

Efficient ANN-Guided Distillation: Aligning Rate-based Features of Spiking Neural Networks through Hybrid Block-wise Replacement

Shu Yang^{1,†}, Chengting Yu^{1,2,†}, Lei Liu¹, Hanzhi Ma^{1,2,*}, Aili Wang^{1,2,*}, Erping Li^{1,2,*}

¹ ZJU-UIUC Institute, Zhejiang University

² College of Information Science and Electronic Engineering, Zhejiang University

Abstract

Spiking Neural Networks (SNNs) have garnered considerable attention as a potential alternative to Artificial Neural Networks (ANNs). Recent studies have highlighted SNNs' potential on large-scale datasets. For SNN training, two main approaches exist: direct training and ANN-to-SNN (ANN2SNN) conversion. To fully leverage existing ANN models in guiding SNN learning, either direct ANN-to-SNN conversion or ANN-SNN distillation training can be employed. In this paper, we propose an ANN-SNN distillation framework from the ANN-to-SNN perspective, designed with a block-wise replacement strategy for ANN-guided learning. By generating intermediate hybrid models that progressively align SNN feature spaces to those of ANN through rate-based features, our framework naturally incorporates rate-based backpropagation as a training method. Our approach achieves results comparable to or better than state-of-the-art SNN distillation methods, showing both training and learning efficiency.

1. Introduction

Spiking Neural Networks (SNNs) are third-generation neural networks with biological interpretability [36, 42]. Unlike Artificial Neural Networks (ANNs), SNN neurons simulate the activity of biological neurons, providing both spatial and temporal expressiveness, while outputting binary spikes [39]. This makes SNNs highly energy-efficient for deployment on neuromorphic hardware [1, 8, 40], offering promising development prospects.

Current SNN training methodologies primarily follow two paradigms: direct training through Backpropagation Through Time (BPTT) and ANN-to-SNN (ANN2SNN) conversion. BPTT-based approaches utilize surrogate gradients to approximate the non-differentiable spike function [38, 53]. While these methods achieve competitive per-

formance with limited timesteps [11, 25, 33, 38, 45, 53, 55, 65], they typically encounter increased computational costs due to temporal gradient calculations [37, 56, 59, 67]. Additionally, the surrogate gradients do not always align with the theoretical gradients, requiring further refinement of BPTT-based SNN training. ANN-to-SNN conversion methods establish analytical mappings between Integrate-and-Fire (IF) neuron firing rates and ReLU activations [43]. Recent approaches adopt an ANN→QNN→SNN pipeline [4, 5, 22, 24, 31], incorporating quantization-aware training (QAT) [3, 60] as an intermediate step. However, this introduces mapping errors during QNN-to-SNN conversion, particularly pronounced with reduced timesteps. Furthermore, while empirical evidence suggests Leaky Integrate-and-Fire (LIF)-based architectures outperform IF-based SNNs [23, 37], the absence of theoretical support for converting LIF neurons remains a limitation.

Notably, given the inherent challenges of both mainstream training approaches, recent work has drawn inspiration from both ends, proposing innovative approaches that offer unique advantages and insights. (1) ANN-guided distillation aims to incorporate "dark knowledge" from ANNs into direct SNN training through a non-strict mapping [18, 41, 57, 64, 68]. In this method, the ANN provides supervisory signals, such as logits or feature alignment, which encourages the SNN to align its features with those of the ANN. This creates an implicit ANN-to-SNN conversion [6], where the ANN serves as a form of regularization, subtly guiding the SNN's learning. (2) Recent studies have noted that SNN features are often represented in a rate-based manner [13, 28, 44], similar to the feature representations in ANNs. The similarity in rate-based representations has been further highlighted by recent findings [34]. This has prompted the development of lightweight training methods that optimize SNNs to leverage this alignment with ANN representations [59]. However, both approaches face limitations. ANN-guided distillation struggles to explicitly align intermediate feature spaces due to the limited expressive power of binary spike representations, hindering effective SNN-ANN feature map matching [41, 64]. On the other

[†]Equal contribution, ^{*}Corresponding authors: mahanzhi@zju.edu.cn, ailiwang@intl.zju.edu.cn, liep@zju.edu.cn.

hand, rate-based backpropagation, though capturing rate-based representations well, is entirely self-driven within the SNN. This self-driven training introduces gradient distortions in intermediate variables due to the Straight-Through Estimator (STE) [2], which impedes optimal convergence of rate-based features.

These observations inspire us to explore the integration of ANN-guided training with rate-based representation. While spike-based representations in ANN-guided training capture features over T timesteps, they impose strict alignment constraints with intermediate ANN features [41, 64], underscoring the need for a more flexible alignment approach. By adopting a rate-based backpropagation framework, we leverage a more intuitive and efficient training alternative compared to BPTT’s spike focus. Moreover, integrating an ANN-guided alignment mechanism within rate-based backpropagation may alleviate gradient distortions from the STE, adding a subtle regularization term that guides convergence toward optimal points.

Building on these insights, we delve deeper into the synergy between ANN-guided training and rate-based representations, aiming to bridge the gap between the two. To effectively learn rate-based representations, rather than strictly aligning the intermediate features of ANNs and SNNs, we propose an innovative approach that aligns the mapping relationships between ANN and SNN modules. In other words, instead of hard-aligning features, we encourage the SNN’s input-output mappings to approximate those of the ANN, forming an implicit functional alignment. Following this idea, we designed an ANN-guided scheme based on block-wise replacement within the rate-based backpropagation framework. In this setup, the terminal modules of the ANN act as auxiliary branches in the feedforward process, allowing the ANN to more effectively guide the learning of rate-based representations. This approach achieves two main goals: (1) flexible alignment of intermediate features between the ANN and SNN, and (2) mitigation of gradient distortion in rate-based backpropagation. Empirical results demonstrate that our method enables more effective knowledge transfer, ultimately enhancing the SNN’s learning process and achieving state-of-the-art performance across CIFAR-10, CIFAR-100, DVS-CIFAR10, and ImageNet.

Our contributions can be summarized as follows:

1. We introduce a framework for ANN-guided learning in SNNs that effectively leverages pretrained ANN knowledge for SNN distillation.
2. By using rate-based representation to construct the guidance signal, our approach enables efficient distillation training using rate-based backpropagation.
3. Our method achieves or surpasses state-of-the-art performance on the ImageNet, CIFAR-10, CIFAR-100, and CIFAR-10-DVS datasets.

