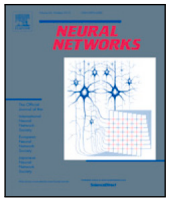
Neural Networks 178 (2024) 106475

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



Full Length Article

Self-architectural knowledge distillation for spiking neural networks

Haonan Qiu [a,1], Munan Ning [a,1], Zeyin Song [a], Wei Fang [b,c], Yanqi Chen [b,c], Tao Sun [a], Zhengyu Ma [c,*], Li Yuan [a,c,*], Yonghong Tian [a,b,c,*]

[a] Peking University, School of Electronic and Computer Engineering, Shenzhen Graduate School, China
[b] Peking University, School of Computer Science, China
[c] PengCheng Laboratory, China

ARTICLE INFO

Dataset link: https://github.com/Maybe2022/SAKD

Keywords:
Spiking neural networks
ANN-to-SNN
Knowledge distillation
Image classification
Semantic segmentation

ABSTRACT

Spiking neural networks (SNNs) have attracted attention due to their biological plausibility and the potential for low-energy applications on neuromorphic hardware. Two mainstream approaches are commonly used to obtain SNNs, *i.e.*, ANN-to-SNN conversion methods, and Directly-trained-SNN methods. However, the former achieve excellent performance at the cost of a large number of time steps (*i.e.*, latency), while the latter exhibit lower latency but suffers from suboptimal performance. To tackle the performance-latency trade-off, we propose Self-Architectural Knowledge Distillation (SAKD), an intuitive and effective method for SNNs leveraging Knowledge Distillation (KD). We adopt a bilevel teacher–student training strategy in SAKD, *i.e.*, level-1 involves directly transferring same-architectural pre-trained ANN weights to SNNs, and level-2 encourages the SNNs to mimic ANN's behavior, considering both final responses and intermediate features aspects. Learning with informative supervision signals fostered by labels and ANNs, our SAKD achieves new state-of-the-art (SOTA) performance with a few time steps on widely-used classification benchmark datasets. On ImageNet-1K, with only 4 time steps, our Spiking-ResNet34 model attains a Top-1 accuracy of 70.04%, outperforming the previous same-architectural SOTA methods. Notably, our SEW-ResNet152 model reaches a Top-1 accuracy of 77.30% on ImageNet-1K, setting a new SOTA benchmark for SNNs. Furthermore, we apply our SAKD to various dense prediction downstream tasks, such as object detection and semantic segmentation, demonstrating strong generalization ability and superior performance. In conclusion, our proposed SAKD framework presents a promising approach for achieving both high performance and low latency in SNNs, potentially paving the way for future advancements in the field.

1. Introduction

Artificial neural networks (ANNs) have achieved remarkable success in the last decade, leading to breakthroughs in various pattern recognition tasks. Recently, there has been growing interest in spiking neural networks (SNNs), the third-generation neural networks that mimic biological neurons and spike-firing patterns (Gerstner, Kistler, Naud, & Paninski, 2014; Maass, 1997; Roy, Jaiswal, & Panda, 2019). Due to their biological plausibility, low-energy computing, and compatibility with neuromorphic hardware, SNNs have wide application prospects, especially with neuromorphic hardware development.

However, obtaining high-performance SNNs with low latency remains a challenge. Current methods are mainly divided into two categories. The first is the *ANN-to-SNN conversion methods*, as shown in Fig. 1(a), which directly transfers pre-trained ANNs' weights to SNNs and then replaces ANNs' activation functions with spiking neurons (*e.g.* LIF Neuron in Eq. (1)) (Bu et al., 2021; Cao, Chen, & Khosla, 2015; Han, Srinivasan, & Roy, 2020; Hu, Zheng, Jiang, & Pan, 2023; Hunsberger & Eliasmith, 2015; Li, Deng, Dong, Gong, & Gu, 2021; Rueckauer, Lungu, Hu, Pfeiffer, & Liu, 2017; Wang et al., 2022; Yan, Zhou, & Wong, 2023). This method can be regarded as straightforward teacher–student training, where SNNs can easily achieve comparable performance with their ANNs teachers. Nevertheless, these methods obtain SNNs through direct weight duplication without temporal dynamic training under label supervision, which usually requires a large number of time-steps to reach a reliable firing rate approximation (e.g., 256 steps in Bu et al. (2021), Li et al. (2021)).

The second category is *Directly-train-SNN methods*, as shown in Fig. 1(b), which train SNNs directly with gradient backpropagation algorithm (Meng et al., 2022; Neftci, Mostafa, & Zenke, 2019). These studies use various techniques to enable the training of SNNs with

* Corresponding authors.
E-mail addresses: qiuhaonan@stu.pku.edu.cn (H. Qiu), mazhy@pcl.ac.cn (Z. Ma), yuanli-ece@pku.edu.cn (L. Yuan), yhtian@pku.edu.cn (Y. Tian).
[1] These authors contributed equally to this work

https://doi.org/10.1016/j.neunet.2024.106475
Received 7 November 2023; Received in revised form 16 May 2024; Accepted 17 June 2024
Available online 19 June 2024

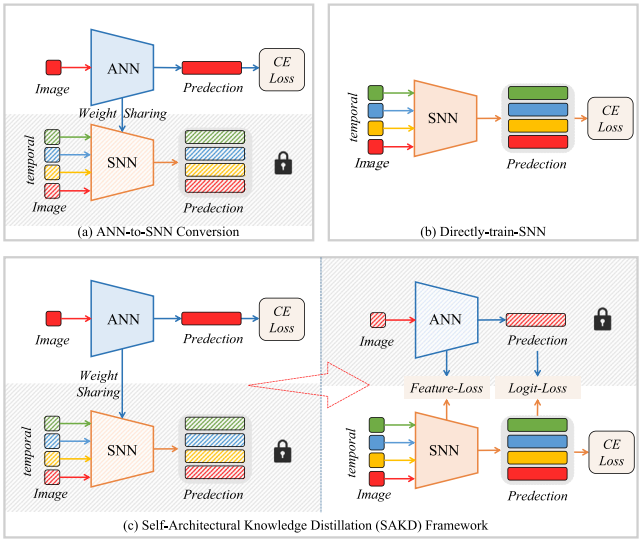


Fig. 1. An illustration of existing mainstream SNNs training methods. **(a)**: Convert pre-trained ANNs to SNNs (ANN-to-SNN). **(b)**: Gradient backpropagation training SNNs (Directly-train-SNN). **(c)**: Our proposed method (SAKD): knowledge of ANNs transferred to SNNs of the same architecture through weight, features, and logits distillation. Gray background denotes that weights do not need to be updated during training.

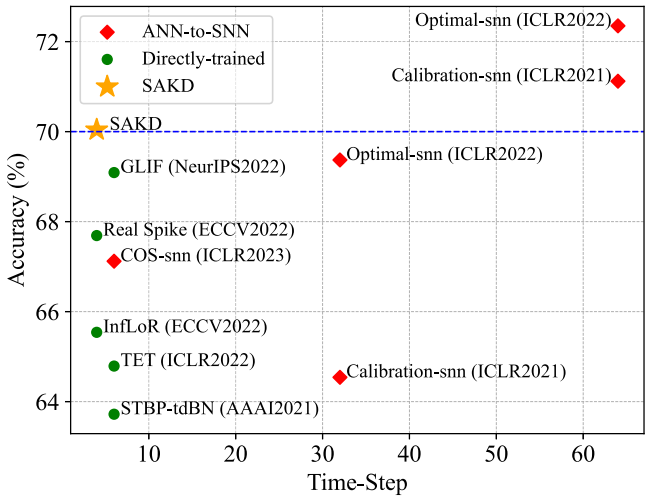


Fig. 2. Comparisons with state-of-the-art methods on ImageNet-1K on Spiking-ResNet34. Points closer to the left indicate lower latency, while those closer to the top indicate better performance. Our method effectively addresses the trade-off between performance and latency in SNNs, achieving both good performance and low latency. For more results on ImageNet, please refer to Table 2.

backpropagation, such as surrogate gradient (SG) and backpropagation through time (BPTT). These methods have significantly improved SNNs' performance at low timesteps. However, while their performance surpasses that of ANN-to-SNN methods at low timesteps, they still cannot achieve satisfactory results, leaving a considerable gap compared to the peak performance of ANN-to-SNN methods at high timesteps. Several potential reasons (*e.g.*, SG cannot reflect the exact gradient, BPTT may lead to the accumulation of gradients over time, resulting in unstable training (Fang, Yu, Chen, Huang, et al., 2021)) make it challenging for conventional single-label supervised learning paradigms to supply sufficient knowledge (supervisory signals) for SNN training.

In this paper, we strike a balance between performance and latency by proposing an intuitive but effective method called the **Self-Architectural Knowledge Distillation (SAKD)** framework, as shown in Fig. 1(c). This approach combines the strengths of the two methods to achieve both high performance and low latency for SNNs. Specifically, we make full use of the pre-trained ANNs, and follow a bilevel teacher–student training regime in SAKD: (i) directly inheriting the weights of pre-trained ANNs with the same architecture to initialize the target SNNs without training; (ii) learning to mimic the intermediate features and logits of ANNs during training. Moreover, we propose a simple module to eliminate the inherent differences between SNNs and ANNs' intermediate features. We first flatten the SNNs features over the time step dimensions, then project SNNs' discrete features into ANNs' continuous space. This mitigates the differences in feature dimensionality (*i.e.*, SNNs have an extra temporal dimension) and space (*i.e.*, binary values and floating-point values) between SNNs and ANNs, resulting in improved knowledge transfer.

Here are some related works that also attempt to introduce ANNs' knowledge into the SNN training process. This includes feature-based methods like (Hong, Shen, Qi, & Wang, 2023; Kundu, Datta, Pedram, & Beerel, 2021; Li et al., 2021; Xu, Li, Shen, et al., 2023; Yang, Wu, et al., 2022), logit-based methods like (Dong, Zhao, & Zeng, 2023; Kushawaha, Kumar, Banerjee, & Velmurugan, 2021; Takuya, Zhang, & Nakashima, 2021; Xu, Li, Fang, et al., 2023; Xu, Li, Shen, et al., 2023) and weight-based methods like (Chowdhury, Rathi, & Roy, 2021; Datta & Beerel, 2022; Datta, Liu, Diffenderfer, Kailkhura, & Beerel, 2023; Hao, Bu, Ding, Huang, & Yu, 2023; Hu et al., 2023; Li et al., 2021; Rathi & Roy, 2021; Tang, Lai, Xie, Yang, & Zheng, 2023). However, our SAKD framework differs from previous similar works in the following ways: (i) While previous works have mainly focused on compressing SNNs using KD, we aim to push the limits of low-latency SNNs. (ii) Previous works have only exploited a portion of the knowledge from ANNs, whereas we have made full use of ANN knowledge and explored optimal implementation strategies. (iii)Previous works have not evaluated performance on large datasets or more challenging tasks, whereas we extended our approach to ImageNet and explored its effectiveness on various tasks.

