0.647	0.375	0.198	0.100
11	2.071	2.438	1.614	0.898	0.465	0.235
12	0.797	0.943	0.756	0.461	0.247	0.127
13	4.503	2.156	1.209	0.655	0.343	0.171
14	25.898	13.883	7.770	3.852	1.887	0.936
15	33.585	16.864	8.945	4.474	2.227	1.108

Table 10. Comparison of firing counts percentage difference between the baseline and our proposed TPP method for ResNet-34 on ImageNet using QCFS.

Layer	T=4	T=8	T=16	T=32	T=64	T=128
1	0.587	0.306	0.149	0.079	0.036	0.018
2	-0.921	-0.435	-0.212	-0.108	-0.053	-0.025
3	0.353	0.189	0.082	0.036	0.019	0.010
4	-2.786	-1.583	-0.920	-0.506	-0.270	-0.141
5	0.469	0.277	-0.107	-0.020	-0.019	-0.011
6	-3.955	-1.865	-0.705	-0.344	-0.166	-0.086
7	-0.381	0.321	-0.090	-0.031	-0.020	-0.013
8	6.615	3.261	1.494	0.628	0.290	0.131
9	-5.116	-3.006	-1.555	-0.794	-0.391	-0.195
10	-2.938	3.431	3.096	1.794	0.975	0.498
11	1.184	0.466	0.359	0.102	0.053	0.022
12	-17.739	-7.302	-1.788	-0.609	-0.270	-0.132
13	0.105	-0.138	-0.287	-0.292	-0.166	-0.087
14	-8.597	-2.626	0.006	0.327	0.289	0.140
15	-0.522	-0.214	-0.273	-0.299	-0.173	-0.094
16	-11.196	-5.194	-1.990	-0.813	-0.405	-0.217
17	-3.828	-1.192	-0.320	-0.192	-0.105	-0.058
18	-6.869	-2.392	-0.644	0.007	-0.002	0.001
19	0.092	-0.299	-0.181	-0.138	-0.074	-0.035
20	-5.639	-0.308	0.923	0.796	0.448	0.234
21	0.399	-0.968	-0.796	-0.509	-0.275	-0.145
22	-4.474	3.712	4.440	3.033	1.700	0.880
23	0.456	-0.901	-0.703	-0.533	-0.281	-0.145
24	-5.863	4.241	5.617	3.797	2.090	1.074
25	1.433	-0.464	-0.774	-0.632	-0.347	-0.182
26	-5.034	4.908	6.328	4.362	2.459	1.271
27	0.661	-0.914	-1.156	-0.931	-0.530	-0.284
28	-15.667	4.763	9.616	6.975	4.062	2.096
29	-9.747	1.663	3.836	2.455	1.384	0.673
30	-0.151	16.639	15.387	9.638	5.334	2.769
31	-5.403	0.917	1.957	1.555	1.009	0.574
32	17.796	6.777	3.728	3.231	2.507	1.583
33	-4.935	-2.141	2.055	2.931	2.395	1.561

Table 11. Comparison of firing counts percentage difference between the baseline and our proposed TPP method for VGG-16 on ImageNet using QCFS.

Layer	T=4	T=8	T=16	T=32	T=64	T=128
1	5.487	2.776	1.444	0.712	0.363	0.179
2	0.418	0.173	-0.005	0.007	0.007	0.006
3	-2.375	-0.883	-0.351	-0.128	-0.062	-0.031
4	6.170	2.181	0.627	0.121	0.024	-0.002
5	-3.338	-0.318	0.327	0.306	0.173	0.097
6	7.036	2.769	0.993	0.385	0.173	0.078
7	-5.722	-3.482	-1.661	-0.800	-0.400	-0.200
8	-6.155	0.310	1.411	0.955	0.507	0.269
9	-0.718	1.172	0.725	0.337	0.162	0.081
10	-12.833	-9.060	-4.882	-2.359	-1.145	-0.564
11	12.966	11.241	7.718	4.443	2.344	1.188
12	-11.194	-14.874	-12.032	-7.889	-4.437	-2.395
13	-37.388	-30.782	-20.701	-12.296	-6.527	-3.377
14	-23.619	-12.312	-3.929	-0.233	0.585	0.382
15	-10.988	-18.476	-13.953	-7.904	-4.091	-2.015

E.5. Energy consumption estimation

We follow what most did such as (Rathi & Roy, 2023; Yao et al., 2023) We computed the energy cost for ANNs and SNNs in 45nm CMOS technology. The energy cost for 32-bit ANN MAC operation (4.6pJ) is 5.1×more than SNN addition operation (0.9pJ) (Horowitz, 2014).