2. Related Work

SNNs Learning Methods Among primary learning approaches for SNNs, direct training based on BPTT faces significant computational overhead, with memory and time complexity increasing linearly with the number of timesteps [11, 25, 33, 37, 46, 55, 56, 65]. Some studies have drawn inspiration from the online learning strategies used in Recurrent Neural Networks (RNNs) [52], proposing online learning methods that retain the distinctive characteristics of SNN neurons, aimed at reducing BPTT’s substantial memory requirements and demonstrated effectiveness on large-scale datasets [37, 56, 67]. Recently, [59] advanced an efficient direct training method for SNNs by using rate-based representation and backpropagation to decouple memory and training time from timesteps. We aim for the SNN model to learn rate-based representations, and since rate-based backpropagation inherently suffers from gradient estimation errors in intermediate variables, incorporating ANN-guided rate-based representation learning naturally complements this issue.

ANN-guided Training Training-required ANN2SNN conversions [4, 5, 22, 24, 31] typically grounded in the explicit mapping theories of QNN to SNNs, first applies QAT [3, 60] to obtain a QNN, followed by mapping quantized parameters to SNN neuron parameters. However, these conversion methods remain tied to IF neurons, which limits their ability to leverage the higher-performance LIF neurons. Knowledge distillation (KD) is a well-established transfer learning technique effectively utilized for model compression [19, 35, 47, 50, 61]. Recent works have adapted KD for SNNs [18, 20, 29, 30, 48, 57, 62, 63] to enhance the performance of student SNNs by pre-training an ANN as a guide for SNN training. Methods like [48, 49] distill knowledge into an ANN teacher model, then convert the distilled ANN into an SNN. Other approaches, such as [18, 20], follow a similar strategy by optimizing SNNs through a joint ANN framework. In addition, [41, 64] work to establish a mapping between the intermediate feature spaces of SNNs and ANNs, enabling direct feature distillation. However, refining the feature space for ANN-guided SNN learning continues to pose challenges that warrant further exploration.

3. Method

3.1. Preliminary

Each neuron in an SNN has a membrane potential and a threshold. After receiving a series of inputs, when the membrane potential reaches the threshold, it fires a binary spike and resets the membrane potential. We consider the commonly used LIF neuron model, whose equations for the membrane potential are given by:

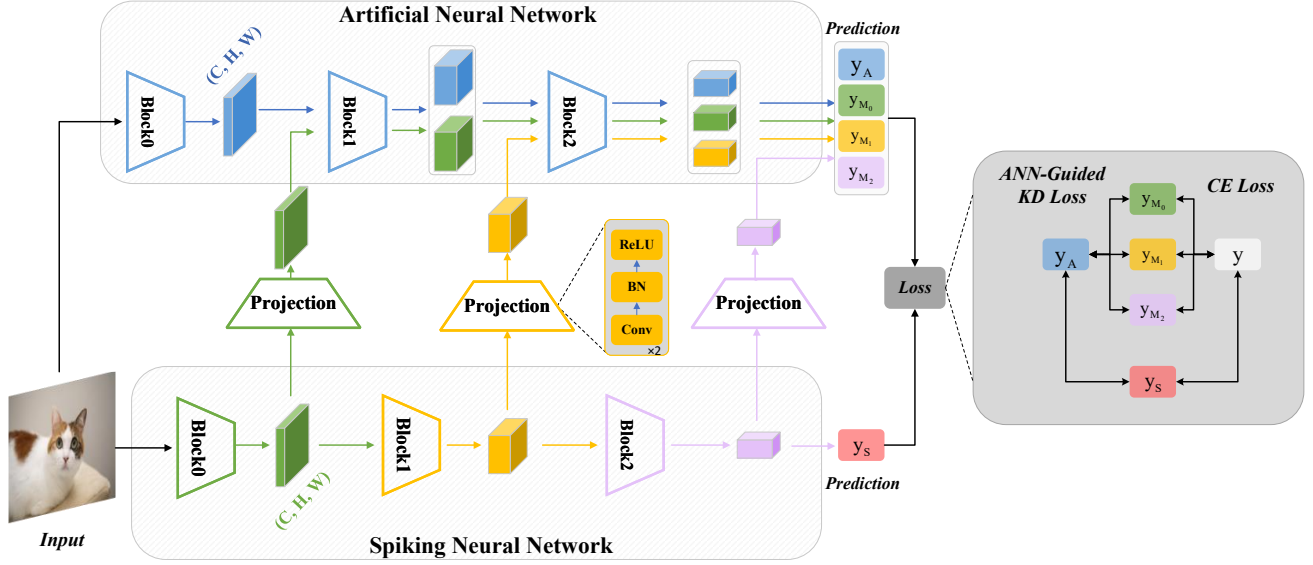


Figure 1. Framework Overview. In the rate propagation of the SNN, the rate-based representation is transformed via learnable modules to align with the ANN’s intermediate feature space and then propagated forward along the subsequent ANN layers, yielding a series of hybrid model outputs, $\{y_{M_0}, y_{M_1}, y_{M_2}\}$. At the logits end, losses are computed among the outputs of the ANN (y_A), the SNN (y_S), and the hybrid models. The ANN-guided distillation loss interacts with the ANN output, while the standard cross-entropy loss interacts with the hard labels y .

$$\begin{cases} u_{t+1}^l = \lambda(u_t^l - V_{th}s_t^l) + W^l s_t^{l-1} + b^l, \\ s_{t+1}^l = H(u_{t+1}^l - V_{th}), \end{cases} \quad (1)$$

where u_t^l and s_t^l represent the membrane potential and fired spikes of the l -th layer neurons at time t , respectively. Here, λ is the membrane potential constant, V_{th} is the membrane threshold, W^l denotes the weights of the l -th layer neurons, b^l is the bias, and H is the step function. In gradient surrogate methods, this function is typically replaced by differentiable functions such as the sigmoid.

3.2. ANN-guided SNN Training Objectives

We first partition the ANN model into different blocks based on depth and feature map size, aligning with current mainstream architectures that stack blocks. Initially, we perform rate encoding on the inputs, defining the ANN model as $A = \{A_0, A_1, \dots, A_n\}$ and the SNN model as $S = \{S_0, S_1, \dots, S_n\}$, where n represents the number of blocks in each model.

