SELF-ARCHITECTURAL KNOWLEDGE DISTILLATION FOR SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Brain-inspired spiking neural networks (SNNs) have drawn wide attention recently since they are biologically plausible and neuromorphic hardware-friendly. To obtain low-latency (*i.e.*, a small number of timesteps) SNNs, the surrogate gradients (SG) method has been widely applied. However, SNNs trained by the SG method still have a huge performance gap from artificial neural networks (ANNs). In this paper, we find that the knowledge distillation paradigm can effectively alleviate the performance gap by transferring the knowledge from ANNs (teacher) to SNNs (student), but it remains a problem to find the architecture of teacher-student pairs. We introduce neural architecture search (NAS) and find that the performance is insensitive to the architectures of SNNs. Hence, we choose the same architecture for ANN-teacher and SNN-student since it is easy to implement and the student can initiate the weight from the teacher. We thus propose a Self-Architectural Knowledge Distillation framework (SAKD), which transfers the knowledge (*i.e.*, the features and logits) of ANNs to that of SNNs with the same architecture. Although adopting a teacher model in training, SNNs trained via our SAKD still keep ultra-low latency (T=4) compared with other methods and achieve state-of-the-art performance on a variety of datasets (*e.g.*, CIFAR-10, CIFAR-100, ImageNet, and DVS-CIFAR10), and we demonstrate that this simple training strategy can provide a new training paradigm of SNNs.

1 INTRODUCTION

Deep neural networks have achieved great success in the last decade and made breakthrough progress on various tasks. Recently, there has been a lot of interest in brain-inspired spiking neural networks (SNNs), which are third-generation neural networks by mimicking biological neurons and spike firing patterns (Maass, 1997; Gerstner et al., 2014; Roy et al., 2019). At the same time, with the development of neuromorphic hardware, spiking neural network has broad application prospects, due to their advantages in low-energy computing and adaptability with neuromorphic hardware. However, because of their complex neuronal dynamics and the non-differentiable spike firing, SNNs still face optimization difficulties under the current deep learning framework dominated by gradient backpropagation, which limits its performance.

A convenient solution is skipping the gradient backpropagation phase and directly conversing a pre-trained ANN to SNN through weight sharing and substitution activation functions, named ANN-to-SNN conversion as shown in Figure 1. However, the practice of this strategy is limited by the extremely high latency. To achieve the performance of ANNs, the number of timesteps are always large (*e.g.*, 256 steps in Li et al. (2021); Bu et al. (2021) and 2048 steps in Han et al. (2020)), which lead to extremely high computation cost. Better methods are in urgent need to reduce the timestep while maintaining the performance. Another mainstream method is direct training SNNs with backpropagation (Neftci et al., 2019; Meng et al., 2022), and we summarize this series of methods as BP-for-SNN. These studies have addressed the problem of non-differentiable spiking activation function for backpropagation learning. A typical solution is to design the backpropagation gradient by accumulating inputs and outputs in a certain period of time (Meng et al., 2022; Wu et al., 2021a). However, this method still needs a large number of timesteps to ensure the reliability of the designed backpropagation gradient to maintain high performance. In contrast, to train an ultra-low-latency SNN, some studies (Wu et al., 2018; Shrestha & Orchard, 2018; Lee et al., 2016; Huh & Sejnowski, 2018; Lee et al., 2020; Neftci et al., 2019) have proposed the surrogate gradient (SG)

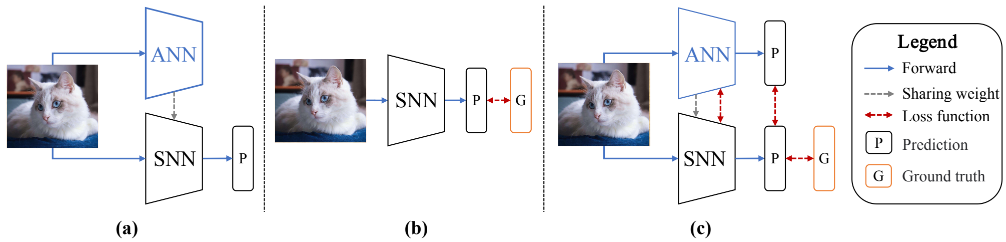


Figure 1: An illustration of existing mainstream SNNs training methods. **(a)**: Convert pre-trained ANNs to SNNs (ANN-to-SNN). **(b)**: Gradient back-propagation training SNNs (BP-for-SNN). **(c)**: Our proposed method: knowledge of ANNs is transferred to SNNs of the same architecture through features and logits distillation (SAKD).