Following prior work such as (Rathi & Roy, 2023; Yao et al., 2023), we computed the energy cost for both ANNs and SNNs using 45nm CMOS technology. The energy consumption of a 32-bit MAC operation in ANNs (4.6 pJ) is approximately 5.1× higher than that of an addition operation in SNNs (0.9 pJ) (Horowitz, 2014).

In ANNs, the computational cost per layer, denoted as $\mathcal{C}^{(l)}_{\text{ANN}}$, is defined as:

$$\mathcal{C}^{(l)}_{\text{ANN}} = \begin{cases} w_k \cdot h_k \cdot c_{in} \cdot h_{out} \cdot w_{out} \cdot c_{out}, & \text{Convolution layer} \\ f_{in} \cdot f_{out}, & \text{Fully-connected layer} \end{cases} \tag{22}$$

where w_k and h_k denote the kernel width and height; c_{in} and c_{out} are the numbers of input and output channels; h_{out} and w_{out} denote the height and width of the output feature map; and f_{in}, f_{out} represent the input and output feature dimensions in fully-connected layers.

For an SNN, the number of operations per layer is given by:

$$\mathcal{C}^{(l)}_{\text{SNN}} = \nu^{(l)} \cdot \mathcal{C}^{(l)}_{\text{ANN}} \tag{23}$$

where the spike rate $\nu^{(l)}$ for layer l is defined as the average spike rate per neuron over all inference timesteps:

$$\nu^{(l)} = \frac{1}{T} \sum_{t=1}^{T} \frac{N^{(l,t)}_{\text{spikes}}}{N^{(l)}_{\text{neurons}}} \tag{24}$$

Here, T denotes the total number of inference timesteps, $N^{(l,t)}_{\text{spikes}}$ is the number of spikes at timestep t in layer l, and $N^{(l)}_{\text{neurons}}$ is the number of neurons in layer l.