We then define a series of ANN-SNN hybrid models as follows:

$$M_k = \{S_0, S_1, \dots, S_k, C_k, A_{k+1}^{frozen}, \dots, A_n^{frozen}\} \quad (2)$$

where C_k is a learnable module that maps features from the SNN rate-based representation space to the ANN representation space, with $0 \leq k < n$. The trainable weights are

limited to the SNN model S and C_k , while the weights of the full ANN model sequence $A = \{A_0, A_1, \dots, A_n\}$, including $\{A_{k+1}^{frozen}, \dots, A_n^{frozen}\}$, remain fixed and are not updated.

We present the definition of the learning loss function for ANN-guided SNN training. Let the outputs of the models be defined as follows: the output of the ANN is denoted as y_A , the output of the SNN is represented as y_S , and the outputs of the hybrid models are given by $\{y_{M_k}\}_{k=1}^{n_b}$, where n_b is the number of hybrid models. The true labels are denoted as y . To facilitate the efficient transfer of knowledge from the ANN to the SNN, we define the loss of the k -th hybrid model as follows:

$$L_{blk}^k = (1 - \alpha) \cdot L_{ce}(y_{M_k}, y) + \alpha \cdot L_{kd}(y_{M_k}, y_A), \quad (3)$$

where L_{ce} represents the cross-entropy loss for classification tasks, L_{kd} denotes the logit-based knowledge distillation loss defined via KL divergence, and α is a hyperparameter that balances the contributions of the cross-entropy and distillation losses. The learning objective of our proposed ANN-SNN distillation framework is defined as:

$$L_{total} = (1 - \alpha) \cdot L_{ce}(y_S, y) + \alpha \cdot L_{kd}(y_S, y_A) + \sum_{k=1}^{n_b} \beta_k L_{blk}^k. \quad (4)$$

Here, β_k represents the loss weight for the hybrid model sequence M_k . Our detailed ANN-guided SNN distillation framework is illustrated in Fig. 1.

3.3. Implicit Feature Alignments

We now outline the process through which we developed the approach for rate-based ANN-guided SNN distillation learning. The theoretical foundation of ANN2SNN is that the firing rate of neurons has an equivalent relationship with the ReLU activation function: $\phi^l[T] = W^l \phi^{l-1}[T] - \frac{u^l[T] - u^l[0]}{T}$, where $\phi^l[T]$ represents the firing rate of the l -th layer SNN neurons after T time steps, and $u^l[T]$ and $u^l[0]$ denote the membrane potential of the l -th layer neurons at time step T and its initial value, respectively.

Let the output of the l -th layer neurons in the ANN be denoted as a^l , and the conversion error of the l -th layer can be defined as:

$$Err^l = \|\phi^l[T] - a^l\|. \quad (5)$$

Inspired by the conversion error in ANN2SNN, we aim to define an ANN-guided loss in a similar form. This approach circumvents the challenge of explicitly defining the intermediate feature space alignment between ANN and SNN. The rate-based intermediate feature outputs of the model sequence $\{S_0, S_1, \dots, S_k\}$ are denoted as $F_{S_k}^{rate}$, while the intermediate feature outputs of the ANN model sequence $\{A_0, A_1, \dots, A_k\}$ are denoted as F_{A_k} . A mapping g_k is assumed to exist, which maps the rate-based feature $F_{S_k}^{rate}$ to F_{A_k} , expressed as $F_{S_k} = g_k(F_{S_k}^{rate})$. This allows the distance between the SNN rate-based representation intermediate feature space and the ANN intermediate feature space to be formulated as follows, similar to Eq. (5):

$$Err^k = \|F_{S_k} - F_{A_k}\|. \quad (6)$$

Rather than enforcing strict imitation of the ANN's intermediate features, the SNN is guided to achieve high task performance through effective learning. No explicit learning objective is set to minimize Err^k . Instead, F_{S_k} is forwarded through the remaining ANN model sequence, and the loss is computed at the final logit layer, where the conversion error is inherently integrated into the loss calculation. For the knowledge distillation component, the loss is defined based on Kullback-Leibler (KL) divergence without temperature:

$$L_{kd}(x, y) = \text{KL}(y \parallel x) = \sum_i y_i \cdot \log \frac{y_i}{x_i}, \quad (7)$$

where y_i represents the probability of class i as predicted by the teacher model, and x_i denotes the corresponding probability predicted by the student model.

Let M^p denote the sequence after softmax operation on $\{A_{k+1}^{frozen}, \dots, A_n^{frozen}\}$. $L_{kd}(y_{M_k}, y_A)$ is denoted as L_{SA} , and the distillation loss is formulated as:

$$\begin{aligned} L_{SA} &= \text{KL}(M^p(F_{A_k}) \parallel M^p(F_{S_k})) \\ &= \sum_i M^p(F_{A_k})_i \log \frac{M^p(F_{A_k})_i}{M^p(F_{S_k})_i}. \end{aligned} \quad (8)$$

The gradient with respect to F_{S_k} is derived as:

$$\frac{\partial L_{SA}}{\partial F_{S_k}} = \sum_i -\frac{M^p(F_{A_k})_i}{M^p(F_{S_k})_i} \cdot \frac{\partial M^p(F_{S_k})_i}{\partial F_{S_k}}. \quad (9)$$

When the ANN output shows high confidence for class i while SNN features processed through M^p yield low confidence for the same class, the gradient is amplified, indicating strong ANN guidance for SNN learning. Conversely, if the ANN exhibits low confidence for class i but M^p outputs a high probability, the gradient is reduced, and updates are largely governed by the standard cross-entropy loss $L_{ce}(y_{M_k}, y)$. This approach prevents the SNN from over-adjusting to uncertain ANN predictions, thereby enhancing stability in the learning process.