Our experiments demonstrate that our proposed SAKD framework effectively addresses the trade-off between performance and latency in SNNs, even for high-level semantic tasks and complex dense prediction tasks. Notably, on ImageNet-1K, with only four time steps, our Spiking-ResNet34 model attains a Top-1 accuracy of 70.04%, outperforming the previous SOTA method for the same model by 2.35%, as shown in Fig. 2. The SEW-ResNet50 model achieves a Top-1 accuracy of 74.70%, surpassing the previous SOTA method for the same model by 4%. Furthermore, our SEW-ResNet152 model reaches a Top-1 accuracy of 77.30%, exceeding the original paper's accuracy by 8.04% and setting a new SOTA benchmark for SNNs. In addition to the classification task, we also extend our framework to semantic segmentation and object detection tasks, delivering an average performance improvement of 5%–15% compared to baseline SNN models. These results highlight the effectiveness and generality of our SAKD framework for training high-performance, low-latency SNNs, with potential applications across a broad range of domains.

Our main contributions are as follows:

- We propose the Self-Architectural Knowledge Distillation (SAKD) framework, which combines the advantages of both ANN-to-SNN conversion and Directly-trained-SNN methods to achieve a trade-off between the performance and latency, enabling high-performance and low-latency SNNs. The framework transfers knowledge from pre-trained ANNs to SNNs with identical architectures through a bilevel method, which enhances the training and optimization efficiency of SNNs.
- We investigate the optimal implementation of knowledge transfer from ANNs to SNNs, including how to better achieve feature distillation and combine it with logit distillation to improve the learning process. This ensures better adaptation of the transferred weights to the temporal dynamics of SNNs.

- Our SAKD framework achieves new SOTA performance with few time steps on various benchmark datasets and tasks, including CIFAR (Krizhevsky, Nair, & Hinton, 2014), DVS-CIFAR10 (Li, Liu, Ji, Li, & Shi, 2017), ImageNet-1K (Deng et al., 2009), PASCAL VOC 2012 (Everingham, Van Gool, Williams, Winn, & Zisserman, 2010) and Cityscapes (Cordts et al., 2016) for classification, object detection, and semantic segmentation. This demonstrates the strong generalization ability, superior performance, and potential to advance SNNs in various applications.

2. Related work

Learning Methods of Spiking Neural Networks. Inspired by the breakthrough success of ANNs in various tasks, researchers have sought to introduce techniques from ANNs to improve SNNs. Among them, *ANN-to-SNN conversion* (Cao et al., 2015; Deng & Gu, 2021; Diehl et al., 2015; Ding, Yu, Tian & Huang, 2021; Han & Roy, 2020; Hunsberger & Eliasmith, 2015; Kim, Park, Na, & Yoon, 2020; Li et al., 2021; Rueckauer et al., 2017; Sengupta, Ye, Wang, Liu, & Roy, 2019; Stöckl & Maass, 2021; Yan, Zhou, & Wong, 2021) and *Directly-train-SNN* (Bellec, Salaj, Subramoney, Legenstein, & Maass, 2018; Bohte, Kok, & La Poutré, 2000; Esser, Appuswamy, Merolla, Arthur, & Modha, 2015; Fang, Yu, Chen, Huang, et al., 2021; Guo, Zhang, et al., 2022; Huh & Sejnowski, 2018; Kim, Li, Park, Venkatesha, & Panda, 2022; Lee, Delbruck, & Pfeiffer, 2016; Li, Kim, Park, Geller, & Panda, 2022; Mostafa, 2017; Neftci et al., 2019; Shrestha & Orchard, 2018; Wu, Deng, Li, Zhu, & Shi, 2018; Wu et al., 2019; Xiao, Meng, Zhang, Wang, & Lin, 2021; Yang, Zhang, & Li, 2021; Yao, Li, Mo, & Cheng, 2022; Zhang & Li, 2020; Zheng, Wu, Deng, Hu, & Li, 2021) are the two mainstream training schemes. The conversion method obtains high-performance SNNs by replacing the activation function of pre-trained ANNs with the spike activation function and sharing the weights of ANNs. However, while there is no overhead in training SNNs, this method requires a large number of time steps to match the spike firing frequency with the activation value of ANNs, which limits the performance of SNNs under low latency and impedes their application deployment. In contrast to the conversion method, which does not require training SNNs, the Directly-train-SNN approach employs gradient backpropagation to train SNNs. Recent studies have trained SNNs by solving the non-differentiable problem of the spike activation function in various ways. Among them, the surrogate gradient (SG) method for training SNNs using BPTT can achieve ultra-low latency (Deng, Li, Zhang, & Gu, 2022; Fang, Yu, Chen, Masquelier, et al., 2021). However, the most significant limitation of SG is that it can only achieve suboptimal performance. Some works (Kundu et al., 2021; Rathi & Roy, 2021; Rathi, Srinivasan, Panda, & Roy, 2020) combine the advantages of the two methods to train high-performance SNNs, motivating us to further utilize ANN knowledge to improve SNNs' performance.

Knowledge Distillation. Knowledge distillation (Hinton, Vinyals, Dean, et al., 2015) (KD) is a model compression and knowledge transfer method adopted by many works. Feature distillation was first proposed by Romero et al. (2014) through feature transfer, and has since been improved by a series of works (Chen, Liu, Zhao, & Jia, 2021; Heo et al., 2019; Yang, Li, et al., 2022). Some studies (Li, Yu, et al., 2022; Touvron et al., 2021) have taken this approach a step further by incorporating knowledge from Convolutional Neural Networks to help the training process of Vision Transformers, showing that knowledge could also be transferred across different architectures. Recent research has introduced KD to the field of SNNs (Dong et al., 2023; Hong et al., 2023; Kundu et al., 2021; Kushawaha et al., 2021; Rathi & Roy, 2021; Takuya et al., 2021; Xu, Li, Fang, et al., 2023; Xu, Li, Shen, et al., 2023, 2023; Yang, Wu, et al., 2022), demonstrating that SNNs can learn from ANNs despite their different properties. However, these methods still have some limitations, such as only partially utilizing ANN knowledge, which results in SNNs' performance not being fully improved. Additionally, their methods lack evaluations on large-scale datasets and more challenging tasks, which are needed to demonstrate the generalizability of their approach. These limitations have inspired our work to push the limits of low-latency SNNs even further.

3. Method

3.1. Spiking neuron model

The spiking neuron is the core of the SNNs, integrating inputs, adjusting membrane potentials, and controlling spike firing. In this paper, we select LIF neurons as the basic computational unit of SNNs. The neuron adjusts the membrane potentials based on the current input:

$$U_t^{pre} = \lambda * U_{t-1} + X_t, \tag{1}$$

where λ represents the decay factor, U_t^{pre} and U_t ($U_{-1} = 0$) denote the membrane potential of the neuron before and after firing a spike at time-step t, respectively, and X_t corresponds to the input for the current layer at time-step t. Next, the neuron decides whether to fire a spike based on the current membrane potential and threshold:

$$S_t = \Theta(U_t^{pre} - V_{th}), \tag{2}$$

where S_t denotes the spike output of the neuron at time-step t (the current layer's output). Θ represents the Heaviside step function, defined by $U_t^{pre} - V_{th} \geq 0 \ : \ S_t = 1$, $U_t^{pre} - V_{th} < 0 \ : \ S_t = 0$. When membrane potentials at the time-step t exceed a threshold, the neuron fires a spike, and the membrane potentials adjust simultaneously. We adjust membrane potentials post-spikes according to the hard reset mechanism (Ledinauskas, Ruseckas, Juršėnas, & Buračas, 2020):

$$U_t = U_t^{pre} * (1 - S_t). \tag{3}$$

Since the step function is non-differentiable, we use surrogate gradients as replacements. We select the triangle surrogate gradient, as in Deng et al. (2022), Rathi and Roy (2021). The gradient of its backward is:

$$\Theta'(x) = \max(0, 1 - |x|). \tag{4}$$

3.2. Teacher–student paradigm

We implement a teacher–student framework to achieve a tradeoff between the performance and latency of SNNs, in which the ANN serves as the teacher and the SNN as the student. To fully utilize ANNs' knowledge to train SNNs, we adopt a bilevel method: at level-1, similar to ANN-to-SNN, we directly inherit the pre-trained weights of ANNs to obtain a favorable initial state for SNNs, termed direct knowledge transfer. It can be described as:

$$\mathcal{W}_{SNN}^{init} = \mathcal{W}_{ANN}, \tag{5}$$

where $\mathcal{W}_{SNN}^{init}$ and $\mathcal{W}_{ANN}$ represent the initial weight of SNNs and the pre-trained weight of ANNs, respectively, not including the spiking neuron parameters of SNNs.

At level-2, analogous to direct training of SNNs, we train SNNs through BPTT under low-latency conditions while learning the features and logits of ANNs to indirectly transfer ANN's knowledge and improve convergence performance. The overall loss can be expressed as follows:

$$\mathcal{L}_{all} = \alpha * \mathcal{L}_{ce} + \beta * \mathcal{L}_{feaKD} + \gamma * \mathcal{L}_{logKD}, \tag{6}$$

where $\mathcal{L}_{ce}$ denotes the typical cross-entropy loss, assisting the model in learning from the image-label pair, $\mathcal{L}_{feaKD}$ and $\mathcal{L}_{logKD}$ represent the loss of features and logits from ANNs and SNNs, respectively, and α, β, and γ are the hyperparameters controlling the weight of different loss.

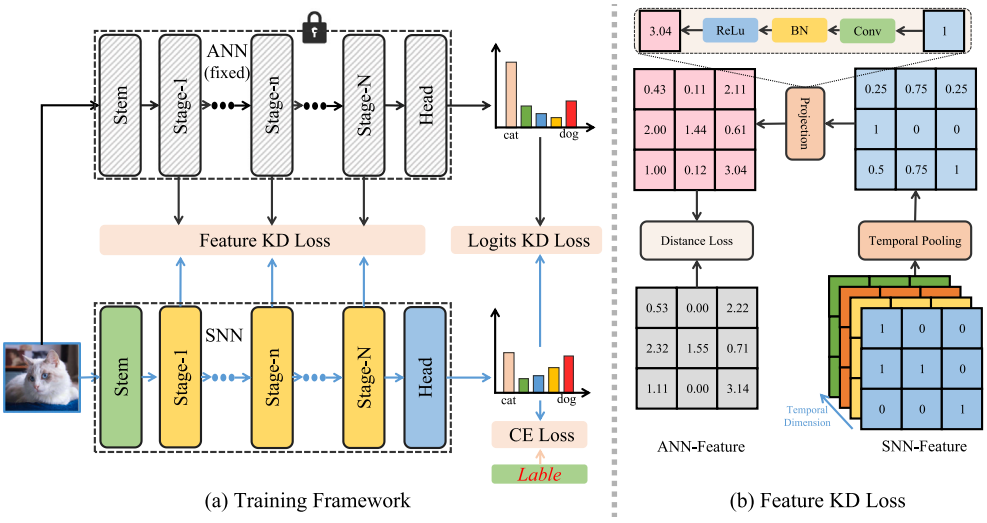


Fig. 3. An illustration of the proposed method, i.e., the knowledge distillations for SNNs. We forced SNNs to learn the feature and logit distribution of pre-trained ANNs with the same architecture. As SNNs and ANNs features are in discrete and continuous space, we use a projection module to map SNNs features to continuous space, matching ANNs features.