F. Membrane potential Distribution

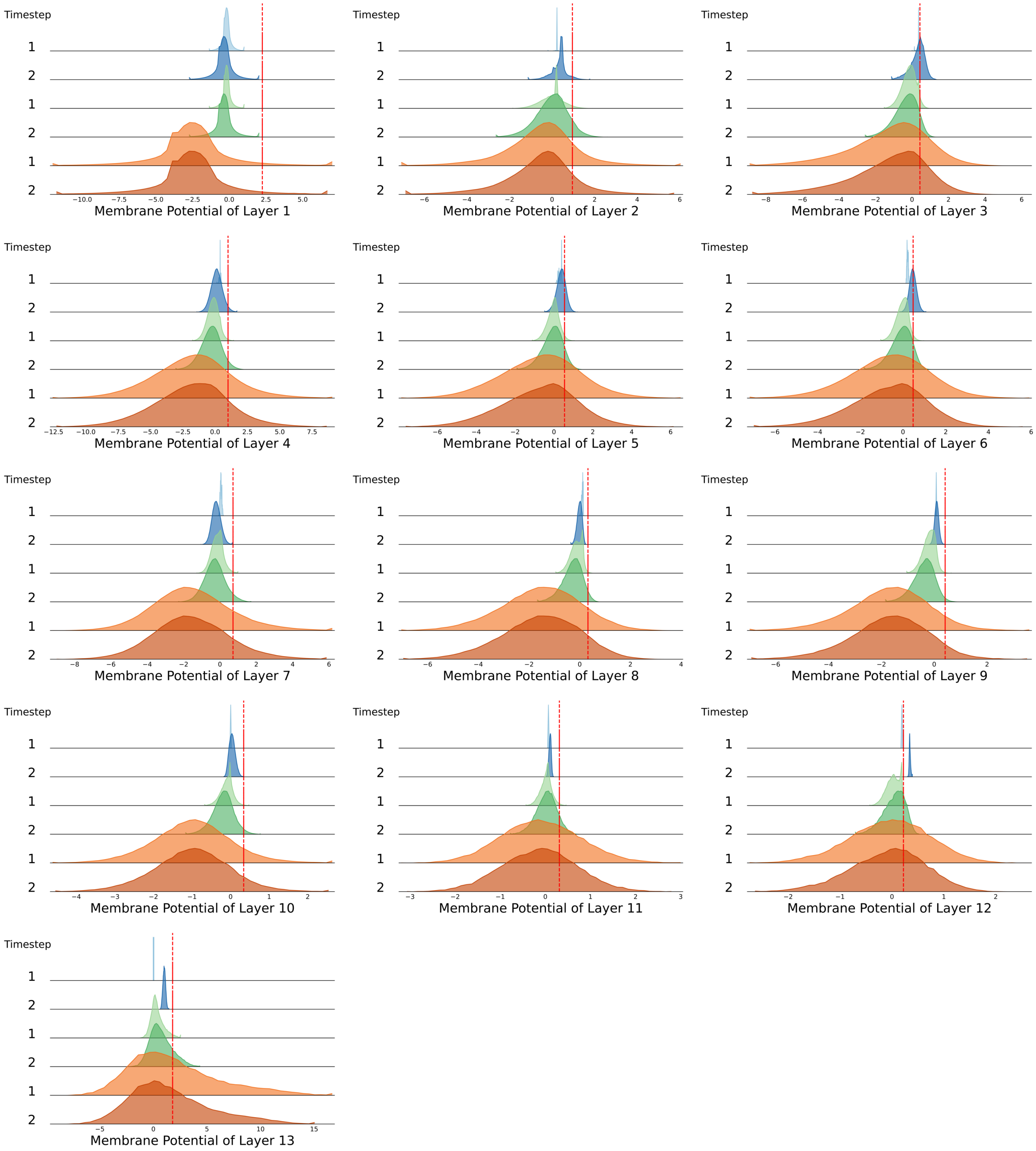


Figure 8. The membrane potential distributions of the first channel (randomly selected) across three modes (baseline, shuffle, and probabilistic) in VGG-16 on CIFAR-100. For comparison, the first two timesteps (t=1, t=2) from a total of eight timesteps (T=8) are selected for each mode. The baseline mode (blue) achieves an accuracy of 24.22%, while the shuffle mode (light green) improves accuracy to 70.54%, and the probabilistic mode (dark orange) further increases accuracy to 73.42%. The distributions are shown before firing, and the red dashed line indicates the threshold voltage (Vth) for the layer.