Moreover, the training objective implicitly integrates learning that focuses on minimizing Err^k . We can approximate $M^p(F_{A_k})_i$ using first-order Taylor expansion:

$$M^p(F_{A_k})_i \approx M^p(F_{S_k})_i + \frac{\partial M^p(F_{A_k})_i}{\partial F_{S_k}} \cdot (F_{A_k} - F_{S_k}). \quad (10)$$

The gradient can be derived as follows:

$$\frac{\partial L_{SA}}{\partial F_{S_k}} = \sum_i (\xi - 1) \cdot \frac{\partial M^p(F_{S_k})_i}{\partial F_{S_k}}, \quad (11)$$

where:

$$\xi = \frac{\partial M^p(F_{A_k})_i}{\partial F_{S_k}} \cdot \frac{F_{S_k} - F_{A_k}}{M^p(F_{S_k})_i}. \quad (12)$$

It can be observed that Err^k is included in ξ . In Sec. 4.4, we demonstrate through experiments that our method facilitates the alignment of F_{S_k} and F_{A_k} , thereby reducing Err^k . This training objective enables the SNN to adaptively learn rate-based intermediate features, aligning them more effectively with the feature representations of the ANN model.

3.4. Rate-based backpropagation

During SNN forward propagation, T spike-based forward passes are performed without gradient tracking, with statistics required for backpropagation updated iteratively. This is followed by a single rate-based pass, at which point the gradient computation graph is established. As shown in Algorithm 1, during the hybrid model M_k 's forward propagation, we operate only on the rate-based pass, with C_k directly receiving rate-based SNN intermediate features. This approach enables the SNN to learn rate-based representations efficiently.

Specifically, define the average rate as $r^l = \mathbb{E}[s_t^l] = \frac{1}{T} \sum_{t \leq T} s_t^l$. The input to the linear layer is given by $I_t^l = W^l s_t^{l-1}$, and the average rate of the input can be expressed as $c^l = \mathbb{E}[I_t^l] = \mathbb{E}[W^l s_t^{l-1}] = W^l \mathbb{E}[s_t^{l-1}] = W^l r^{l-1}$. We

Table 1. **Performance comparison across CIFAR-10, CIFAR-100, and ImageNet datasets.** Results are averaged over five runs, except for ImageNet single-crop evaluations. * indicates that the SNN model architecture is SEW-ResNet-34. † indicates that the ANN teacher model is ResNet-50.

Method	Model	Time Steps	CIFAR-10 Top-1 Acc. (%)	CIFAR-100 Top-1 Acc. (%)	ImageNet Top-1 Acc. (%)
STBP-tdBN [66]	ResNet-19	4	92.92	-	-
	ResNet-34	6	-	-	63.72
Dspike [33]	ResNet-18	6	94.25	74.24	-
		4	93.66	73.35	-
		2	93.13	71.68	-
	ResNet-34	6	-	-	68.19
RecDis-SNN [16]	ResNet-19	6	95.55	-	-
		4	95.53	74.10	-
		2	93.64	-	-
TET [10]	ResNet-19	4	94.44	74.47	-
	ResNet-34	6	-	-	64.79
OTTT [56]	ResNet-34	6	-	-	65.15
SEW ResNet [14]	ResNet-34*	4	-	-	67.04
Real Spike [17]	ResNet-34	4	-	-	67.69
GLIF [58]	ResNet-18	6	94.88	77.28	-
		4	94.67	76.42	-
		2	94.15	74.60	-
	ResNet-19	2	94.44	75.48	-
		4	-	-	67.52
	ResNet-18	6	95.90	79.02	-
		4	95.61	78.26	-
		2	94.75	75.97	-
	ResNet-34	4	-	-	70.01
		4	-	-	-
LaSNN [20]	VGG16	100	91.22	61.52	-
KDSNN [57]	ResNet-18	4	93.41	-	-
Joint A-SNN [18]	ResNet-18	4	95.45	77.39	-
		2	94.01	75.79	-
EnOF [15]	ResNet-19	2	96.19	82.43	-
	ResNet-34	4	-	-	67.40
TSSD [68]	ResNet-18	2	93.37	73.40	-
SAKD [41]	ResNet-19	4	96.06	80.10	-
	ResNet-34	4	-	-	70.04
SuperSNN [64]	ResNet-19	6	95.61	77.45	-
Ours	ResNet-18	6	96.14 $\pm$ 0.03 (96.17)	79.40 $\pm$ 0.16 (79.50)	-
		4	95.92 $\pm$ 0.09 (96.01)	78.85 $\pm$ 0.19 (79.08)	-
		2	95.19 $\pm$ 0.10 (95.31)	77.06 $\pm$ 0.16 (77.25)	-
	ResNet-19	2	96.56 $\pm$ 0.06 (96.66)	81.44 $\pm$ 0.12 (81.57)	-
	ResNet-34*	4	-	-	68.12
	ResNet-34	4	-	-	70.64
	ResNet-34†	4	-	-	71.76

use c^l to estimate I_t^l , and the weight gradient is formulated as follows:

$$(\nabla_w L)_{\text{rate}} = \frac{\partial L}{\partial c^l} \cdot \frac{\partial c^l}{\partial W^l} = \frac{\partial L}{\partial c^l} \cdot r^{l-1\top}. \quad (13)$$

Algorithm 1 ANN-guided SNN Distillation Training

Require:

- 1: Pretrained ANN model $A = \{A_0, A_1, \dots, A_n\}$
- 2: SNN model $S = \{S_0, S_1, \dots, S_n\}$
- 3: Dataset $\mathcal{D}$
- 4: Statistical parameters $\{\rho_t^l, g_t^l\}$ for rate-based backpropagation

Ensure: Trained SNN model S

- 5: Define $\{M_k^p = \{C_k, A_{k+1}^{\text{frozen}}, \dots, A_n^{\text{frozen}}\}\}_{k=1}^{n_b}$
 - 6: **for** $x_{\text{batch}} \in \mathcal{D}$ **do**
 - 7: **for** $l = 1$ to L **do**
 - 8: **for** $t = 1$ to T **do**
 - 9: Update statistics $\{\rho_t^l, g_t^l\}$
 - 10: **end for**
 - 11: **end for**
 - 12: $y_A \leftarrow A^{\text{frozen}}(x_{\text{batch}})$ {Forward pass on ANN}
 - 13: $y_S, \{F_{S_k}\}_{k=1}^{n_b} \leftarrow S(x_{\text{batch}})$ {Forward pass on SNN}
 - 14: **for** $k = 1$ to n_b **do**
 - 15: $y_{M_k} \leftarrow M_k^p(F_{S_k})$ {Forward pass on M_k^p }
 - 16: Compute block loss L_{blk}^k as in Eq. (3)
 - 17: **end for**
 - 18: Compute total loss L_{total} as in Eq. (4)
 - 19: Update SNN model parameters using L_{total}
 - 20: **end for**
-