method. They train the SNN with backpropagation through time (BPTT) and greatly reduce the timesteps. However, limited by the self-accumulating dynamics (Fang et al., 2021b) and the unique spike activation function, the SG method usually suffers from the problems of gradient vanishing or exploding, and deeper SNNs are still unable to converge completely ((*e.g.*, ResNet-101/152)). Moreover, the performance of directly trained SNNs is still far behind that of ANNs.

In this paper, we seek knowledge distillation (Hinton et al., 2015) to alleviate the problem. The knowledge distillation can transfer the knowledge of the high-performance teacher to improve the weak-performance student. In addition, features and logits distillation can provide effective supervision signals in the shallow, middle and deep layers of the network to optimize the convergence of the model.

However, it remains a problem for the knowledge distillation to choose the architecture of the teacher model and student model. To explore the impact of this architecture on knowledge distillation, we use a single-stage neural architecture search (NAS) (Yu et al., 2020) to find the best student model of the fixed teacher model. As discussed in Section.3.3, we find that the teacher does not favor a particular architecture and the performance of the student with the same architecture is outstanding enough. Based on the above observations, a natural idea is to choose ANNs of the same architecture to teach SNNs, namely self-architecture knowledge distillation. Moreover, choosing the same ANNs and SNNs is not only simple enough but also brings additional benefits for SNNs training, such as weight initialization.

As a result, we proposed the Self-Architectural Knowledge Distillation (SAKD) framework. We migrate the knowledge of the pre-trained ANNs into the SNNs of the same architecture to obtain high-performance SNNs with ultra-low latency. Extensive experiments show that our method is effective enough to achieve state-of-the-art performance on mainstream datasets.

Our main contributions are as follows:

- We introduce the NAS to explore the impact of architecture in SNN distillation. Based on the above observation, we introduce the self-architecture knowledge distillation strategy to SNNs.
- We introduce the self-architecture knowledge distillation strategy to effectively migrate the knowledge of the pre-trained ANNs into the SNNs, it greatly reduces the performance gap between SNNs and ANNs of the same architecture.
- We empirically show that this simple distillation strategy can overcome the convergence difficulty of many SNNs with complex structures (*e.g.*, ResNet-101/152), which makes SNNs not limited to shallow networks.
- We conduct extensive experiments to demonstrate the superiority of our knowledge distillation framework. The proposed method is validated on on CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), ImageNet (Deng et al., 2009), and DVS-CIFAR10 (Li et al., 2017) datasets with different architectures. Compared to current state-of-the-art (SOTA) methods, we can achieve better performance while maintaining an ultra-low latency.

2 RELATED WORK

ANN-to-SNN conversion. The conversion method obtains high-performance SNNs by replacing the activation function of pre-trained ANNs with the spike activation function and sharing the weight of ANNs. However, although there is no overhead in training SNNs, this method requires a large enough number of timesteps to match the spike firing frequency with the activation value of ANNs, which limits the performance of SNNs under low latency and affects their application deployment. Some recent studies have made great efforts to reduce the latency of SNNs after conversion. Such as, Rueckauer et al. (2017) proposed a soft reset mechanism, Li et al. (2021) proposed parameter calibration, and Bu et al. (2021); Deng & Gu (2021) analyzed the source of conversion error of ANN-to-SNN and alleviated the error from different perspectives. However, these methods still require a lot of timesteps to maintain performance and are not suitable for training low-latency SNNs.