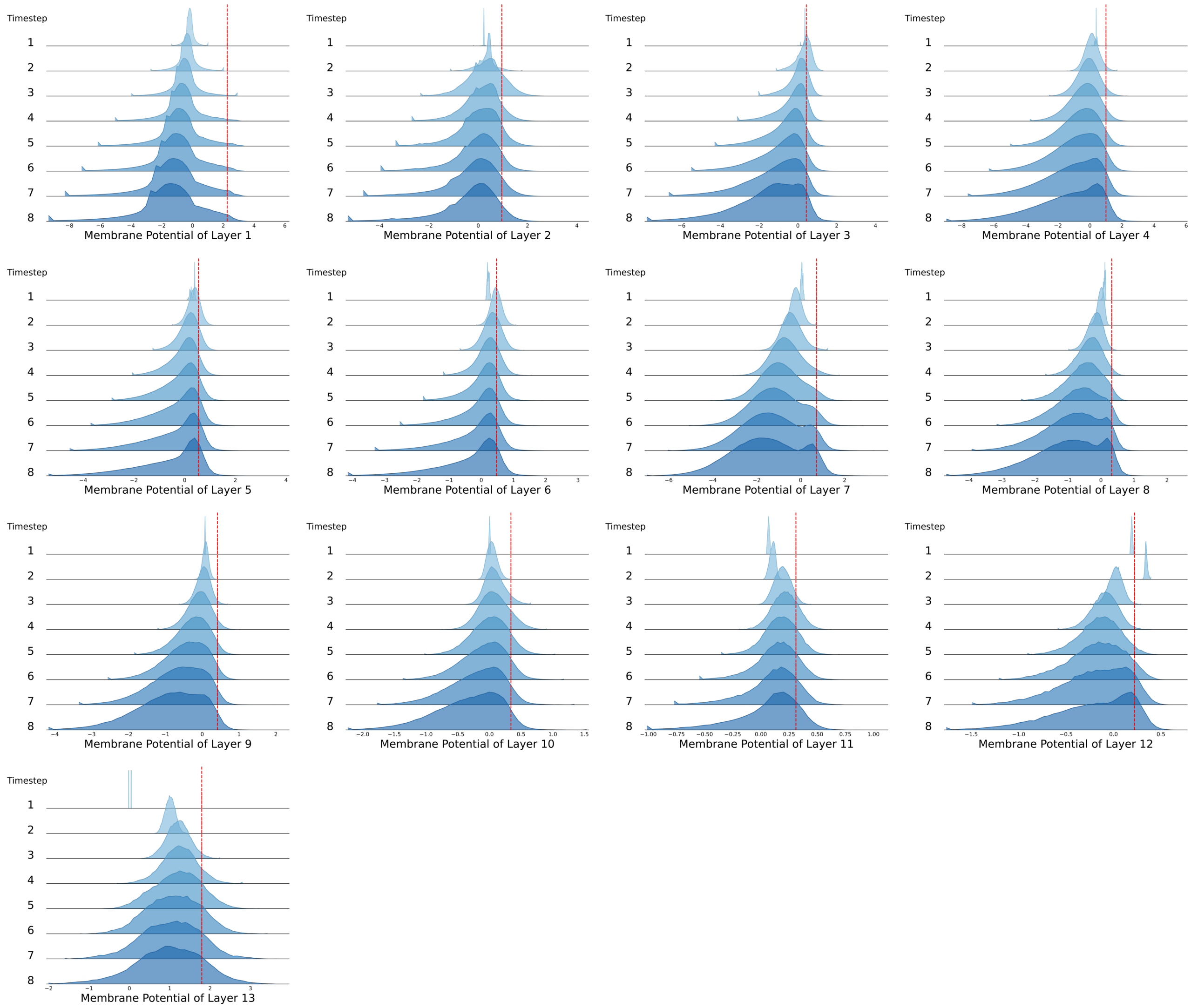


Figure 9. The membrane potential of the first channel (randomly selected) from layer 1 in SNNC baseline mode using VGG-16 on CIFAR-100 achieves an accuracy of 24.22% before firing.

The first two timesteps exhibit an abnormal distribution compared to those at t=4 to t=8. This discrepancy arises from the initially incorrect membrane potential before firing, which affects the firing rate and propagates errors layer by layer. A detailed quantifiable error analysis is provided in Appendix B. Furthermore, as shown in Figure 10, shuffling the membrane potential effectively alleviates this effect.