3.3. Self-architectural knowledge distillation

The choice of the structure of ANNs and SNNs, as well as efficient knowledge transfer from ANNs to SNNs, is critical for successful SNN knowledge distillation. In our approach, we used a bilevel method, where the first step involves directly inheriting the weights from the ANNs. To achieve this, we employed the same architecture for ANNs to teach SNNs (*i.e.*, self-architectural strategy). Additionally, using the same architecture for ANNs and SNNs has several advantages, such as saving time in selecting a teacher, facilitating deployment, and having similar learning capacities. Moreover, our experimental results demonstrate that choosing the same architecture is indeed the best approach, as shown in Table 9.

In this section, we discuss in detail how to efficiently transfer the knowledge of ANNs into SNNs with the same architecture, as part of our bilevel approach to knowledge distillation. As shown in Fig. 3(a), our approach consists of two modules: (1) logits distillation, where the logits of ANNs constrain the logits in SNNs; (2) feature distillation, where the features of intermediate layers in SNNs are aligned with the corresponding features in ANNs. Meanwhile, as shown in Fig. 3(b), we adopted a series of feature transformations to eliminate the inherent gap between SNN and ANN features, enabling better knowledge transfer from ANNs to SNNs.

Logits Knowledge Distillation. We first introduce the logit knowledge distillation by encouraging the SNN students to learn the predicted distribution of the ANN teachers. As shown in Fig. 3(a), the predicted distribution of the teacher is more informative than the one-hot label, which contains "dark knowledge", *i.e.*, the information of correlation between different categories. We utilize the KL-divergence to enhance the student SNNs by outputting a similar category distribution with the teacher ANNs as follows:

$$\mathcal{L}_{logKD} = \tau^2 \sum p_\tau^t \log(\frac{p_\tau^t}{p_\tau^s}), \tag{7}$$

$$p_\tau^s(i) = \frac{\exp(p^s(i)/\tau)}{\sum \exp(p^s/\tau)}, \tag{8}$$

where p_τ^t and p_τ^s represent the predicted distributions of ANNs and SNNs, respectively. τ is the temperature to smooth the logits.

Features Knowledge Distillation. We further introduce feature knowledge distillation, which transfers the teacher's knowledge by forcing the student to imitate the teacher's feature maps. It can be described as:

$$\mathcal{L}_{feaKD} = \left\| F_s - F_t \right\|^2, \tag{9}$$

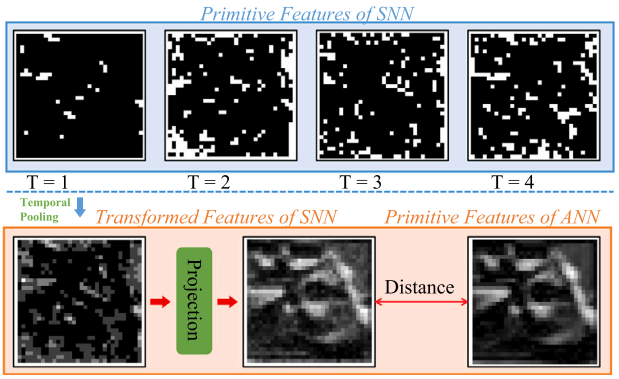


Fig. 4. A perspective of feature distillation. We project sparse discrete features of SNNs to continuous features of ANNs and calculate L_2 distance.

where F_t and F_s represent the predicted distributions of ANNs and SNNs, respectively.

Due to the teacher and the student having the same architecture, we choose to directly match the original features of the ANNs instead of indirectly designing new knowledge forms (Ahn, Hu, Damianou, Lawrence, & Dai, 2019; Tung & Mori, 2019) to transfer knowledge. Inspired by recent studies (Heo et al., 2019), we investigated the various factors that influence the efficiency of feature distillation, including feature transformation, distillation position, and distance function.

❶ ***Feature Transformation*** typically plays a vital role in feature-based distillation methods, addressing feature dimension and space mismatch during teacher–student training.

In this work, feature transformation is essential for effective knowledge transfer between ANNs and SNNs. Although ANNs and SNNs share identical architectures, a natural gap exists between their features. This gap can be attributed to two factors: (1) dimension mismatch, where self-architecture ensures consistency in spatial and channel dimensions, but SNNs inherently have an additional temporal dimension; and (2) feature space mismatch, where SNNs' features are binary-valued and temporal, while ANNs' features are continuous-valued and non-temporal. Hence, direct alignment of these features using Eq. (9) is unfeasible.

To address these challenges, we propose a two-step solution, as shown in the right column of Fig. 3. First, we apply temporal average

pooling (Temporal Pooling) to aggregate the features over the temporal dimension, ensuring consistent feature size between ANNs and SNNs. Second, we employ an additional projection module (*e.g.*, a multilayer perceptron (MLP)), which maps SNNs' features to the same content space as ANNs' features. This approach ensures more effective feature knowledge transfer between the two types of networks and can be formulated as follows:

$$\hat{F}_s = T_s(F_s) = \text{Projection}\left(\frac{1}{T}\sum_{t=1}^{T} F_s^{(t)}\right), \tag{10}$$

where Projection is the MLP in Fig. 3, specifically consisting of a 1×1 Convolution, followed by a BatchNorm layer and a ReLU activation function. The projection network is learnable, and its parameters are variable during the distillation process. Importantly, this projection network exists only during the training phase and will be discarded after training, thus not adding to the complexity or parameter of the SNNs. We do not transform ANN's features to preserve the original information.

Our experiments in Fig. 7 demonstrate the importance of adopting the projection module. As illustrated in Fig. 4, this method acknowledges that we cannot make the spiking features of SNNs identical to the continuous features of ANNs. However, if the spiking features of SNNs can project features similar to those of ANNs, it also means that SNNs can effectively encode high-quality spiking information.

We default to the projection module as feature transform.

❷ ***Distillation Position*** directly affects the effectiveness of feature distillation. Most feature distillation positions often involve selecting the end of each stage (either before or after the activation function), specific layers, or the final layer (Yue, Deng, & Zhou, 2020; Zagoruyko & Komodakis, 2016). In our proposed SAKD paradigm, since the structure of ANNs and SNNs is the same, the distillation location can be chosen from any layer of the network (*e.g.*, end of each stage or each block) without considering the feature dimension mismatch. However, competition between losses from multiple branches might negatively affect performance (Romero et al., 2014; Xu et al., 2022), indicating that it is not necessary to distill every layer. However, in special cases, for particularly deep SNNs (*e.g.*, Spiking-ResNet-101/152), it becomes challenging to backpropagate the gradient effectively, resulting in convergence difficulties. From a gradient perspective, choosing block-by-block distillation optimizes each block (*i.e.*, more layers than each stage) of the network and helps converge, as shown in Table 6.

For simplicity, we default to distilling features at the end of each stage and after the activation function.

4. Experiments

In this section, we systematically present our experimental approach, which thoroughly evaluates the efficiency of our proposed SAKD framework across various tasks, datasets, and architectures. We first focus on the widely-adopted image classification task, verifying the effectiveness of our method on multiple datasets, including CIFAR, DVS-CIFAR10, and ImageNet-1K. Subsequently, we extend our framework to tackle more challenging dense prediction tasks, such as semantic segmentation and object detection, on datasets like PASCAL VOC and Cityscape. This comprehensive experimentation demonstrates the strong generalization ability of our SAKD framework, which can be effectively applied to a diverse range of tasks. Our code is based on SpikingJelly (Fang et al., 2023),[2] an open-source SNNs framework, and we set the neuron parameters $\lambda = 0.5$ and $V_{th} = 1$ in all experiments. [We fix the time step as 4 in experiments by default, both during training and testing phases.] In most of our experiments, the SNN models are based on the Spiking-ResNet series, which we abbreviate as ResNet for simplicity.

[2] https://github.com/fangwei123456/spikingjelly

5. Classification

In this section, we evaluate the effectiveness of our proposed method for mainstream image classification tasks in the SNNs domain. We first outline the experimental details, followed by a comparison of our method with existing state-of-the-art methods. Finally, we extend our method to SEW-ResNet (Fang, Yu, Chen, Huang, et al., 2021), one of the most popular architectures in deep SNNs, to further enhance SNNs' performance on the ImageNet-1K dataset.

5.1. Implementation details

Preprocessing. For static image datasets (*i.e.,* CIFAR-10/100, ImageNet), we directly send the original image into the SNNs (*i.e.*, direct encoding). We apply standard data augmentation techniques to the input images: random cropping, flipping, and normalization. For the neuromorphic DVS-CIFAR10 dataset, we reduce the resolution of the input image from 128×128 to 48×48 and apply random cropping.

Training Setting. For small datasets (*e.g.*, CIFAR-10/100, DVS-CIFAR10), we use SGD optimizer to train 200 epochs, and the learning rate is initialized to 0.1, decayed with a cosine strategy for shallow networks, such as ResNet-18/19/34. For deep networks, such as ResNet-50/101/152, we use the AdamW optimizer to train 200 epochs. The learning rate increases linearly from 0 to 0.01 in the first five epochs, and then the cosine decays to 0.

For ImageNet, we use the AdamW optimizer to train 120 epochs for Spiking-ResNet (*i.e.*, ResNet in Table 2). For shallow networks, such as ResNet-18/34, the learning rate was 0.001 cosine decayed to 0. For deep networks, such as ResNet-50, the learning rate increases linearly from 0 to 0.001 in the first five epochs, and then the cosine decays to 0.

Regarding Batch Size. On CIFAR, our experiments run on a single GPU with a default batch size of 128. On ImageNet, most experiments run on 4 GPUs with a default single GPU batch size of 12. If a model is too large for GPU memory, we iteratively halve the batch size until it fits (e.g., 64, 32).

5.2. Comparison with the other method

As shown in Tables 1 and 2, we compare the proposed framework with previous works on four datasets.

CIFAR-10/100. We employ the ResNet-19 (Deng et al., 2022) (*i.e.*, different from the standard ResNet (He, Zhang, Ren, & Sun, 2016) architecture) architecture for CIFAR experiments. Our method achieves top-1 accuracies of 96.06% and 80.10% on CIFAR-10 and CIFAR-100 with 4 time-steps, outperforming all compared methods. Notably, on CIFAR-100, the proposed method surpasses the current best by 5.38%. Meanwhile, with only one time-step, our method can also achieve performances of 95.12% and 77.36%.

DVS-CIFAR10. We employ VGG-11 (Simonyan & Zisserman, 2014) and ResNet-19 architectures on DVS-CIFAR10, achieving the top-1 accuracy of 81.50% and 80.30% with 4 time-steps, respectively. Notably, our method attains 6.80% and 6.30% improvements over the baseline performance of VGG-11 (*i.e.*, 75.20% w/o KD) and ResNet-19 (*i.e.*, 73.50% w/o KD) respectively.