BP-for-SNN. Unlike the conversion method, which does not need to train SNNs, BP-for-SNN uses gradient backpropagation to train SNNs. Recent studies have trained SNNs by solving the non-differentiable problem of spike activation function in different ways. Among them, Wu et al. (2021a;b); Meng et al. (2022) train SNNs by accumulating the input and output of neurons to design the gradient of backpropagation, but a period of timesteps is needed to maintain high performance, and it is difficult to achieve ultra-low-latency SNNs. Surrogate gradient as a method of training SNNs by BPTT can achieve ultra-low inference latency with $T = 4$ (Fang et al., 2021b; Deng et al., 2022). However, the biggest limitation of SG is that they can only achieve a suboptimal performance of SNNs, especially in the face of the deeper network architecture (Zheng et al., 2021; Fang et al., 2021a), gradient disappearance and explosion phenomenon begin to appear, which made it difficult for SNNs to be effectively optimized and limited to the shallow network.

Knowledge Distillation. Knowledge distillation (KD) was proposed by Hinton et al. (2015) as a method of model compression and followed by many works. Romero et al. (2014) first proposed knowledge distillation through features transfer and a series of works improved this method (Heo et al., 2019; Chen et al., 2021; Yang et al., 2022). Yuan et al. (2020) reviewed the relationship between knowledge distillation and label smoothing, then pointed out that KD is not only a model compression method, but also a regularization method. Touvron et al. (2021); Li et al. (2022) took this view further and introduced ANN's knowledge to help the training process of Vision Transformers. Some recent work has attempted to introduce distillation of knowledge into the SNNs domain (Kushawaha et al., 2021; Takuya et al., 2021). However, they mostly start from a model compression perspective and fail to achieve competitive performance. In this paper, we explored the knowledge distillation paradigm of ANN as the teacher and SNN as the student, introducing features-based and logits-based knowledge distillation to significantly improve SNNs' performance.

Neural Architecture Search. NAS is designed to automatically find the optimal architecture without manual design. Some recent advances define a super network (supernet) including all architectures (subnet) through weight sharing strategy and train the supernet only once (one-shot NAS). This results in a significant reduction in the computational overhead of NAS, making the lightweight of NAS a success. Li & Talwalkar (2020) propose that random search has comparable performance and is simple enough compared to other complex search strategies. Guo et al. (2020) proposed the single-path one-shot NAS, which uses simple search space to achieve competitive performance. Yu et al. (2020) use a series of training methods to achieve competitive performance without the need for fine-tuning or retraining of subnets after the completion of supernet training, which implements the single-stage NAS. Some recent work on NAS combined with KD has also led to many novel perspectives (Liu et al., 2020). Li et al. (2020) use knowledge distillation to improve NAS performance. Liu et al. (2020) found that structure knowledge was also very important in the process of KD, and the specific teacher model would be biased towards the student model with a specific structure. This also inspired our exploration of architecture in distillation.

3 METHOD

3.1 SPIKING NEURON MODEL

The spiking neuron is the core of the SNNs, which integrates inputs, adjusts membrane potentials, and controls the firing of spikes. In this paper, we chose LIF neurons as the basic computational unit of SNNs. First, the neuron adjusts the membrane potentials based on the input it receives at the