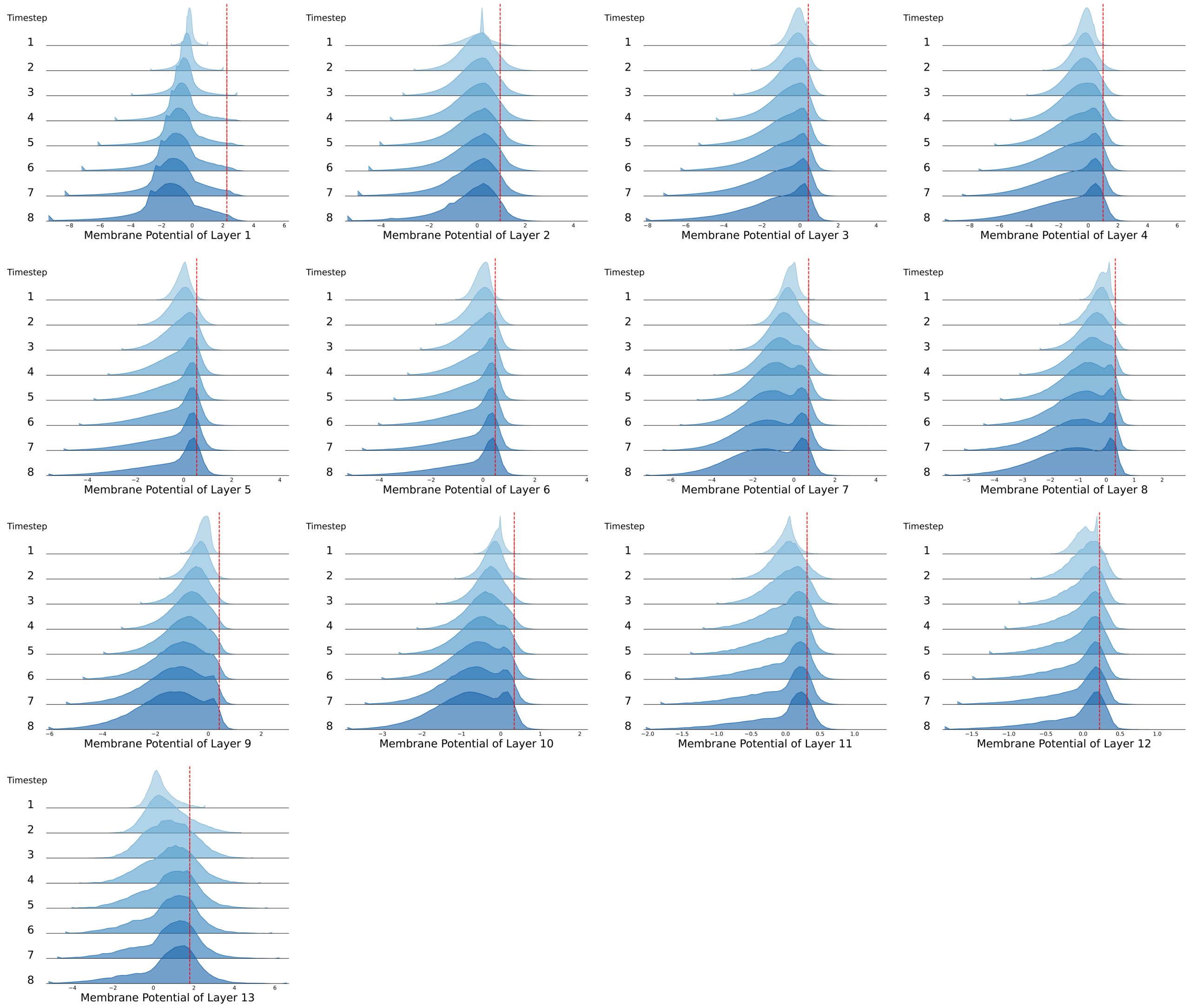


Figure 10. Membrane potential of the first channel (randomly selected) before firing in SNNC shuffle mode using VGG-16 on CIFAR-100. The achieved accuracy is 70.54%, indicating the impact of random spike rearrangement.

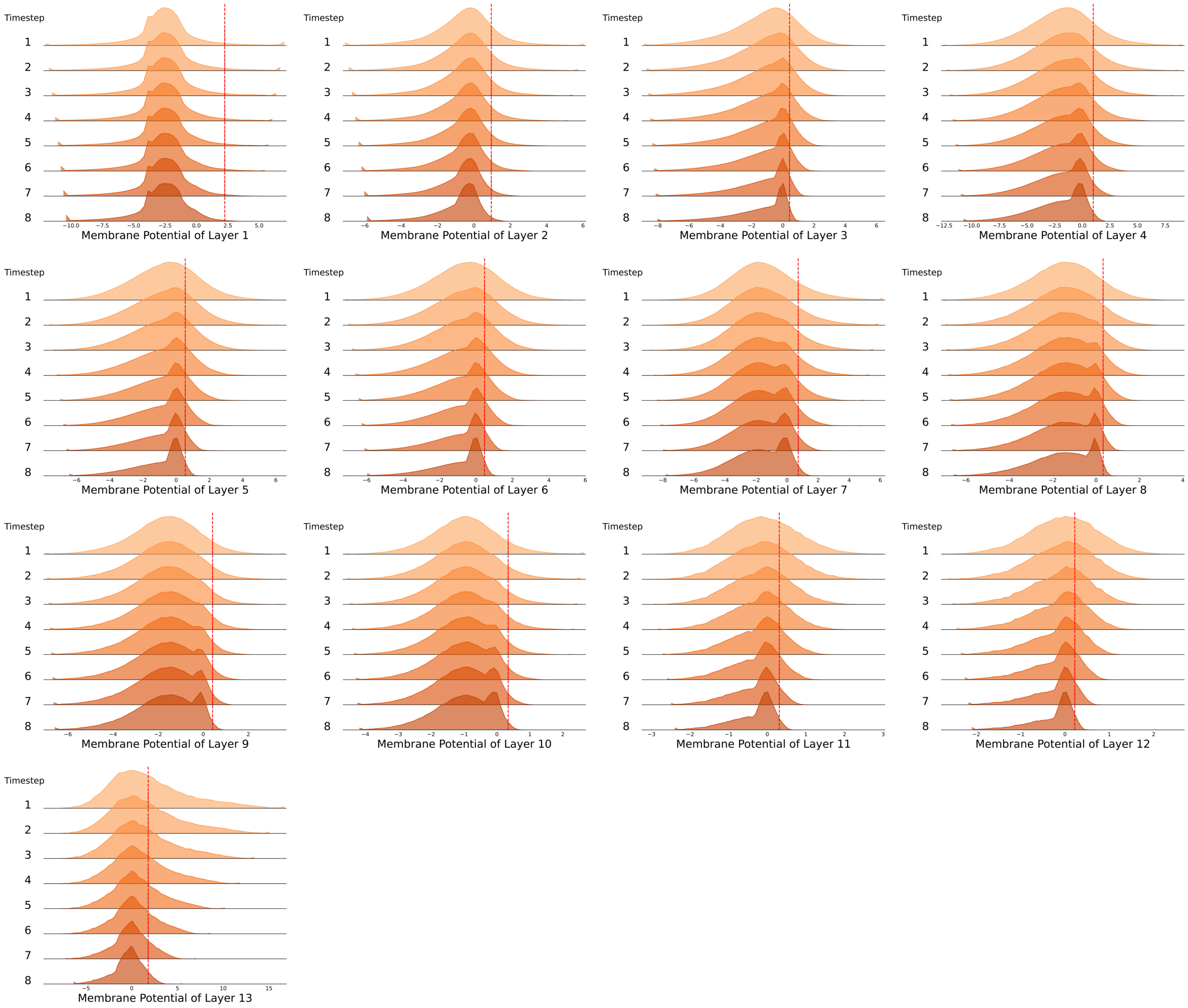


Figure 11. Membrane potential of the first channel (randomly selected) before firing in SNNC probabilistic mode using VGG-16 on CIFAR-100. The accuracy increases to 73.42%.