ImageNet. We use the Spiking-ResNet models (*e.g.*, ResNet-18, ResNet-34, and ResNet-50) to verify our algorithm on ImageNet. When operating under conditions of low latency, our method outperforms all the other compared methods and achieves the best performance. Our proposed method achieves a top-1 accuracy of 70.04% for the ResNet-34 model using only 4 time steps. Compared to the Directly-train-SNN method, our approach outperforms the Real Spike method with the same number of time-steps by 2.35% and surpasses the current state-of-the-art GLIF method that uses 6 time-steps by 0.95%. Although the

Table 1
Comparisons with current state-of-the-art methods on CIFAR and DVS-CIFAR. Teacher ANNs' performance: ResNet-19: 96.3% on CIFAR-10, 80.48% on CIFAR-100, 79.60% on DVS-CIFAR-10; VGG-11: 80.90% on DVS-CIFAR.

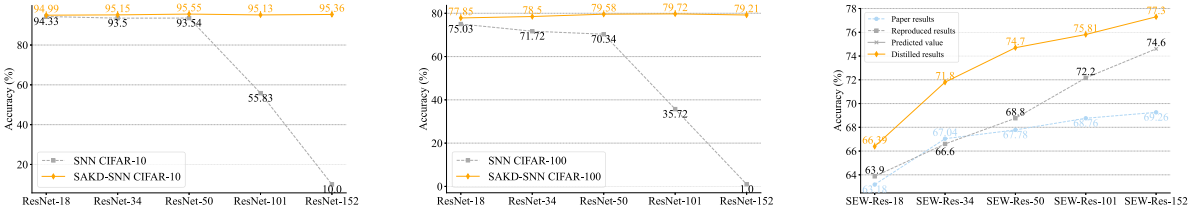
Dataset	Method	Pub. Year	Train	Model	Time-step	Accuracy
CIFAR-10	Rmp-snn (Han et al., 2020)	CVPR2020	ANN-to-SNN	VGG-16	2048	93.63
	Calibration-snn (Li et al., 2021)	ICLR2021	ANN-to-SNN	VGG-16	32	93.71
	Optimal-snn (Bu et al., 2021)	ICLR2022	ANN-to-SNN	ResNet-18	4	90.43
	STBP-tdBN (Zheng et al., 2021)	AAAI2021	Directly-train-SNN	ResNet-19	6	93.16
	TET (Deng et al., 2022)	ICLR2022	Directly-train-SNN	ResNet19	6	94.50
	Rec-Dis (Guo, Tong, et al., 2022)	CVPR2022	Directly-train-SNN	ResNet-19	6	95.55
	DSR (Meng et al., 2022)	CVPR2022	Directly-train-SNN	ResNet-18	20	95.40
	ours	–	SAKD	ResNet-19	1	95.12
					4	96.06
CIFAR-100	Rmp-snn (Han et al., 2020)	CVPR2020	ANN-to-SNN	VGG-16	2048	70.93
	Calibration-snn (Li et al., 2021)	ICLR2021	ANN-to-SNN	VGG-16	256	77.68
	Optimal-snn (Bu et al., 2021)	ICLR2022	ANN-to-SNN	VGG-16	32	77.01
	STBP-tdBN (Zheng et al., 2021)	AAAI2021	Directly-train-SNN	ResNet-19	6	71.72
	TET (Deng et al., 2022)	ICLR2022	Directly-train-SNN	ResNet-19	6	74.72
	Rec-Dis (Guo, Tong, et al., 2022)	CVPR2022	Directly-train-SNN	ResNet-19	6	74.10
	DSR (Meng et al., 2022)	CVPR2022	Directly-train-SNN	ResNet-18	20	78.50
	ours	–	SAKD	ResNet-19	1	77.36
					4	80.10
DVS-CIFAR10	STBP-tdBN (Zheng et al., 2021)	AAAI2021	Directly-train-SNN	ResNet-19	10	67.80
	TET (Deng et al., 2022)	ICLR2022	Directly-train-SNN	VGG-11	10	**83.17**
	DSR (Meng et al., 2022)	CVPR2022	Directly-train-SNN	VGG-11	20	77.27
	Rec-Dis (Guo, Tong, et al., 2022)	ECCV2022	Directly-train-SNN	ResNet-19	10	72.42
	ours	–	SAKD	VGG-11	4	81.50
				ResNet-19		80.30

Table 2
Comparisons with current state-of-the-art methods on ImageNet-1K. Teacher ANNs' performance: ResNet-18: 71.5%, ResNet-34: 76.4%, ResNet-50: 80.4%, ResNet-152: 82.0%.

Method	Pub. Year	Train	Model	Time-step	Accuracy
Rmp-snn (Han et al., 2020)	CVPR2020	ANN-to-SNN	ResNet-34	4096	69.89
Calibration-snn (Li et al., 2021)	ICLR2021	ANN-to-SNN	ResNet-34	32	64.54
Optimal-snn (Bu et al., 2021)	ICLR2022	ANN-to-SNN	ResNet-34	32	67.37
COS–snn (Hao, Ding, Bu, Huang, & Yu, 2023)	ICLR2023	ANN-to-SNN	ResNet-34	6	67.12
STBP-tdBN (Zheng et al., 2021)	AAAI2021	Directly-train-SNN	ResNet-34	6	63.72
SEW (Fang, Yu, Chen, Huang, et al., 2021)	NeurIPS2021	Directly-train-SNN	SEW-ResNet-50	4	67.78
			SEW-ResNet-152		69.26
TET (Deng et al., 2022)	ICLR2022	Directly-train-SNN	ResNet-34	6	64.79
Rec-Dis (Guo, Tong, et al., 2022)	CVPR2022	Directly-train-SNN	ResNet-19	6	67.33
DSR (Meng et al., 2022)	CVPR2022	Directly-train-SNN	ResNet-18	20	67.74
InfLoR (Guo, Chen, et al., 2022)	ECCV2022	Directly-train-SNN	ResNet-34	4	65.54
GLIF (Yao et al., 2022)	NeurIPS2022	Directly-train-SNN	ResNet-34	6	69.09
Real Spike (Guo, Zhang, et al., 2022)	ECCV2022	Directly-train-SNN	ResNet-34	4	67.69
ours	–	SAKD	ResNet-18	4	66.09
			ResNet-34		70.04
			ResNet-50		71.71
			SEW-ResNet-50		74.70
			SEW-ResNet-152		77.30

ANN-to-SNN method can achieve peak performance similar to ANNs under high latency conditions, our proposed method demonstrates a better tradeoff between performance and latency. Even with only 4 time steps, our approach exhibits significantly better performance than the ANN-to-SNN method with 32 time steps.

Compatibility. Knowledge distillation employs a two-stage training process, which first trains powerful ANNs, and then trains SNNs through distillation. As shown in Table 3, it adds an acceptable training overhead but can bring significant performance gains to SNNs (*e.g.*, when training the ResNet-50 model with 4 time steps on CIFAR, using the SAKD method reduces the training speed by 10%, but is still faster than using a 5 time-step training mode, while providing a performance gain of 11.7% (8.25%). Even when considering the cost of training ANNs, the overall training cost of using SAKD method to train SNNs with 4 time steps is still not higher than training SNNs with 5 time steps without SAKD). At the same time, we believe that SAKD can be effectively combined with existing techniques (*e.g.*, GLIF, SEW, TET) for further improvement in SNNs' performance.



(a) Performance of Spiking-ResNet with different depths on CIFAR-10.

(b) Performance of Spiking-ResNet with different depths on CIFAR-100.

(c) Distillation Performance of SEW-ResNet on ImageNet.

Fig. 5. Performance comparison of SNNs with and without SAKD on CIFAR-10, CIFAR-100, and ImageNet datasets.

Table 3
The computational cost of whether to use distillation on CIFAR. We measure peak memory and speed while training on one GPU RTX3090 24GB with batch size 128, automatic mixed precision.

	Step	Speed (image/s)	Memory (GB)	Top-1 acc %
ResNet-50 (ANN)	1	1877	4.07	80.49
ResNet-50	4	460	19.42	70.34
ResNet-50 w KD	4	411	19.84	78.59
ResNet-50	5	378	23.37	71.12

5.3. Deeper spiking neural networks under SAKD

In this section, we conduct a series of experiments with Spiking ResNet of different depths on CIFAR to verify the distillation effect on the deeper network. For each structure, we present the results of ANNs, directly trained SNNs, and SNNs trained with our SAKD method. As shown in Figs. 5(a) and 5(b), notably, traditional SNNs suffer from a severe performance degradation problem for deep networks like ResNet-101/152, while SNNs fostered by our SAKD can converge well and achieve excellent performance. As stated by Fang, Yu, Chen, Huang, et al. (2021), spiking activation cannot achieve identity mapping, and residual connections in SNNs do not solve the gradient problem. However, knowledge distillation can significantly alleviate this issue by transferring knowledge from ANNs to optimize the shallow, middle, and final layers of deep SNNs.

We also push the limits of low-latency SNNs on ImageNet by demonstrating the performance of more advanced SNN models, SEW-ResNet, under the SAKD framework in Fig. 5(c). The figure presents the results for the original paper, our reproduced results, and models trained under SAKD. All models were trained for 210 epochs. As shown in Fig. 5(c), the SEW-ResNet model exhibits impressive performance improvements under the SAKD framework. For instance, SEW-ResNet50 achieves a Top-1 accuracy of 74.7%, reflecting a 6.92% enhancement compared to the original paper. Notably, SEW-ResNet-152 achieves a Top-1 accuracy of 77.3%, an improvement of 7.56% compared to the original paper. This result highlights that SEW-ResNet-152 is currently the highest-performing pure spiking SNNs model on ImageNet. These findings not only validate the effectiveness of our SAKD framework but also underscore the potential of advanced SNN models in large-scale image recognition tasks.

6. Object detection and semantic segmentation

In this section, we extend our approach to tackle dense prediction tasks, including semantic segmentation and object detection. Since SNNs are rarely used in computationally intensive detection tasks, we first discuss the differences between distillation frameworks for object detection and semantic segmentation compared to image classification tasks. Next, we outline the experimental implementation details. All SNN models are trained and evaluated with 4 time steps, which enables a fair comparison of their performances. Finally, we present our experimental results, showcasing the effectiveness of our proposed method in various dense prediction tasks.

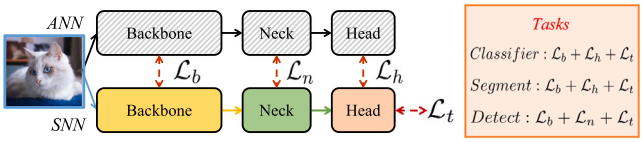


Fig. 6. An illustration of the proposed method for image classification, object detection, and semantic segmentation.

6.1. Framework

For classification tasks, the model typically comprises a Backbone module for feature extraction and a Head module for task-specific output. In our SAKD framework, the SNNs learn the information from these two ANNs modules.

In contrast, detection and segmentation tasks require an additional Neck module to aggregate features from multiple scales. For semantic segmentation tasks, as the Neck module is generally lightweight, our framework omits it (*i.e.*, SNNs do not learn this part of the information of ANNs).