According to [59], an equivalent form for $\frac{\partial L}{\partial c^l}$ is derived:

$$\frac{\partial L}{\partial c^l} = \left(\frac{\partial L}{\partial c^l} \prod_{i=L-1}^l \left(W^{i\top} \mathbb{E} \left[\frac{\partial s_t^l}{\partial u_t^l} \rho_t^l \right] \right) \right). \quad (14)$$

Here, the statistical term is defined as:

$$\rho_t^l = 1 + \rho_{t-1}^l \left(\frac{\partial u_t^l}{\partial u_{t-1}^l} + \frac{\partial u_t^l}{\partial s_{t-1}^l} \frac{\partial s_{t-1}^l}{\partial u_{t-1}^l} \right). \quad (15)$$

Furthermore, define g_t^l as follows:

$$g_t^l = \frac{1}{t} \left((t-1)g_{t-1}^l + \frac{\partial s_t^l}{\partial u_t^l} \rho_t^l \right). \quad (16)$$

The gradients contained in the statistics ρ_t^l and g_t^l can be computed directly. As demonstrated in [59], the expected value $\mathbb{E} \left[\frac{\partial s_t^l}{\partial u_t^l} \rho_t^l \right]$ can be obtained as:

$$g_T^l = \mathbb{E} \left[\frac{\partial s_T^l}{\partial u_T^l} \rho_T^l \right]. \quad (17)$$

This allows us to derive $\frac{\partial L}{\partial c^l}$ and subsequently perform a single backpropagation to update the weights.

3.5. ANN-guided SNN Distillation Framework

Our entire ANN-guided SNN Distillation Framework is outlined in Algorithm 1. Initially, we define the ANN components of the n_b hybrid models, denoted as $\{M_k^p\}_{k=1}^{n_b}$. For

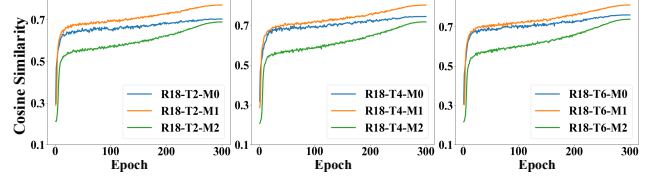


Figure 2. **Measures of feature similarity.** The results of cosine similarity distances are obtained by ResNet-18 on CIFAR-100. Each subplot is labeled according to the naming convention “R18(ResNet-18)-T(timesteps)-M(M_k).”

each mini-batch of data, the ANN performs a forward pass to generate the teacher labels y_A . The SNN then executes T forward passes without recording gradients to update the statistical parameters for backpropagation, $\{\rho_t^l, g_t^l\}$. This is followed by a single forward pass that records gradients to obtain the student labels y_S and the intermediate rate features $\{F_{S_k}\}_{k=1}^{n_b}$.

Next, we conduct the forward pass for each M_k^p to obtain the corresponding outputs y_{M_k} . We first compute the loss for each hybrid model L_{blk}^k and then aggregate these to determine the total loss L_{total} . Finally, we perform backpropagation of the loss, wherein the gradients for the SNN parameters are computed using rate-based backpropagation, leveraging $\{\rho_T^l, g_T^l\}$ to decouple the time dimension during SNN training.

It is worth noting that our approach, in terms of implementation, requires only minimal modifications during the forward propagation phase, and there are no structural constraints on the teacher and student models.

4. Experiments

In this section, we evaluate the effectiveness of the proposed ANN-guided SNN distillation framework, focusing on the accuracy at different time steps for image classification tasks. Sec. 4.2 present a comparative analysis of our approach against existing SNN direct training methods and several established SNN distillation techniques. Sec. 4.3 provides a comprehensive overview of the ablation studies, while Sec. 4.4 analyzes the effectiveness of our proposed approach in conjunction with experimental results.

4.1. Experimental Setup

Dataset. We conduct experiments on three static image classification datasets and one neuromorphic dataset: ImageNet [9], CIFAR-10, CIFAR-100 [26], and CIFAR10-DVS [32]. ImageNet comprises 1.2 million training and 50,000 validation images across 1,000 classes. CIFAR-10 includes 60,000 images of size 32x32 across 10 categories, while CIFAR-100 offers a more challenging task with 60,000 images across 100 categories. CIFAR10-DVS, based on dy-

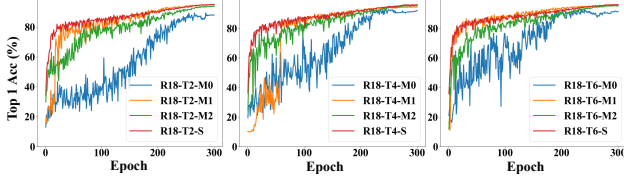


Figure 3. **Validation accuracy of ANN-SNN hybrid models and SNN model during training.** The results are obtained by ResNet-18 on CIFAR-10. Each subplot is labeled according to the naming convention “R18(ResNet-18)-T(timesteps)-M(M_k)/S(SNN).”

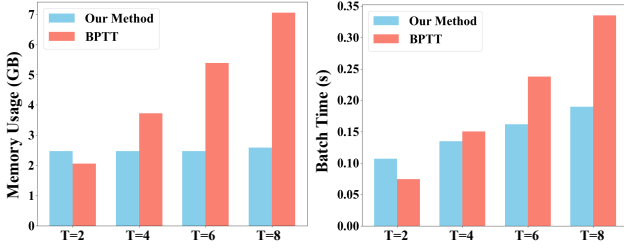


Figure 4. **Comparison of Training overhead between our method and BPTT for direct training.** The results are obtained by averaging over three epochs of stable operation with ResNet-18 on CIFAR-100, using a single NVIDIA 3090 GPU.

namic vision sensors (DVS), contains event streams adapted from CIFAR-10 images, making it suitable for neuromorphic model evaluation.

Training Setting. Our experiments use models from the ResNet family. For CIFAR and CIFAR10-DVS, models are trained for 300 epochs with a learning rate of 0.1, weight decay of $5e-4$, and a batch size of 128. For ImageNet, we use 100 epochs with a 0.2 learning rate, $2e-5$ weight decay, and batch size 512. More detailed training specifics are provided in the supplement.