G. Permutations and ANN-SNN conversion

Heuristics behind permutations We come back to the original motivation, and the mysterious effect of temporal misalignment. To this end, we notice that permutations may act as a "uniformizer" of the inputs to the spiking neuron, which is highly related to notions of phase lag or unevenness of the inputs (see (Li et al., 2022) and (Bu et al., 2022c), respectively).

Theorem 2. *Suppose we have N spiking neurons that produced spike trains $s_i[1], s_i[2], \ldots, s_i[T]$, $i = 1, \ldots, N$. Furthermore, suppose that these spike trains are modulated with weights $w_1, \ldots, w_N$, respectively. For a given permutation $\pi = (\pi_1, \ldots, \pi_N)$, let πs_i denote the permutation of the spike train s_i. Then, for every $t_1, t_2 \in \{1, 2, \ldots, T\}$,*

$$E_\pi[\sum w_i \pi s_i[t_1]] = E_\pi[\sum w_i \pi s_i[t_2]].$$

Proof. It is enough to prove that for each $i = 1, \ldots, N$,

$$E_\pi[\pi s_i[t_1]] = E_\pi[\pi s_i[t_2]]. \tag{25}$$

Let $A(t_i)$ be the cardinality of the set of all the permutations that end up with a spike in step t_i, and note that the probability of having a spike at t_i is then $\frac{A(t_i)}{T!}$. But, for each permutation that ends up with a spike at t_i, one can find a permutation that ends up with a spike at t_2 (by simply applying a cyclic permutation) and moreover this correspondence is bijective. In particular $A(t_i)$ is independent of i. The equation (25) and the statement follow. □

The previous result deals with the expected outputs with respect to the permutations. When it comes to the action of a single permutation, we make the following observation. The effect of a single permutation is mostly visible on spike trains that have a **low number of spikes**. This, in turn, is related to the situation where the input to the neuron is low throughout time, and it takes longer for a neuron to accumulate enough potential in order to spike, hence the neuron spikes at a later time during latency. In this case, a single permutation of the output spike(s) actually move the spikes forward in time (in general) and as such contributes to the elimination of the unevenness error, which appears when the input to a neuron in the beginning is higher than the average input through time (hence, the neuron produces superfluous spikes in the beginning, which shouldn't be the case).

Table 12. Recorded accuracy after $t \leq T$ time steps, when the baseline model is "permuted" in latency T. Setting is VGG-16, CIFAR-100.

Method	ANN	t=1	t=2	t=4	t=8	t=16	t=32
QCFS (Bu et al., 2022c)		49.09	63.22	69.29	73.89	75.98	76.52
Ours (Permute)	T=4	68.11	71.91	74.2			
Ours (Permute)	T=8	71.76	74.11	75.53	75.86		
Ours (Permute)	T=16	72.75	74.27	75.63	76.0	76.39	
Ours ((Permute)	T=32	73.15	75.23	75.74	76.27	76.59	76.52
RTS (Deng & Gu, 2021)		1.0	1.03	23.76	43.81	56.23	67.61
Ours (Permute)	T=4	22.9	30.78	34.54			
Ours ((Permute)	T=8	45.11	52.7	59.2	62.58		
Ours ((Permute)	T=16	54.58	64.37	68.6	70.8	71.79	
Ours (Permute)	T=32	62.76	69.12	71.76	73.31	74.09	74.6

Remarks:

1. In Table 12 we combine permutations with baseline models in fixed latency T. Afterwards, we record the accuracies of such "permuted" model for lower latencies t. We can notice a sharp increase in the accuracies compared to the baselines, and in particular, the variance in accuracies across t is reduced.
2. **Baseline analysis:**
   (a) SNN models converted from a pretrained ANN aim to approximate the ANN activation values with firing rates. In particular, in lower time steps, the approximation is too coarse as the firing rate has only few possibilities to use to approximate the ANN (continuous) values. For example, in $T = 1$, the baselines are attempting to approximate ANN activations with binary values 0 and θ.