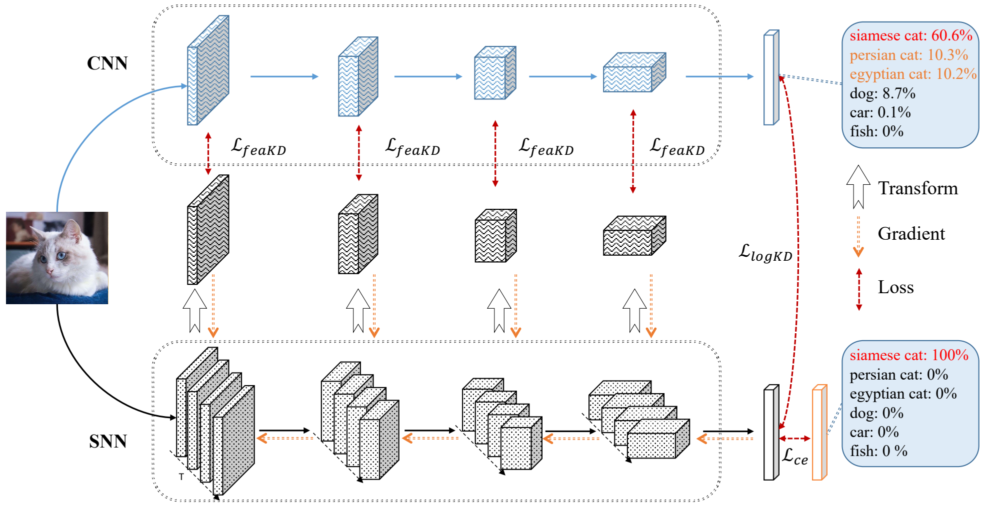


Figure 2: An illustration of the proposed method, i.e. the knowledge distillations for SNNs. We forced SNNs to learn the features and logits distribution of pre-trained ANNs of the same architecture. Since the features of SNNs and ANNs are located in discrete space and continuous space respectively, a convolutional layer is used to map the features of SNNs to the continuous space to match the features of ANNs.

current moment:

$$U^t = \lambda * U^{t-1} + X, \tag{1}$$

where λ is the delay (usually set to a hyperparameter like 0.5), U^t is the membrane potential of the neuron at time-step t, and X is the input for the current layer. Then, the neuron will choose whether to fire a spike according to the current membrane potential and threshold:

$$S^t = \Theta(U^t - V_{th}), \tag{2}$$

where S^t is the spike output of the neuron at time-step t (the output of the current layer, $S^0 = 0$). Θ is the Heaviside step function, when membrane potentials at the time-step t exceeds a threshold (usually set to 1, $V_{th} = 1$), the neuron will fire a spike ($U^t - V_{th} \geq 0 : S^t = 1, U^t - V_{th}<0 : S^{t-1} = 0$) and membrane potentials will adjust at the same time-step t. We choose the hard reset mechanism (Ledinauskas et al., 2020) to adjust membrane potentials after firing spikes:

$$U^t = U^t * (1 - S^t), \tag{3}$$

Because the step function is not derivable, we usually choose the surrogate gradient to replace it. Same to Deng et al. (2022); Rathi & Roy (2021), we choose the triangle surrogate gradient. 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, which treats ANNs as teachers and SNNs as students. To be specific, the knowledge is transferred from ANNs to SNNs with the knowledge distillation, composed of features distillation and logits distillation. The overall loss can be formulated as follow:

$$\mathcal{L}_{all} = \alpha * \mathcal{L}_{ce} + \beta * \mathcal{L}_{feaKD} + \gamma * \mathcal{L}_{logKD}. \tag{5}$$

where $\mathcal{L}_{ce}$ denotes the typical cross-entropy loss to help the model learn 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 that control the weight of different loss.

How to choose the structure of ANNs and SNNs and how to efficiently transfer knowledge from ANNs to SNNs are the necessary factors for a successful knowledge distillation of SNNs. We discussed how to choose the architecture in Section.3.3 and the specific distillation implementation form in Section.3.4.

3.3 Architecture Search Under Knowledge Distillation

Table 1: The search space of SNNs contains 36,288 subnets. BasicBlock is the component of the ResNet.

Stage	Operator	Channel	Nums of Block
1	BasicBlock	[48,64,80]	[2,3,4,5]
2	BasicBlock	[96,128,160]	[2,3,4,5]
3	BasicBlock	[192,256,320]	[2,3,4,5,6,7,8]
4	BasicBlock	[384,512,640]	[2,3,4,5]