4.2. Comparison to the State-of-the-Art

Results on CIFAR. We use ResNet-18 and ResNet-19 as student models. For ResNet-18, the teacher model is an ANN of the same structure, achieving accuracies of 96.92% and 79.95% on CIFAR-10 and CIFAR-100, respectively. For ResNet-19, the teacher model is ResNet-34 due to their structural similarity, achieving accuracies of 97.24% and 81.90%. Like ASGL[51], we apply AutoAugment [7] and Cutout [12] for data augmentation. Tab. 1 displays the performance on CIFAR-10 and CIFAR-100 datasets. Compared to other SNN methods based on BPTT and existing SNN distillation techniques, our approach achieves or surpasses state-of-the-art results. For ResNet-18, our method shows significant improvements over Joint A-SNN [18], KDSNN [57] and TSSD [68] across all time steps. For ResNet-19, at $T = 2$, we surpass the accuracy of SAKD

[41] at $T = 4$ and TSSD [68] at $T = 6$.

Results on ImageNet. We employ SEW-ResNet-34 [14] and the commonly used PreAct-ResNet-34 [21] as student models, with the teacher model also being ResNet-34, achieving an accuracy of 76.32%. Additionally, we conduct experiments using ResNet-50 as the teacher model, where the student model, PreAct-ResNet-34, achieved an accuracy of 71.76%. Our experimental results, presented in Tab. 1, demonstrate that our method outperforms existing SNN approaches based on BPTT and previously established SNN distillation methods, achieving state-of-the-art performance. Experiments on static image datasets indicate that our approach effectively transfers knowledge from the ANN pretrained models to the SNN.

Results on CIFAR10-DVS. We use the ResNet-19 as both the student and teacher models on the neuromorphic dataset CIFAR10-DVS. The ANN teacher model is implemented by replacing SNN neurons with ReLU, and its input is the mean over the time dimension, achieving an accuracy of 83.60%. Similar to previous work [33], we split the dataset into training and testing images, with the training set containing 9,000 images and the testing set containing 1,000 images. For data preprocessing and augmentation, we first resize the training images to 48×48 pixels and then randomly flip them horizontally. Additionally, we apply a random rolling operation within a range of 5 pixels. The testing images are directly resized to 48×48 pixels. Tab. 2 shows the performance on CIFAR10-DVS, where our method achieves state of the art with shorter time steps.

4.3. Ablation Study

Ablation study on the position of M_k . To identify optimal insertion points for ANN-guided feature learning, we perform ablation experiments with ResNet-18 on CIFAR-100 (see Tab. 3). Vanilla logit-based knowledge distillation provides only a 0.15% accuracy improvement over direct training, while adding ANN-guided branches further enhances performance. For ResNet-18, we segment the model into four main layers, with a head section preceding them and a classification layer following. We denote the hybrid models inserted after the head as M_H , after each main layer (Layers 1–3) as M_0 through M_2 , and before the classification layer as M_{fc} . Results show that positioning ANN-guided branches at the outputs of main layers achieves the best performance. Omitting any branch in these layers degrades accuracy, underscoring the value of feature alignment across layers. This may stem from distributional differences between ANN and SNN knowledge, as intermediate SNN outputs within main layers appear to align more effectively with ANN feature spaces, thereby enhancing the impact of ANN-guided learning.

Control experiment of hyperparameters α and β_k . We examine different strategies for α and β_k in Tab. 4. For α ,

Table 2. **Performance on CIFAR10-DVS.** Results are averaged over five runs of experiments.

Method	Model	Timesteps	Top-1 Acc (%)
Rollout [27]	DenseNet	10	66.80
LIAF-Net [54]	LIAF-Net	10	71.70
STBP-ttBN [66]	ResNet-19	10	67.80
RecDis-SNN [16]	ResNet19	10	72.42
Real Spike [17]	ResNet-19	10	72.85
Dspike [33]	ResNet-18	10	75.4
GLIF [58]	7B-wideNet	16	78.10
RateBP [59]	VGG-11	10	76.96
ENOF [15]	ResNet19	10	80.10
SAKD [41]	ResNet-19	4	80.30
Ours	ResNet-19	4	80.54±0.71(81.70)

Table 3. **Ablation study of the design of the position of M_k .** The results are obtained using ResNet-18 with $T = 4$ on CIFAR-100, averaged over five runs of experiments.

y_A	y_{M_H}	y_{M_0}	y_{M_1}	y_{M_2}	$y_{M_{fc}}$	Accuracy (%)
						78.26±0.16
✓						78.41±0.27
✓	✓					78.69±0.07
✓		✓				78.70±0.06
✓			✓			78.85±0.19
✓				✓		78.75±0.10
✓		✓		✓		78.72±0.15
✓		✓	✓			78.68±0.17

Table 4. **Control experiment of hyper-parameter α and β_k .** $\uparrow$ represents "increase", and $\downarrow$ represents "decrease".

β_k	$1/2^{n-k}$			$1/n_b$		
α	$\uparrow$	$\downarrow$	0.5	$\uparrow$	$\downarrow$	0.5
Top-1 Acc(%)	78.34±0.11	78.21±0.21	78.29±0.11	78.28±0.16	78.52±0.20	78.85±0.19

three options are tested: an "increase" strategy that linearly interpolates from 0 to 1 over epochs, a "decrease" strategy with the opposite trend, and a fixed value of 0.5, which balances the ANN-guided loss and hard-label loss. For β_k , we evaluate decay by powers of 2 in descending order of k and a uniform assignment of $1/n_b$, where n_b is the number of hybrid models. Results suggest that setting $\alpha = 0.5$ and $\beta_k = 1/n_b$ yields optimal performance.

4.4. Analysis and Discussion

Cosine similarity between intermediate features. Cosine similarity measures the alignment between two vectors by quantifying the cosine of the angle between them, with values approaching 1 indicating closer directional alignment. As shown in Fig. 2, we use this metric to examine the correspondence between the SNN's rate-based intermediate feature space (F_{S_k}) and the ANN's feature space (F_{A_k}) throughout ResNet-18 training at time steps $T = 2, 4,$

and 6. Although no explicit objective is set for minimizing $\|F_{S_k} - F_{A_k}\|$, cosine similarity between F_{S_k} and F_{A_k} increases consistently across training. Among the hybrid models, M_1 shows the highest similarity at all time steps, followed by M_0 , with M_2 showing the lowest. This pattern suggests that intermediate feature mappings in dense backbone layers are more aligned with those of the ANN, while layers nearer the classifier exhibit a task-focused orientation.