For object detection tasks, our framework considers the Neck module since it is generally more complex and informative. However, the Head module's diverse information makes balancing distillation loss weights difficult. Thus, our framework has SNNs align features only from the teacher's Backbone and Neck modules for training stability.

Our overall framework is shown in Fig. 6.

6.2. Implementation details

We conduct experiments for detection and segmentation tasks based on *YOLOv6* (Li, Li, et al., 2022) and *pytorch-segmentation*link.[3]

Preprocessing. We used the PASCAL VOC 2012 (Everingham et al., 2010) and Cityscapes (Cordts et al., 2016) datasets for segmentation evaluation, and an expanded PASCAL VOC dataset for detection. The PASCAL VOC 2012 dataset was augmented with extra coarse annotations (Hariharan, Arbeláez, Bourdev, Maji, & Malik, 2011), resulting in 10,582 training and 1449 validation images. The Cityscapes dataset consists of 2975 training and 500 validation finely annotated images.

For detection, we combined the PASCAL VOC 2012 and 2007 datasets, yielding 16,551 training and 4952 validation images (Li, Li, et al., 2022). We assessed model performance using mIoU for segmentation and AP for detection on validation sets.

We evaluated all models using mean Intersection over Union (mIoU) for segmentation and Average Precision (AP) for detection on the validation sets.

Training Setting. All training parameters, including data augmentation techniques, optimizer parameters, and learning rate scheduling curves, were set to their default values based on the *YOLOv6* (Li, Li,

[3] https://github.com/yassouali/pytorch-segmentation

Table 4
Results of SAKD for semantic segmentation. Backbone is ResNet-18.

MODEL		EVAL		DATASET
ANN	SNN	$ACC\%$	$mIoU\%$	
PSP−Res18	S-PSP−Res18	85.3	51.6	VOC
(68.6%)	SAKD	90.4	65.6 (+14.0%)	VOC
PSP−Res18	S-PSP−Res18	92.2	48.3	Cityscapes
(57.1%)	SAKD	93.1	54.0 (+5.7%)	Cityscapes
DeepLabV3	S-DeepLabV3	84.0	45.4	VOC
(68.3%)	SAKD	89.6	62.0 (+16.6%)	VOC
DeepLabV3	S-DeepLabV3	92.3	48.7	Cityscapes
(54.2%)	SAKD	92.8	52.8 (+4.1%)	Cityscapes

Table 5
Results of SAKD for object detection on VOC.

MODEL		EVAL				
ANN	SNN	$AP_{0.5}$	AP_S	AP_M	AP_L	$mAP(\%)$
YOLOv6	S-YOLOv6	73.0	17.1	30.5	55.6	49.1
(58.6%)	SAKD	77.8	15.3	35.8	61.0	54.1 (+5.0%)

et al., 2022) and *pytorch-segmentation* frameworks. We used the AdamW optimizer to train our SNNs for 200 epochs in segmentation and 400 epochs in detection.

6.3. Segmentation

For the semantic segmentation task, we conducted experiments on two architectures, PSPNet (Zhao, Shi, Qi, Wang, & Jia, 2017) and DeepLabV3 (Chen, Papandreou, Kokkinos, Murphy, & Yuille, 2017). We converted these two architectures into their spiking versions (SNNs), S-PSPNet and S-DeepLabV3, and evaluated their baseline performance and performance after applying SAKD. The results are shown in the Table 4. When trained using the SAKD method, SNNs demonstrated substantial performance improvements. Specifically, S-PSPNet-Res18 achieved 85.3% accuracy and 51.6% mIoU on the VOC dataset, while SAKD remarkably improved the performance to 90.4% accuracy and 65.6% mIoU. For the Cityscapes dataset, S-PSPNet-Res18 achieved 92.2% accuracy and 48.3% mIoU, while SAKD impressively increased the performance to 93.1% accuracy and 54.0% mIoU. Similarly, S-DeepLabV3 achieved 80.4% accuracy and 45.4% mIoU on the VOC dataset, while SAKD significantly improved the performance to 89.6% accuracy and 62.0% mIoU. For the Cityscapes dataset, S-DeepLabV3 achieved 92.3% accuracy and 48.7% mIoU, while SAKD effectively increased the performance to 92.8% accuracy and 52.8% mIoU.

Overall, the SAKD framework enhances the SNNs' performance by approximately 15% on the VOC dataset and improves the model by about 5% on the Cityscapes dataset. These results show the effectiveness of our SAKD framework in semantic segmentation tasks.

6.4. Detection

For the object detection task, we construct our experiment on the single-stage detector YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016) specifically using the latest release, YOLOv6 (Li, Li, et al., 2022). The basic component is the RepVGG module (Ding, Zhang, et al., 2021), distributed in the BackBone and Neck modules. We converted it to a pure spiking version called S-YOLOv6. The experimental results are shown in the Table 5. Under the SAKD training method, S-YOLOv6 achieves a 54.1% mAP, showing a significant 5% mAP performance improvement compared to not using our SAKD framework.

7. Ablation study

In this section, we conduct a series of experiments to explore the optimal implementation strategies for transferring knowledge from ANNs to SNNs using the bilevel teacher–student training strategy in SAKD. We first investigate the most effective ANN knowledge transfer (including feature and logit distillation) during SNNs training at level-2. Next, we examine the impact of initializing SNNs with ANN weights at level-1, focusing on knowledge transfer before training. We also explore knowledge distillation with different architectures and discuss the influence of distillation on the spike firing rate and feature distribution in the intermediate layers of SNNs.

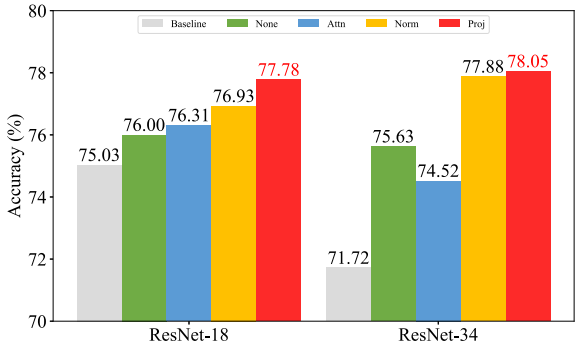


Fig. 7. Effect of features transform on CIFAR-100.

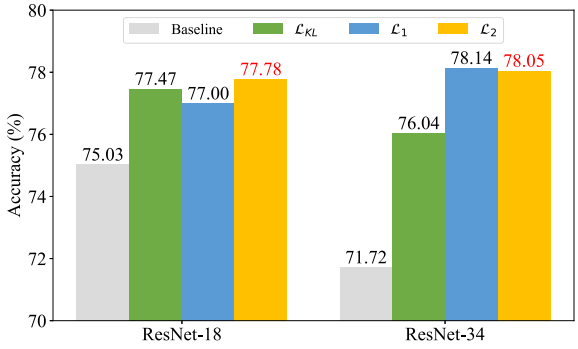


Fig. 8. Effect of distance function on CIFAR-100.

7.1. Feature-based and logit-based distillation

Feature-based Distillation. To determine the optimal configuration for feature-based distillation, we conduct experiments analyzing feature transformation, distance functions, and distillation positions on feature distillation. We disable logit distillation (*i.e.*, $\gamma = 0$, $\beta = 100$ by default) to focus on identifying the best feature distillation implementation.

Feature Transformation for Feature Distillation. Fig. 7 demonstrates the importance of mitigating differences between the original ANNs and SNNs features. Proj represents the projection module in Fig. 3 used to project SNN features into the same space as ANNs. Norm (Liu, Yang, & Chen, 2022) indicates that the original features of ANNs and SNNs are normalized. Attn refers to the attention transfer technique (Zagoruyko & Komodakis, 2016). Both methods significantly enhance SNNs' performance, and we default to using the Proj method to achieve superior results.

Distance Functions for Feature Distillation. Fig. 8 illustrates the impact of various distance functions (*e.g.*, KL, $\mathcal{L}_1$, $\mathcal{L}_2$) on distillation. We select the loss coefficient from $\beta = 1, 10, 100$ to control the loss value obtained by different distance functions. The results indicate that $\mathcal{L}_2$ distance functions exhibit more stable performance, and we adopt $\mathcal{L}_2$ by default.

Table 6
Effect of distillation positions on CIFAR-100. Stage and Block denote aligning features after each stage or each block. For instance, the ResNet-18 structure contains four Stage modules, and each Stage contains two Block modules.

Baseline	ResNet-18		ResNet-34		ResNet-50		ResNet-101		ResNet-152	
	C: 78.35 S: 75.03		C: 79.30 S: 71.72		C: 80.49 S: 70.34		C: 80.83 S: 35.72		C: 81.36 S: 10.00	
	Stage	Block	Stage	Block	Stage	Block	Stage	Block	Stage	Block
Before-spike	77.48	77.41	78.41	78.19	79.16	78.93	*unstable*	*unstable*	*unstable*	*unstable*
After-spike	77.78	77.04	78.05	78.11	78.59	78.66	*unstable*	79.41	*unstable*	79.25

Table 7
Comparison of ImageNet classification performance under different ANN's knowledge. **Init** represents ANN weight initiation, **Log** represents Logit distillation, and **fea** represents feature distillation.

Init	Log	fea	ResNet-18	ResNet-34	ResNet-50
			61.13	64.40	67.45
	✓		63.31	67.32	70.37
		✓	62.03	65.99	69.09
	✓	✓	63.56	68.05	71.18
✓			65.00	66.57	66.21
✓	✓		66.11	67.98	69.45
✓		✓	64.68	67.92	69.96
✓	✓	✓	66.09 (+4.96%)	70.04 (+5.64%)	71.71 (+4.26%)

Location of Feature Distillation. Table 6 shows the performance of different distillation positions. Since the SNNs share the same architecture as ANNs, we can choose any layer to align their features. We compare two strategies to determine the alignment position. *Stage* refers to aligning features after each entire stage, while *Block* refers to aligning features after each single block (*e.g.*, Bottleneck or Basicblock in ResNet (He et al., 2016)). Furthermore, we explore whether feature distillation should be located before or after the spiking neuron in experiments. The results suggest that distillation for each stage and before spiking neurons may be a better choice in shallow networks. However, in deeper network architectures (such as ResNet-101/152), only matching the features after the block and spiking neuron can ensure the normal convergence of SNNs (*e.g.*, ResNet-101: 79.41% (Block and After-spike) vs. unstable (Block and Before-spike) vs. unstable (Stage)). We set the feature distillation after each stage and the spiking neuron as the default for simplicity.

Logits-based Distillation. Historically, the best distillation methods have typically been feature distillation (Zhao, Cui, Song, Qiu, & Liang, 2022). However, recent studies have demonstrated that logit distillation can provide significant performance improvements and even be more efficient than feature-based methods (Beyer et al., 2022; Hsu, Smith, Shen, Kira, & Jin, 2022; Ridnik, Lawen, Ben-Baruch, & Noy, 2022). We aim to leverage both types of knowledge to enhance SNN performance. As shown in Table 7, on large datasets (*i.e.*, ImageNet), learning the logits knowledge from ANNs alone often results in better improvements than learning the features alone (*e.g.*, ResNet18 63.31% vs. 62.03%, ResNet34 67.32% vs. 65.99%, and ResNet50 70.37% vs. 69.09%). Our SAKD framework takes this a step further by simultaneously transferring both types of knowledge, leading to even better performance enhancements for SNNs (*e.g.*, ResNet18 63.56%, ResNet34 68.05%, and ResNet50 71.18%).