(b) Moreover, at each spiking layer, the spiking neurons at early time steps, use only the outputs of the previous spiking layer from the same, early, time steps. As this information is already too coarse, **the approximation error accumulates throughout the network**, finally yielding in models that are underperfoming in low latencies.

(c) With longer latencies, the model is using more spikes and is able to approximate the ANN values more accurately, and to correct the results from the first time steps.

3. **Effect of permutations:**

(a) When performing permutations on spike trains after spiking layers in the baseline models, the input to the next spiking layer in lower time steps, **no longer depends only on the outputs of the previous layer in the same lower time steps, but it depends on the outputs in all time steps** T.

(b) In particular, when spiking layer is producing spikes at time step $t = 1$, it does so "taking into account" (via permutation) outputs at all the time steps from the previous spiking layer.

(c) As a way of example, consider two spiking neurons N_1 and N_2, where N_2 receives the weighted input from N_1. If a spiking neuron N_1 in one layer has produced spike train $s = [1, 0, 0, 0]$, in approximating ANN value of $.25$, then a spiking neuron N_2 at the first time step will use 1 as the approximation and will receive the input $W \cdot 1$ from neuron N_1. However, after a generic permutation of s, the probability of having zero at the first time step of output of neuron N_1 is $\frac{3}{4}$ (as oppose to having 1 with probability $\frac{1}{4}$), and at the first time step neuron $N2$ will most likely receive the input $W \cdot 0 = 0$ from neuron N_1, which is a rather better approximation for $W \cdot .25$ than W itself.

(d) This property of receiving input at lower t but taking into account the previous layer spike outputs at all the time steps is not only exclusive to lower t. Indeed, at every time step $t \leq T$, the input at a spiking layer is formed by taking into account spiking train outputs from the previous layer at all the time steps, but having already accounted for for the observed input at the first $t < 1$ steps.

(e) In general, the permutations overall increase the performance of the baselines because the spike trains are "uniformized" in accordance to their rate, and the accumulation error is reduced. If a layer l has produced spike outputs that well approximate the l layer in ANN, then, after a generic permutation, at each time step starting with the first, the next layer is receiving the most likely binary approximation of those rates.

(f) This is nothing but Theorem 2 in visible action.

(g) Besides Table 12, we provide further evidence on how permutation affect the baselines through the observed membrane potential in the following sections.