Validation accuracy of M_k during training. In the process of ANN-guided SNN training, we record the validation accuracy trends of ResNet-18 and the ANN-SNN hybrid models on CIFAR-10 across different time steps, as shown in Fig. 3. The validation accuracy trends for M_k closely resemble the cosine similarity patterns observed in Fig. 2. Specifically, the validation accuracies of M_1 and M_2 closely align with that of the SNN, while M_0 shows slightly lower accuracy. We interpret the hybrid models M_k as forming a series of unique teacher models, establishing a self-distillation-like framework that effectively guides the learning of the SNN.

Training overhead comparing to BPTT. We compare the training cost of ANN-guided SNN distillation with that of BPTT-based direct training, as shown in Fig. 4. At $T = 2$, our method shows slightly higher batch time and memory usage than BPTT; however, as the time steps increase, our method does not exhibit a linear increase in memory usage and batch time like BPTT. At $T = 4, 6$, and 8, both memory and computation time are significantly lower than BPTT. This efficiency is due in part to rate-based backpropagation, as well as the relatively low number of additional parameters introduced by our method—such as only 0.17M additional parameters for C_k in ResNet-18 and ResNet-34. These results highlight the training efficiency of our approach.

5. Conclusion

In this paper, we proposed a rate-based ANN-guided SNN distillation framework that incorporates pretrained ANN models via a block-wise replacement approach. This method allows for flexible alignment of SNN feature spaces with rate-based representations, achieving effective knowledge transfer. Our experiments demonstrate that the proposed framework achieves state-of-the-art performance, and we hope it provides valuable insights that contribute to further advancements in ANN-guided SNN learning.

Acknowledgments

This work was supported by Zhejiang Provincial Natural Science Foundation of China under Grant No. ZCLZ24F0102, the National Natural Science Foundation of China under Grant No. 62304203, 62401501, 62431012 and 62027805, the National Key Research and Development Program of China under Grant No. 2023YFB3812500.

References

- [1] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34 (10):1537–1557, 2015. 1
- [2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 2
- [3] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 696–697, 2020. 1, 2
- [4] Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, pages 11–20, 2022. 1, 2
- [5] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. *arXiv preprint arXiv:2303.04347*, 2023. 1, 2
- [6] Tong Bu, Maohua Li, and Zhaofei Yu. Training-free conversion of pretrained anns to snns for low-power and high-performance applications. *arXiv preprint arXiv:2409.03368*, 2024. 1
- [7] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 7
- [8] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018. 1
- [9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6
- [10] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022. 5
- [11] Shikuang Deng, Hao Lin, Yuhang Li, and Shi Gu. Surrogate module learning: Reduce the gradient error accumulation in training spiking neural networks. In *International Conference on Machine Learning*, pages 7645–7657. PMLR, 2023. 1, 2
- [12] Terrance DeVries. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 7
- [13] Rida El-Allami, Alberto Marchisio, Muhammad Shafique, and Ihcen Alouani. Securing deep spiking neural networks against adversarial attacks through inherent structural parameters. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 774–779. IEEE, 2021. 1
- [14] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021. 5, 7
- [15] Yufei Guo, Weihang Peng, Xiaode Liu, Yuanpei Chen, Yuhang Zhang, Xin Tong, Zhou Jie, and Zhe Ma. Enof-snn: Training accurate spiking neural networks via enhancing the output feature. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 5, 8
- [16] Yufei Guo, Xinyi Tong, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Zhe Ma, and Xuhui Huang. Rectdis-snn: Rectifying membrane potential distribution for directly training spiking neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 326–335, 2022. 5, 8
- [17] Yufei Guo, Liwen Zhang, Yuanpei Chen, Xinyi Tong, Xiaode Liu, YingLei Wang, Xuhui Huang, and Zhe Ma. Real spike: Learning real-valued spikes for spiking neural networks. In *European Conference on Computer Vision*, pages 52–68. Springer, 2022. 5, 8
- [18] Yufei Guo, Weihang Peng, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Xuhui Huang, and Zhe Ma. Joint a-snn: Joint training of artificial and spiking neural networks via self-distillation and weight factorization. *Pattern Recognition*, 142:109639, 2023. 1, 2, 5, 7
- [19] Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2
- [20] Di Hong, Jiangrong Shen, Yu Qi, and Yueming Wang. Lasnn: Layer-wise ann-to-snn distillation for effective and efficient training in deep spiking neural networks. *arXiv preprint arXiv:2304.09101*, 2023. 2, 5
- [21] Yifan Hu, Yujie Wu, Lei Deng, and Guoqi Li. Advancing residual learning towards powerful deep spiking neural networks. *arXiv preprint arXiv:2112.08954*, 7, 2021. 7
- [22] Yangfan Hu, Qian Zheng, Xudong Jiang, and Gang Pan. Fast-snn: fast spiking neural network by converting quantized ann. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 1, 2
- [23] Eric Hunsberger and Chris Eliasmith. Spiking deep networks with lif neurons. *arXiv preprint arXiv:1510.08829*, 2015. 1
- [24] Haiyan Jiang, Srinivas Anumasa, Giulia De Masi, Huan Xiong, and Bin Gu. A unified optimization framework of ann-snn conversion: towards optimal mapping from activation values to firing rates. In *International Conference on Machine Learning*, pages 14945–14974. PMLR, 2023. 1, 2
- [25] Jinseok Kim, Kyungsu Kim, and Jae-Joon Kim. Unifying activation-and timing-based learning rules for spiking neural networks. *Advances in Neural Information Processing Systems*, 33:19534–19544, 2020. 1, 2
- [26] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6
- [27] Alexander Kugele, Thomas Pfeil, Michael Pfeiffer, and Elisabetta Chicca. Efficient processing of spatio-temporal data streams with spiking neural networks. *Frontiers in neuroscience*, 14:512192, 2020. 8