7.2. Hyperparameter

Following the recommendations of Kim, Oh, Kim, Cho, and Yun (2021), Ridnik et al. (2022) and our experiments, we set the logits loss hyperparameters as $\tau = 20, \alpha = 0, \gamma = 1, \beta = 100$ for CIFAR and $\tau = 1, \alpha = 1, \gamma = 10, \beta = 10$ for ImageNet, which perform well. Our experiments in Fig. 10 show that our method consistently improves performance under different distillation loss factors (α, γ, β).

Early Termination on ImageNet. *It is worth noting that we find it unnecessary or even harmful to force SNNs to strive to simulate ANN's features on ImageNet*, as shown in Fig. 9. Some previous work has also explored the effectiveness of feature distillation on ImageNet. Kim et al. (2021) discovered that the teacher's information contains too much noise because the model cannot completely fit the large dataset. Cho and Hariharan (2019) also pointed out that when the capacity gap between the teacher and the student is too large, the continuous imitation of the teacher's knowledge will limit the student's ability. However, this problem can be avoided by ending the distillation process early. Interestingly, Chen et al. (2022) also found that matching features of ANNs in the early training stage and canceling them in the later stage would improve the performance of Vision Transformer. Therefore, we significantly reduce the feature loss coefficient β in the middle of training on ImageNet (*i.e.*, set β to 0.01 after 10 (20) epochs during the entire 120 (210) epoch training cycle).

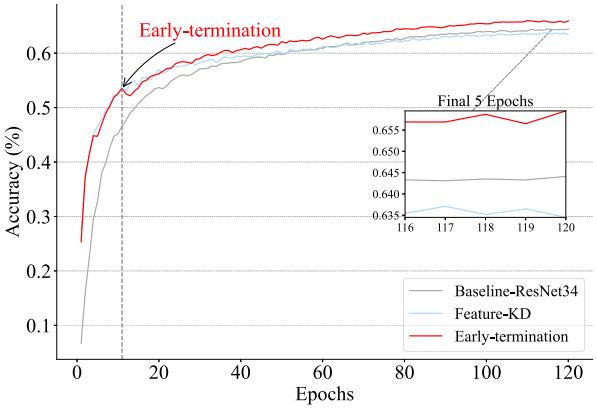


Fig. 9. Early Termination for Feature Distillation on ImageNet.

7.3. Weight-initialization training on ImageNet

In our SAKD framework, loading the pre-trained ANNs weights is the first step in training the SNNs, as shown in Eq. (5). Some recent studies (Meng et al., 2022; Rathi & Roy, 2021; Rathi et al., 2020) show that loading the weights of pre-trained ANNs before training SNNs can accelerate convergence and improve performance. As part of our proposed method, we investigate the influence of weight initialization on distillation and explore the effect of using ANN weights with different performance levels, obtained from PyTorch (Paszke et al., 2017) and Timm (Wightman, Touvron, & Jégou, 2021), respectively (*e.g.*, ANN: ResNet-50 76.1% vs. 80.4%). The results are shown in Table 8. Loading the pre-trained weights of ANNs can significantly improve the performance of SNNs. However, loading higher-performing ANN weights does not necessarily lead to better performance improvements for SNNs, as seen with ResNet-50 (*e.g.*, SNN: ResNet-50 67.99% vs. 66.21%). Nevertheless, within our SAKD framework, higher-performing ANNs mean that we can train higher-performing SNNs.

Unfortunately, we observed that weight initialization hurts performance on small datasets (*e.g.*, CIFAR) under SAKD. Hence, we only use weight initialization on ImageNet.

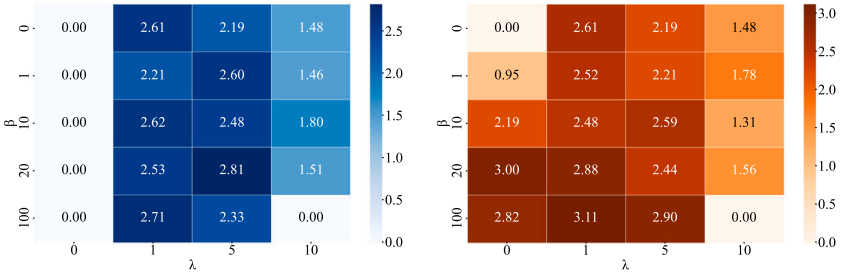


Fig. 10. Ablation study of the impact of different γ, β values under $\alpha = 0$ (left) and $\alpha = 1$ (right) on performance improvement.

Table 8
Comparison on ImageNet classification under the different performance teachers.

	Model	ANN	SNN	weight	SAKD
torch	ResNet-18	69.8	61.13	63.49	65.37 (+4.24%)
	ResNet-34	73.3	64.40	65.05	67.59 (+3.19%)
	ResNet-50	76.1	67.45	67.99	70.71 (+3.26%)
timm	ResNet-18	71.5	61.13	65.00	66.09 (+4.96%)
	ResNet-34	76.4	64.40	66.57	70.04 (+5.64%)
	ResNet-50	80.4	67.45	66.21	71.71 (+4.26%)

Table 9
Distillation performance of ANN-teach-SNN with different architectures on ImageNet (18 denotes ResNet18).

	ANN-18	ANN-34	ANN-50	ANN-101	ANN-152
SNN-18	66.09	66.68	66.25	65.49	65.40
SNN-34	–	70.04	68.09	68.31	68.19
SNN-50	–	–	71.71	71.03	70.62

7.4. Distillation of different architectures

In this section, we discuss the impact of different architectures on distillation. As shown in Table 9, the distillation of the same architecture achieves the best results in most cases. This is consistent with claims in some studies (Liu et al., 2021, 2020; Mirzadeh et al., 2020; Qiu et al., 2022) that architectural knowledge is crucial in distillation, and a better-performing teacher is not always better. Notably, all experiments load weights from ANNs of the same architecture before training SNNs, which may cause SNNs to be more biased toward teachers with identical architecture.

7.5. Distillation of different time steps

Facing different inputs, SNNs may require different time steps to achieve the best output. Therefore, a time step of 4 may not be accurate enough to describe the prediction confidence. We add experiment to explore the impact of larger time steps on our method, and the performance variations for different time steps as show in and Fig. 11.

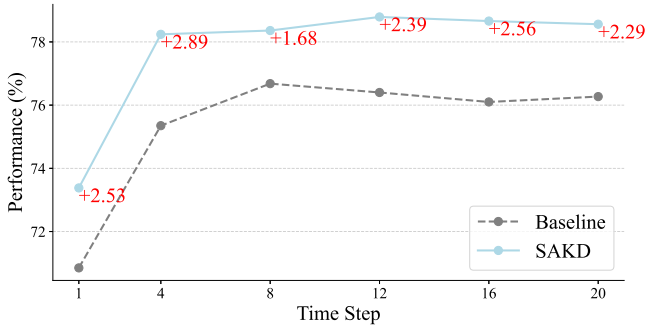


Fig. 11. Comparison on performance under different time steps.

As can be seen, our method can also improve performance at larger time steps. Directly trained SNNs often have a saturation performance, meaning that performance tends to saturate as the training time step increases (because a larger time step increases the training difficulty). However, under distillation, the performance gap between low-latency SNNs and their saturation performance narrows. This experiment shows that even when using a small number of time steps (*e.g.*, 4) for the final output, our method can still accurately describe the prediction confidence.

7.6. Distillation of different surrogate gradient

In this section, we discuss the impact of different surrogate gradients on distillation. The three common types of surrogate gradients are as follows:

$$\Theta'(x) = \max(0, 1 - |x|) - Triangle, \tag{11}$$

$$\Theta'(x) = \eta \cdot (1 - \text{sigmoid}(\eta x)) \cdot \text{sigmoid}(\eta x) - Sigmoid, \tag{12}$$

$$\Theta'(x) = \frac{\eta}{2\left(1 + \left(\frac{\pi}{2}\eta x\right)^2\right)} - ATan. \tag{13}$$

Fig. 12 shows the performance of the SAKD method under different surrogate gradients, with η set to 4 for the *Sigmoid* and 2 for the *ATan*. In all cases, SAKD provides significant performance improvements. This indicates that our method is not sensitive to the nature of the surrogate gradient.

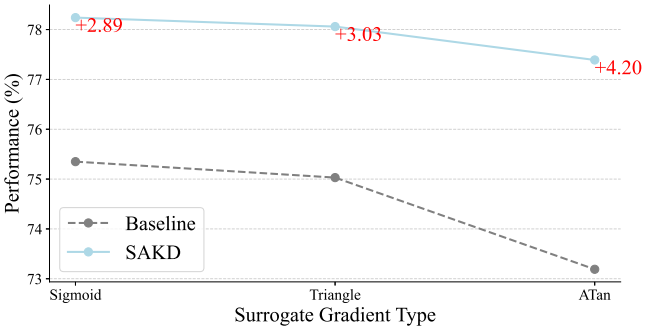


Fig. 12. Comparison on performance under different surrogate gradients.

7.7. Spike firing rate

Sparser spiking activity can make better SNNs (Yao, Zhang, et al., 2023). We conducted additional experiments on ImageNet to further investigate how KD influences the spike sparsity (firing rate) of SNNs. As illustrated in Fig. 13, within the SAKD framework, the spike firing rate of each layer in SNNs is marginally improved, which is necessary to ensure richer spike features. Nevertheless, it still maintains a relatively low firing rate, contributing to the low-power consumption of SNNs.

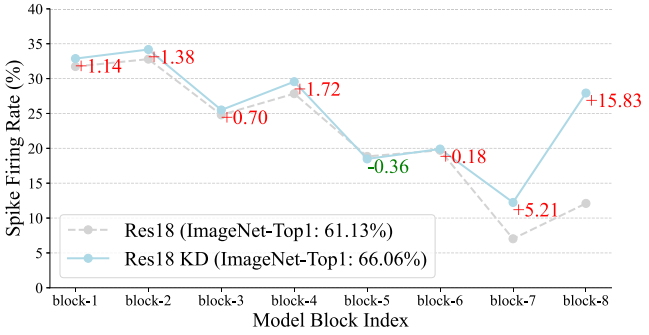


Fig. 13. Comparison on Spike Firing Rate under SAKD.

7.8. Visualization

Feature redundancy is a common issue in SNNs, as indicated in Yao, Hu, et al. (2023). Fig. 14 shows the features in the intermediate layer and the gradient-weighted class activation mapping (Grad-CAM) (Selvaraju et al., 2017) of ANNs and SNNs. The first row represents the features of ANNs. The second row corresponds to SNNs without SAKD, which struggle to produce rich features, often resulting in redundant features with meaningless noise. The last row demonstrates that SNNs generate more informative features by mimicking ANN's features, and Grad-CAM also indicates that SNNs are more focused on the core region under SAKD, which not only enhances the richness of the features but also effectively reduces the redundancy that is typically marked by irrelevant noise within the features.