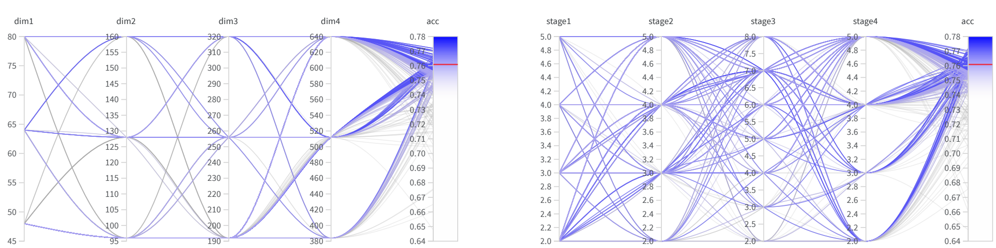
To explore the impact of this architectural knowledge on distillation, we utilize the single-stage one-shot NAS to find the best student model for a fixed teacher model. Inspired by Yu et al. (2020), we train a supernet without fine-tuning and retraining the subnets. First, the one-shot NAS will define the search space (a supernet $\mathcal{A}$ containing all subnets a), and the overall training goal is to optimize the weight of the supernet. Second, after the supernet training, the subnet with the best performance is searched.

Specifically, the subnet is selected for training each time according to a certain strategy (*i.e.*, random sampling) (Guo et al., 2020),

$$\mathcal{W}_{\mathcal{A}}^{*} = \min_{\mathcal{W}_{\mathcal{A}}}(\mathcal{L}_{train}(\mathcal{N}(a, \mathcal{W}_{\mathcal{A}}(a)))) \tag{6}$$

where $\mathcal{W}_{\mathcal{A}}$ is the overall weight of the supernet, $\mathcal{W}_{\mathcal{A}}(a)$ is the weight of the sampled subnet, $\mathcal{N}(a, \mathcal{W}_{\mathcal{A}}(a))$ denotes the sampled subnet.

Our search space is shown in Table 1. We choose ResNet-34 as the teacher, and use features and logits distillation during supernet training according to Section.3.2. After the training, we sampled 300 structures (*i.e.*, bayesian search) and evaluated their performance to explore whether the teacher favored the student with a particular structure. As shown in Figure 3, we find that the teacher does not favor a particular architecture, which means that the impact of this architecture knowledge is not so significant in the SNN distillation process.



(a) channel (b) layers

Figure 3: Performance of different architectures in the search space. The rightmost column represents the performance metrics, and we mark the lines of architectures that achieve higher performance with darker blue. The left column represents the search space of the network structure. The *red line* represents the performance of the student with the same structure as the teacher. **Left**: performance of students with the different number of channels under distillation. **Right**: performance of students with the different number of layers under distillation.

As shown in Table 2, our experiment also shows that the teacher with higher performance may not bring better improvement to the student. More importantly, the teacher with the same architecture can bring more significant improvement to the student on the large dataset than the teacher with a larger scale and higher performance.

Table 2: Different teachers for SNNs

Dataset	SNNs	ResNet Teacher Model		
		R50	R101	R152
CIFAR-100	R50	79.58	**79.62**	79.61
ImageNet	R50	**71.71**	71.03	70.62

Based on these observations, we think it is the best choice to choose ANNs with the same architecture as SNNs to guide the training of SNNs, which is not only simple and flexible but also has some additional benefits (*e.g.*, weight-initialization in Section.A.2).

3.4 SELF-ARCHITECTURAL KNOWLEDGE DISTILLATION

In this section, we discuss in detail how to efficiently transfer the knowledge of ANNs into SNNs with the same architecture. As shown in Figure 2, the features of intermediate layers in SNNs are aligned to the corresponding features in ANNs, and the logits of final layers in SNNs are constrained by the logits of ANNs. The $\mathrm{Transform}$ is used to map the discrete features of SNNs to the continuous space, and it can also solve the problem of dimension mismatch of the features when ANNs and SNNs with different architectures are encountered.

Logits Knowledge Distillation. We first introduce the logits knowledge distillation by letting the student learn the predicted distribution of the teacher. As shown in Figure 2, the predicted distribution of the teacher contains more information (*e.g.*, the correlation between categories) than the traditional one-hot label. We utilize the KL-divergence loss to restrain the distribution of student output to the teacher distribution as follows:

$$\mathcal{L}_{logKD} = \tau^2 \sum p^t_\tau log(\frac{p^t_\tau}{p^s_\tau}), \qquad p^s_\tau(i) = \frac{exp(p^s(i)/\tau)}{\sum exp(p^s/\tau)}, \tag{7}$$

where p^t_τ and p^s_τ represents the predicted distribution of ANNs and SNNs, respectively. τ is the temperature to smooth the logits.