- [28] Souvik Kundu, Massoud Pedram, and Peter A Beerel. Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5209–5218, 2021. 1
- [29] Ravi Kumar Kushawaha, Saurabh Kumar, Biplab Banerjee, and Rajbabu Velmurugan. Distilling spikes: Knowledge distillation in spiking neural networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4536–4543. IEEE, 2021. 2
- [30] Dongjin Lee, Seongsik Park, Jongwan Kim, Wuhyeon Doh, and Sungroh Yoon. Energy-efficient knowledge distillation for spiking neural networks. *arXiv preprint arXiv:2106.07172*, 2021. 2
- [31] Chen Li, Lei Ma, and Steve Furber. Quantization framework for fast spiking neural networks. *Frontiers in Neuroscience*, 16:918793, 2022. 1, 2
- [32] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017. 6
- [33] Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuan Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34: 23426–23439, 2021. 1, 2, 5, 7, 8
- [34] Yuhang Li, Youngeun Kim, Hyoungseob Park, and Priyadarshini Panda. Uncovering the representation of spiking neural networks trained with surrogate gradient. *arXiv preprint arXiv:2304.13098*, 2023. 1
- [35] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2604–2613, 2019. 2
- [36] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997. 1
- [37] Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. 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*, pages 12444–12453, 2022. 1, 2
- [38] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. 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. 1
- [39] Stefano Panzeri and Simon R Schultz. A unified approach to the study of temporal, correlational, and rate coding. *Neural Computation*, 13(6):1311–1349, 2001. 1
- [40] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019. 1
- [41] Haonan Qiu, Munan Ning, Zeyin Song, Wei Fang, Yanqi Chen, Tao Sun, Zhengyu Ma, Li Yuan, and Yonghong Tian. Self-architectural knowledge distillation for spiking neural networks. *Neural Networks*, page 106475, 2024. 1, 2, 5, 7, 8
- [42] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019. 1
- [43] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017. 1
- [44] Saima Sharmin, Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 399–414. Springer, 2020. 1
- [45] Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31, 2018. 1
- [46] Qiaoyi Su, Shijie Mei, Xingrun Xing, Man Yao, Jiajun Zhang, Bo Xu, and Guoqi Li. Snn-bert: Training-efficient spiking neural networks for energy-efficient bert. *Neural Networks*, 180:106630, 2024. 2
- [47] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019. 2
- [48] Sugahara Takuya, Renyuan Zhang, and Yasuhiko Nakashima. Training low-latency spiking neural network through knowledge distillation. In *2021 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*, pages 1–3. IEEE, 2021. 2
- [49] Thi Diem Tran, Kien Trung Le, and An Luong Truong Nguyen. Training low-latency deep spiking neural networks with knowledge distillation and batch normalization through time. In *2022 5th International Conference on Computational Intelligence and Networks (CINE)*, pages 01–06. IEEE, 2022. 2
- [50] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE transactions on pattern analysis and machine intelligence*, 44(6):3048–3068, 2021. 2
- [51] Ziming Wang, Runhao Jiang, Shuang Lian, Rui Yan, and Huajin Tang. Adaptive smoothing gradient learning for spiking neural networks. In *International Conference on Machine Learning*, pages 35798–35816. PMLR, 2023. 7
- [52] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989. 2
- [53] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018. 1
- [54] Zhenzhi Wu, Hehui Zhang, Yihan Lin, Guoqi Li, Meng Wang, and Ye Tang. Liaf-net: Leaky integrate and analog fire

- network for lightweight and efficient spatiotemporal information processing. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6249–6262, 2021. 8
- [55] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Yisen Wang, and Zhouchen Lin. Training feedback spiking neural networks by implicit differentiation on the equilibrium state. *Advances in neural information processing systems*, 34:14516–14528, 2021. 1, 2
- [56] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. *Advances in neural information processing systems*, 35:20717–20730, 2022. 1, 2, 5
- [57] Qi Xu, Yaxin Li, Jiangrong Shen, Jian K Liu, Huajin Tang, and Gang Pan. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7886–7895, 2023. 1, 2, 5, 7
- [58] Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:32160–32171, 2022. 5, 8
- [59] Chengting Yu, Lei Liu, Gaoang Wang, Erping Li, and Aili Wang. Advancing training efficiency of deep spiking neural networks through rate-based backpropagation. *arXiv preprint arXiv:2410.11488*, 2024. 1, 2, 5, 6, 8
- [60] Chengting Yu, Shu Yang, Fengzhao Zhang, Hanzhi Ma, Aili Wang, and Er-Ping Li. Improving quantization-aware training of low-precision network via block replacement on full-precision counterpart. *arXiv preprint arXiv:2412.15846*, 2024. 1, 2
- [61] Chengting Yu, Fengzhao Zhang, Ruizhe Chen, Aili Wang, Zuozhu Liu, Shurun Tan, and Er-Ping Li. Decoupling dark knowledge via block-wise logit distillation for feature-level alignment. *IEEE Transactions on Artificial Intelligence*, 2024. 2
- [62] Chengting Yu, Xiaochen Zhao, Lei Liu, Shu Yang, Gaoang Wang, Erping Li, and Aili Wang. Efficient distillation of deep spiking neural networks for full-range timestep deployment. *arXiv preprint arXiv:2501.15925*, 2025. 2
- [63] Fengzhao Zhang, Chengting Yu, Hanzhi Ma, Zheming Gu, and Er-ping Li. Knowledge distillation for spiking neural network. In *2023 5th International Conference on Robotics, Intelligent Control and Artificial Intelligence (RICAI)*, pages 1015–1020. IEEE, 2023. 2
- [64] Qian Zhang, Chao Ge, Yansong Chua, Chenxiao Dou, and Jibin Wu. Supersnn: Training spiking neural networks with knowledge from artificial neural networks. 1, 2, 5
- [65] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Advances in neural information processing systems*, 33:12022–12033, 2020. 1, 2
- [66] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, pages 11062–11070, 2021. 5, 8
- [67] Yaoyu Zhu, Jianhao Ding, Tiejun Huang, Xiaodong Xie, and Zhaofei Yu. Online stabilization of spiking neural networks. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 2
- [68] Lin Zuo, Yongqi Ding, Mengmeng Jing, Kunshan Yang, and Yunqian Yu. Self-distillation learning based on temporal-spatial consistency for spiking neural networks. *arXiv preprint arXiv:2406.07862*, 2024. 1, 5, 7