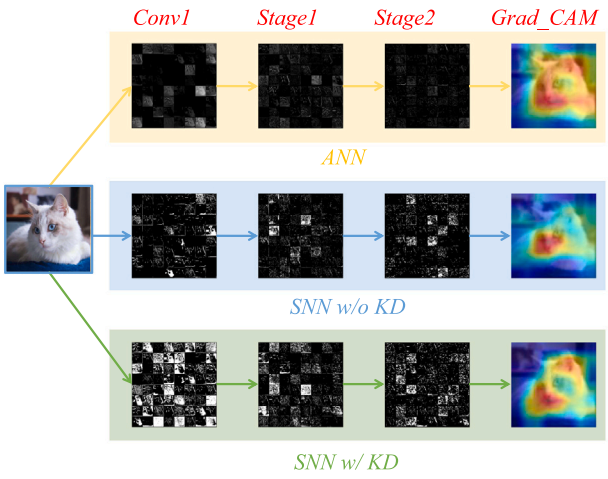


Fig. 14. Visualization of feature maps and GradCAM of ANNs and SNNs w/o SAKD.

8. Conclusion

In this paper, we present a novel Self-Architectural Knowledge Distillation (SAKD) framework effectively addressing the performance-latency trade-off in SNNs. Utilizing knowledge distillation, our method combines ANN-to-SNN and directly-trained-SNN approaches' strengths. We successfully apply the SAKD framework to various datasets and tasks, including image classification, object detection, and semantic segmentation, achieving SOTA performance and showcasing its strong generalization ability. This work demonstrates the immense potential of the SAKD framework for training high-performance, low-latency SNNs across a wide range of visual tasks, paving the way for future advancements in the field.

However, our work has limitations, such as the extra computational cost for the teacher model. We plan to improve this by exploring more effective distillation methods and introducing offline knowledge distillation to save teacher inference costs.

CRediT authorship contribution statement

Haonan Qiu: Conceptualization, Data curation, Investigation, Methodology, Validation, Visualization, Writing – original draft, Writing – review & editing. **Munan Ning:** Writing – original draft. **Zeyin Song:** Writing – original draft. **Wei Fang:** Methodology, Writing – original draft. **Yanqi Chen:** Writing – original draft. **Tao Sun:** Writing – original draft. **Zhengyu Ma:** Supervision, Funding acquisition. **Li Yuan:** Conceptualization, Supervision, Funding acquisition. **Yonghong Tian:** Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request. The source codes of SAKD can be found at its GitHub page: https://github.com/Maybe2022/SAKD.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grants No. 62425101, 62332002, 62027804, and 62088102).

References

Ahn, Sungsoo, Hu, Shell Xu, Damianou, Andreas, Lawrence, Neil D., & Dai, Zhenwen (2019). Variational information distillation for knowledge transfer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9163–9171).

Bellec, Guillaume, Salaj, Darjan, Subramoney, Anand, Legenstein, Robert, & Maass, Wolfgang (2018). Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in Neural Information Processing Systems*, *31*.

Beyer, Lucas, Zhai, Xiaohua, Royer, Amélie, Markeeva, Larisa, Anil, Rohan, & Kolesnikov, Alexander (2022). Knowledge distillation: A good teacher is patient and consistent. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10925–10934).

Bohte, Sander M., Kok, Joost N., & La Poutré, Johannes A. (2000). SpikeProp: backpropagation for networks of spiking neurons. In *ESANN, Vol. 48* (pp. 419–424). Bruges.

Bu, Tong, Fang, Wei, Ding, Jianhao, Dai, PengLin, Yu, Zhaofei, & Huang, Tiejun (2021). Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International conference on learning representations*.

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

Chen, Xianing, Cao, Qiong, Zhong, Yujie, Zhang, Jing, Gao, Shenghua, & Tao, Dacheng (2022). DearKD: Data-efficient early knowledge distillation for vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12052–12062).

Chen, Pengguang, Liu, Shu, Zhao, Hengshuang, & Jia, Jiaya (2021). Distilling knowledge via knowledge review. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5008–5017).

Chen, Liang-Chieh, Papandreou, George, Kokkinos, Iasonas, Murphy, Kevin, & Yuille, Alan L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4), 834–848.

Cho, Jang Hyun, & Hariharan, Bharath (2019). On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 4794–4802).

Chowdhury, Sayeed Shafayet, Rathi, Nitin, & Roy, Kaushik (2021). One timestep is all you need: training spiking neural networks with ultra low latency. arXiv preprint arXiv:2110.05929.

Cordts, Marius, Omran, Mohamed, Ramos, Sebastian, Rehfeld, Timo, Enzweiler, Markus, Benenson, Rodrigo, et al. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3213–3223).

Datta, Gourav, & Beerel, Peter A. (2022). Can deep neural networks be converted to ultra low-latency spiking neural networks? In *2022 design, automation & test in Europe conference & exhibition* (pp. 718–723). IEEE.

Datta, Gourav, Liu, Zeyu, Diffenderfer, James, Kailkhura, Bhavya, & Beerel, Peter A. (2023). When bio-inspired computing meets deep learning: Low-latency, accurate, & energy-efficient spiking neural networks from artificial neural networks. arXiv preprint arXiv:2312.06900.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, & Fei-Fei, Li (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255). Ieee.

Deng, Shikuang, & Gu, Shi (2021). Optimal conversion of conventional artificial neural networks to spiking neural networks. arXiv preprint arXiv:2103.00476.

Deng, Shikuang, Li, Yuhang, Zhang, Shanghang, & Gu, Shi (2022). Temporal efficient training of spiking neural network via gradient re-weighting. arXiv preprint arXiv:2202.11946.

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

Ding, Jianhao, Yu, Zhaofei, Tian, Yonghong, & Huang, Tiejun (2021). Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. arXiv preprint arXiv:2105.11654.

Ding, Xiaohan, Zhang, Xiangyu, Ma, Ningning, Han, Jungong, Ding, Guiguang, & Sun, Jian (2021). Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 13733–13742).

Dong, Yiting, Zhao, Dongcheng, & Zeng, Yi (2023). Temporal knowledge sharing enable spiking neural network learning from past and future. arXiv preprint arXiv:2304.06540.

Esser, Steve K., Appuswamy, Rathinakumar, Merolla, Paul, Arthur, John V., & Modha, Dharmendra S. (2015). Backpropagation for energy-efficient neuromorphic computing. *Advances in Neural Information Processing Systems*, *28*.

Everingham, Mark, Van Gool, Luc, Williams, Christopher K. I., Winn, John, & Zisserman, Andrew (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, *88*, 303–338.

Fang, Wei, Chen, Yanqi, Ding, Jianhao, Yu, Zhaofei, Masquelier, Timothée, Chen, Ding, et al. (2023). Spikingjelly: an open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, *9*(40), eadi1480.

Fang, Wei, Yu, Zhaofei, Chen, Yanqi, Huang, Tiejun, Masquelier, Timothée, & Tian, Yonghong (2021). Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, *34*, 21056–21069.

Fang, Wei, Yu, Zhaofei, Chen, Yanqi, Masquelier, Timothée, Huang, Tiejun, & Tian, Yonghong (2021). 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).

Gerstner, Wulfram, Kistler, Werner M., Naud, Richard, & Paninski, Liam (2014). *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press.

Guo, Yufei, Chen, Yuanpei, Zhang, Liwen, Wang, YingLei, Liu, Xiaode, Tong, Xinyi, et al. (2022). Reducing information loss for spiking neural networks. In *European conference on computer vision* (pp. 36–52). Springer.

Guo, Yufei, Tong, Xinyi, Chen, Yuanpei, Zhang, Liwen, Liu, Xiaode, Ma, Zhe, et al. (2022). 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* (pp. 326–335).

Guo, Yufei, Zhang, Liwen, Chen, Yuanpei, Tong, Xinyi, Liu, Xiaode, Wang, YingLei, et al. (2022). Real spike: Learning real-valued spikes for spiking neural networks. In *European conference on computer vision* (pp. 52–68). Springer.

Han, Bing, & Roy, Kaushik (2020). Deep spiking neural network: Energy efficiency through time based coding. In *European conference on computer vision* (pp. 388–404). Springer.

Han, Bing, Srinivasan, Gopalakrishnan, & Roy, Kaushik (2020). 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* (pp. 13558–13567).

Hao, Zecheng, Bu, Tong, Ding, Jianhao, Huang, Tiejun, & Yu, Zhaofei (2023). Reducing ann-snn conversion error through residual membrane potential. In *Proceedings of the AAAI conference on artificial intelligence, vol. 37, no. 1* (pp. 11–21).

Hao, Zecheng, Ding, Jianhao, Bu, Tong, Huang, Tiejun, & Yu, Zhaofei (2023). Bridging the gap between ANNs and SNNs by calibrating offset spikes. arXiv preprint arXiv:2302.10685.

Hariharan, Bharath, Arbeláez, Pablo, Bourdev, Lubomir, Maji, Subhransu, & Malik, Jitendra (2011). Semantic contours from inverse detectors. In *2011 international conference on computer vision* (pp. 991–998). IEEE.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Heo, Byeongho, Kim, Jeesoo, Yun, Sangdoo, Park, Hyojin, Kwak, Nojun, & Choi, Jin Young (2019). A comprehensive overhaul of feature distillation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1921–1930).

Hinton, Geoffrey, Vinyals, Oriol, Dean, Jeff, et al. (2015). Distilling the knowledge in a neural network, vol. 2, no. 7. arXiv preprint arXiv:1503.02531.

Hong, Di, Shen, Jiangrong, Qi, Yu, & Wang, Yueming (2023). LaSNN: Layer-wise ANN-to-SNN distillation for effective and efficient training in deep spiking neural networks. arXiv preprint arXiv:2304.09101.

Hsu, Yen-Chang, Smith, James, Shen, Yilin, Kira, Zsolt, & Jin, Hongxia (2022). A closer look at knowledge distillation with features, logits, and gradients. arXiv preprint arXiv:2203.10163.

Hu, Yangfan, Zheng, Qian, Jiang, Xudong, & Pan, Gang (2023). Fast-SNN: Fast spiking neural network by converting quantized ANN. arXiv preprint arXiv:2305.19868.

Huh, Dongsung, & Sejnowski, Terrence J. (2018). Gradient descent for spiking neural networks. *Advances in Neural Information Processing Systems*, *31*.

Hunsberger, Eric, & Eliasmith, Chris (2015). Spiking deep networks with LIF neurons. arXiv preprint arXiv:1510.08829.

Kim, Youngeun, Li, Yuhang, Park, Hyoungseob, Venkatesha, Yeshwanth, & Panda, Priyadarshini (2022). Neural architecture search for spiking neural networks. arXiv preprint arXiv:2201.10355.