G.1. The effect of permutations on performance: Further experiments

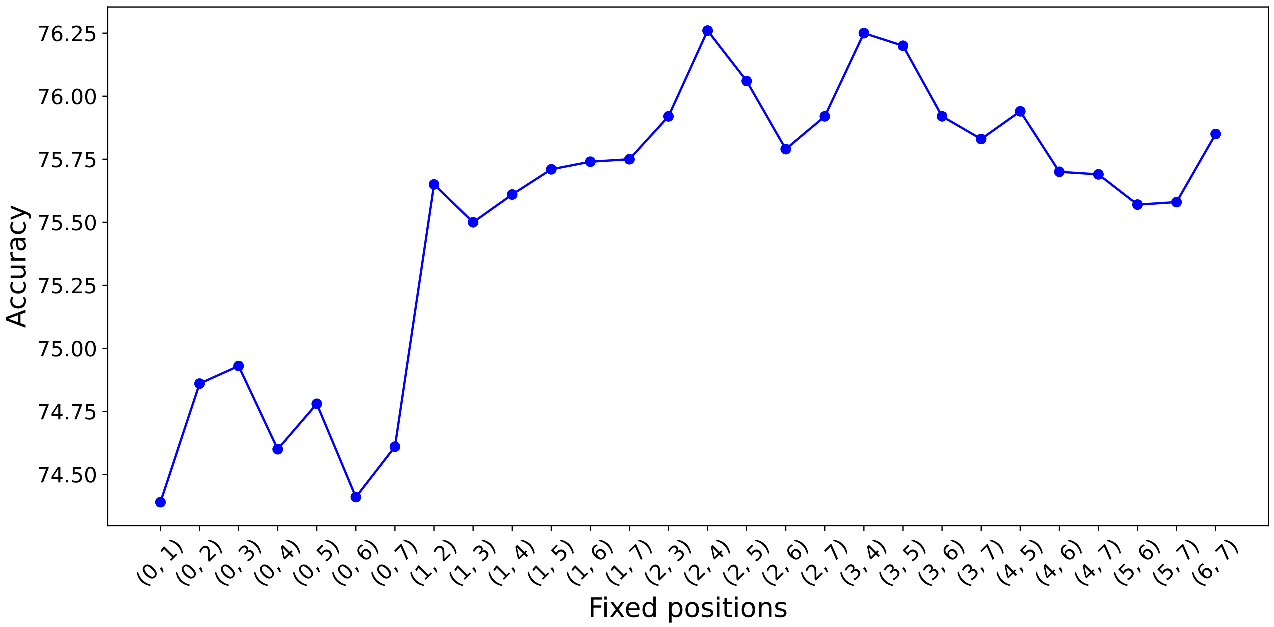


Figure 12. Accuracy comparison for permutations over 8 time steps, fixing given pairs of time steps. Setting is VGG-16, CIFAR-100. The baseline (QCFS) accuracy is 73.89%, ANN accuracy is 76.21%.

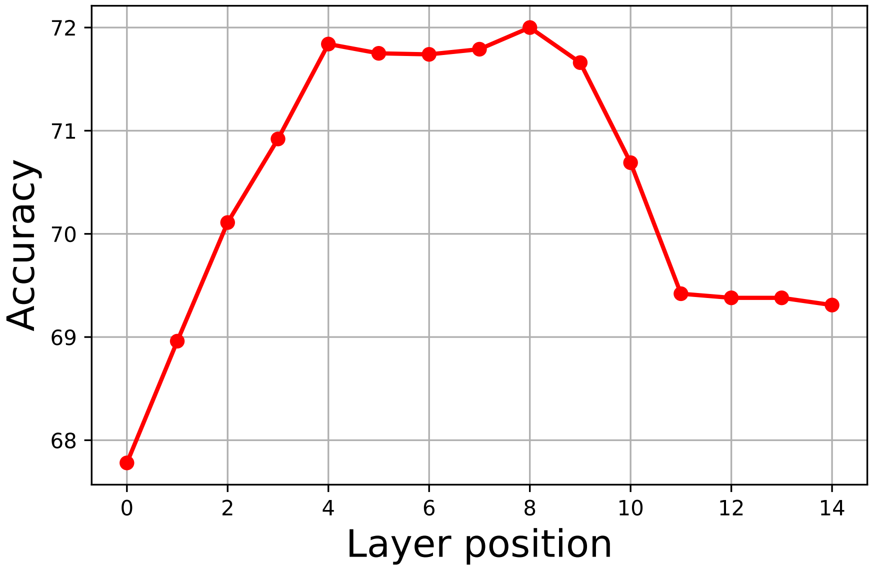


Figure 13. Accuracy of the model when a permutation is applied on a single layer using QCFS baseline. Setting is VGG-16, $T = 4$, CIFAR-100. Baseline accuracy is 69.31%, ANN accuracy is 76.31%

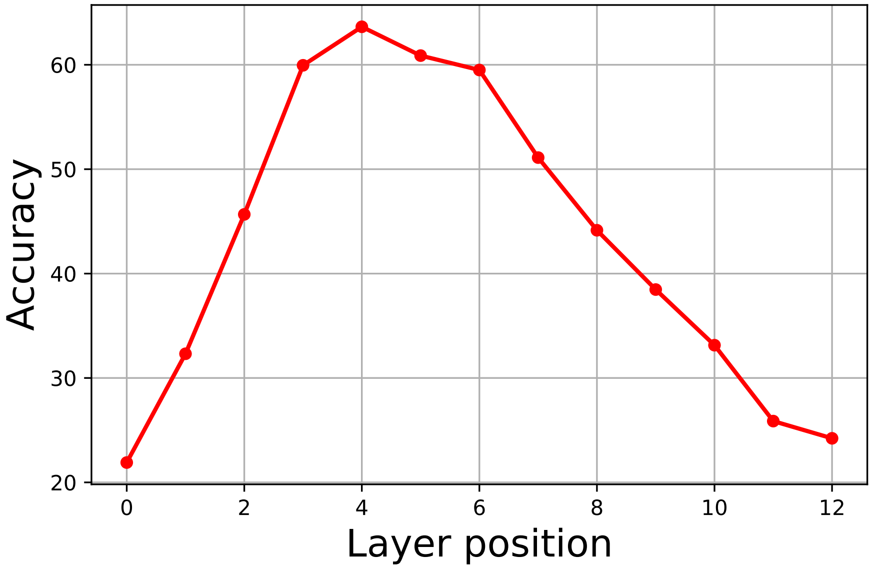


Figure 14. Accuracy of the model when a permutation is applied on a single layer using SNNC baseline. Setting is VGG-16, $T = 8$, CIFAR-100. Baseline accuracy without calibration is 24.22%, ANN accuracy is 77.87%