Kim, Taehyeon, Oh, Jaehoon, Kim, NakYil, Cho, Sangwook, & Yun, Se-Young (2021). Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation. arXiv preprint arXiv:2105.08919.

Kim, Seijoon, Park, Seongsik, Na, Byunggook, & Yoon, Sungroh (2020). Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 07* (pp. 11270–11277).

Krizhevsky, Alex, Nair, Vinod, & Hinton, Geoffrey (2014). The CIFAR-10 dataset, 55(5). online: http://www.cs.toronto.edu/kriz/cifar.html.

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

Kushawaha, Ravi Kumar, Kumar, Saurabh, Banerjee, Biplab, & Velmurugan, Rajbabu (2021). Distilling spikes: Knowledge distillation in spiking neural networks. In *2020 25th international conference on pattern recognition* (pp. 4536–4543). IEEE.

Ledinauskas, Eimantas, Ruseckas, Julius, Juršėnas, Alfonsas, & Buračas, Giedrius (2020). Training deep spiking neural networks. arXiv preprint arXiv:2006.04436.

Lee, Jun Haeng, Delbruck, Tobi, & Pfeiffer, Michael (2016). Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, *10*, 508.

Li, Yuhang, Deng, Shikuang, Dong, Xin, Gong, Ruihao, & Gu, Shi (2021). A free lunch from ANN: Towards efficient, accurate spiking neural networks calibration. In *International conference on machine learning* (pp. 6316–6325). PMLR.

Li, Yuhang, Kim, Youngeun, Park, Hyoungseob, Geller, Tamar, & Panda, Priyadarshini (2022). Neuromorphic data augmentation for training spiking neural networks. arXiv preprint arXiv:2203.06145.

Li, Chuyi, Li, Lulu, Jiang, Hongliang, Weng, Kaiheng, Geng, Yifei, Li, Liang, et al. (2022). YOLOv6: A single-stage object detection framework for industrial applications. arXiv preprint arXiv:2209.02976.

Li, Hongmin, Liu, Hanchao, Ji, Xiangyang, Li, Guoqi, & Shi, Luping (2017). Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in Neuroscience*, *11*, 309.

Li, Kehan, Yu, Runyi, Wang, Zhennan, Yuan, Li, Song, Guoli, & Chen, Jie (2022). Locality guidance for improving vision transformers on tiny datasets. arXiv preprint arXiv:2207.10026.

Liu, Li, Huang, Qingle, Lin, Sihao, Xie, Hongwei, Wang, Bing, Chang, Xiaojun, et al. (2021). Exploring inter-channel correlation for diversity-preserved knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 8271–8280).

Liu, Yu, Jia, Xuhui, Tan, Mingxing, Vemulapalli, Raviteja, Zhu, Yukun, Green, Bradley, et al. (2020). Search to distill: Pearls are everywhere but not the eyes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7539–7548).

Liu, Tao, Yang, Xi, & Chen, Chenshu (2022). Normalized feature distillation for semantic segmentation. arXiv preprint arXiv:2207.05256.

Maass, Wolfgang (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, *10*(9), 1659–1671.

Meng, Qingyan, Xiao, Mingqing, Yan, Shen, Wang, Yisen, Lin, Zhouchen, & Luo, Zhi-Quan (2022). 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).

Mirzadeh, Seyed Iman, Farajtabar, Mehrdad, Li, Ang, Levine, Nir, Matsukawa, Akihiro, & Ghasemzadeh, Hassan (2020). Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 04* (pp. 5191–5198).

Mostafa, Hesham (2017). Supervised learning based on temporal coding in spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(7), 3227–3235.

Neftci, Emre O., Mostafa, Hesham, & Zenke, Friedemann (2019). 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.

Paszke, Adam, Gross, Sam, Chintala, Soumith, Chanan, Gregory, Yang, Edward, DeVito, Zachary, et al. (2017). Automatic differentiation in pytorch.

Qiu, Zengyu, Ma, Xinzhu, Yang, Kunlin, Liu, Chunya, Hou, Jun, Yi, Shuai, et al. (2022). Better teacher better student: Dynamic prior knowledge for knowledge distillation. arXiv preprint arXiv:2206.06067.

Rathi, Nitin, & Roy, Kaushik (2021). DIET-SNN: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*.

Rathi, Nitin, Srinivasan, Gopalakrishnan, Panda, Priyadarshini, & Roy, Kaushik (2020). Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. arXiv preprint arXiv:2005.01807.

Redmon, Joseph, Divvala, Santosh, Girshick, Ross, & Farhadi, Ali (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788).

Ridnik, Tal, Lawen, Hussam, Ben-Baruch, Emanuel, & Noy, Asaf (2022). Solving ImageNet: a unified scheme for training any backbone to top results. arXiv preprint arXiv:2204.03475.

Romero, Adriana, Ballas, Nicolas, Kahou, Samira Ebrahimi, Chassang, Antoine, Gatta, Carlo, & Bengio, Yoshua (2014). Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550.

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

Rueckauer, Bodo, Lungu, Iulia-Alexandra, Hu, Yuhuang, Pfeiffer, Michael, & Liu, Shih-Chii (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, *11*, 682.

Selvaraju, Ramprasaath R., Cogswell, Michael, Das, Abhishek, Vedantam, Ramakrishna, Parikh, Devi, & Batra, Dhruv (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (pp. 618–626).

Sengupta, Abhronil, Ye, Yuting, Wang, Robert, Liu, Chiao, & Roy, Kaushik (2019). Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience*, *13*, 95.

Shrestha, Sumit B., & Orchard, Garrick (2018). Slayer: Spike layer error reassignment in time. *Advances in Neural Information Processing Systems*, *31*.

Simonyan, Karen, & Zisserman, Andrew (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

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

Takuya, Sugahara, Zhang, Renyuan, & Nakashima, Yasuhiko (2021). Training low-latency spiking neural network through knowledge distillation. In *2021 IEEE symposium in low-power and high-speed chips* (pp. 1–3). IEEE.

Tang, Jianxiong, Lai, Jian-Huang, Xie, Xiaohua, Yang, Lingxiao, & Zheng, Wei-Shi (2023). AC2AS: Activation consistency coupled ANN-SNN framework for fast and memory-efficient SNN training. *Pattern Recognition*, *144*, Article 109826.

Touvron, Hugo, Cord, Matthieu, Douze, Matthijs, Massa, Francisco, Sablayrolles, Alexandre, & Jégou, Hervé (2021). Training data-efficient image transformers & distillation through attention. In *International conference on machine learning* (pp. 10347–10357). PMLR.

Tung, Frederick, & Mori, Greg (2019). Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1365–1374).

Wang, Ziming, Lian, Shuang, Zhang, Yuhao, Cui, Xiaoxin, Yan, Rui, & Tang, Huajin (2022). Towards lossless ANN-SNN conversion under ultra-low latency with dual-phase optimization. arXiv preprint arXiv:2205.07473.

Wightman, Ross, Touvron, Hugo, & Jégou, Hervé (2021). Resnet strikes back: An improved training procedure in timm. arXiv preprint arXiv:2110.00476.

Wu, Yujie, Deng, Lei, Li, Guoqi, Zhu, Jun, & Shi, Luping (2018). Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, *12*, 331.

Wu, Yujie, Deng, Lei, Li, Guoqi, Zhu, Jun, Xie, Yuan, & Shi, Luping (2019). Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence, vol. 33, no. 01* (pp. 1311–1318).

Xiao, Mingqing, Meng, Qingyan, Zhang, Zongpeng, Wang, Yisen, & Lin, Zhouchen (2021). Training feedback spiking neural networks by implicit differentiation on the equilibrium state. *Advances in Neural Information Processing Systems*, *34*, 14516–14528.

Xu, Qi, Li, Yaxin, Fang, Xuanye, Shen, Jiangrong, Liu, Jian K, Tang, Huajin, et al. (2023). Biologically inspired structure learning with reverse knowledge distillation for spiking neural networks. arXiv preprint arXiv:2304.09500.

Xu, Qi, Li, Yaxin, Shen, Jiangrong, Liu, Jian K, Tang, Huajin, & Pan, Gang (2023). Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. arXiv preprint arXiv:2304.05627.

Xu, Haohang, Zhang, Xiaopeng, Li, Hao, Xie, Lingxi, Dai, Wenrui, Xiong, Hongkai, et al. (2022). Seed the views: Hierarchical semantic alignment for contrastive representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yan, Zhanglu, Zhou, Jun, & Wong, Weng-Fai (2021). Near lossless transfer learning for spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence, vol. 35, no. 12* (pp. 10577–10584).

Yan, Zhanglu, Zhou, Jun, & Wong, Weng-Fai (2023). CQ$^+$+ training: Minimizing accuracy loss in conversion from convolutional neural networks to spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(10), 11600–11611. http://dx.doi.org/10.1109/TPAMI.2023.3286121.

Yang, Zhendong, Li, Zhe, Shao, Mingqi, Shi, Dachuan, Yuan, Zehuan, & Yuan, Chun (2022). Masked generative distillation. arXiv preprint arXiv:2205.01529.

Yang, Qu, Wu, Jibin, Zhang, Malu, Chua, Yansong, Wang, Xinchao, & Li, Haizhou (2022). Training spiking neural networks with local tandem learning. arXiv preprint arXiv:2210.04532.

Yang, Yukun, Zhang, Wenrui, & Li, Peng (2021). Backpropagated neighborhood aggregation for accurate training of spiking neural networks. In *International conference on machine learning* (pp. 11852–11862). PMLR.

Yao, Man, Hu, Jiakui, Zhao, Guangshe, Wang, Yaoyuan, Zhang, Ziyang, Xu, Bo, et al. (2023). Inherent redundancy in spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 16924–16934).

Yao, Xingting, Li, Fanrong, Mo, Zitao, & Cheng, Jian (2022). GLIF: A unified gated leaky integrate-and-fire neuron for spiking neural networks. arXiv preprint arXiv:2210.13768.

Yao, Man, Zhang, Hengyu, Zhao, Guangshe, Zhang, Xiyu, Wang, Dingheng, Cao, Gang, et al. (2023). Sparser spiking activity can be better: Feature refine-and-mask spiking neural network for event-based visual recognition. *Neural Networks*, *166*, 410–423.

Yue, Kaiyu, Deng, Jiangfan, & Zhou, Feng (2020). Matching guided distillation. In *European conference on computer vision* (pp. 312–328). Springer.

Zagoruyko, Sergey, & Komodakis, Nikos (2016). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. arXiv preprint arXiv:1612.03928.

Zhang, Wenrui, & Li, Peng (2020). Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Advances in Neural Information Processing Systems*, *33*, 12022–12033.

Zhao, Borui, Cui, Quan, Song, Renjie, Qiu, Yiyu, & Liang, Jiajun (2022). Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11953–11962).

Zhao, Hengshuang, Shi, Jianping, Qi, Xiaojuan, Wang, Xiaogang, & Jia, Jiaya (2017). Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2881–2890).

Zheng, Hanle, Wu, Yujie, Deng, Lei, Hu, Yifan, & Li, Guoqi (2021). Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence, vol. 35, no. 12* (pp. 11062–11070).