Features Knowledge Distillation. We introduce features knowledge distillation additionally, which transmits the teacher's knowledge by forcing the student to imitate the teacher's features map.

Due to the teacher and the student being of the same architecture, we choose to directly match the original features of the ANNs instead of indirectly designing new knowledge forms (Ahn et al., 2019; Tung & Mori, 2019) to transfer knowledge.

Features transformation mainly solves the problem that the teacher and the student have different features dimensions (Chen et al., 2021; Heo et al., 2019; Yue et al., 2020; Zagoruyko & Komodakis, 2016). In this work, although the features of ANNs and SNNs share identical architecture, there is a natural gap between the binary features of SNNs and the continuous space features of ANNs. So we introduce the additional convolutional transformation to map the features to the same content space, which can be formulated as follow:

$$\hat{\mathcal{F}}_s = \mathcal{T}_s(\mathcal{F}_s) = \mathrm{BN}(\mathrm{Conv}(\frac{1}{T}\sum_T \mathcal{F}_s)), \quad \hat{\mathcal{F}}_t = \mathcal{T}_t(\mathcal{F}_t) = \mathcal{F}_t, \tag{8}$$

where $\mathcal{F}_s$ and $\mathcal{F}_t$ are the features of SNNs and ANNs, $\mathcal{T}_s$ and $\mathcal{T}_t$ denote the feature transform used for SNNs and ANNs, T represents the time dimension of $\mathcal{F}_s$, $\mathrm{BN}(\cdot)$ and $\mathrm{Conv}(\cdot)$ represent the convolutional layer and batch normalization layer, respectively. The convolutional layer maps the SNN's features to a continuous space and does not transform ANN's features to preserve the original information.

In addition, previous studies have shown that the distillation position and the distance function will directly affect the effect of features distillation, and most of the features distillation positions will select the end of each stage (Zagoruyko & Komodakis, 2016; Yue et al., 2020) and the distance function usually adopts $\mathcal{L}_2$ distance or $\mathcal{L}_1$ distance. Meanwhile, some studies (Heo et al., 2019) have shown that the selection of features before or after the activation function will also have a great impact on the distillation effect. We discussed the influence of feature transformation and distance function on features distillation in detail in Section.4.3. By default, we adopt $\mathcal{L}_2$ distance as our distance function, that is, the feaKD loss is:

$$\mathcal{L}_{feaKD} = \|\hat{\mathcal{F}}_s - \hat{\mathcal{F}}_t\|^2 \tag{9}$$

4 EXPERIMENTS

In this part, we perform vast experiments with different architectures on various datasets including CIFAR-10/100, ImageNet, and DVS-CIFAR10 to validate the efficiency of our proposed method. We discuss comparisons with other methods in Section.4.1, and the performance of distillation when the network gets deeper in Section.4.2. We show ablation experiments and some experimental details in Section.4.3 and Section.4.4.

Table 3: Comparisons with current state-of-the-art methods. All of our models use the ResNet structure with only 4 timesteps.

Dataset	Method		Model	Time-step	Accuracy
CIFAR-10	(Han et al., 2020)	ANN2SNN	VGG-16	2048	93.63
	(Li et al., 2021)	ANN2SNN	VGG-16	32	93.71
	(Bu et al., 2021)	ANN2SNN	ResNet-18	4	90.43
	(Zheng et al., 2021)	STBP-tdBN	ResNet-19	6	93.16
	(Deng et al., 2022)	TET	ResNet19	6	94.50
	(Guo et al., 2022)	Rec-Dis	ResNet-19	6	95.55
	(Meng et al., 2022)	DSR	pre-ResNet-18	20	95.40
	ours		ResNet19	**4**	**96.06**
CIFAR-100	(Han et al., 2020)	ANN2SNN	VGG-16	2048	70.93
	(Li et al., 2021)	ANN2SNN	VGG-16	256	77.68
	(Bu et al., 2021)	ANN2SNN	VGG-16	32	77.01
	(Zheng et al., 2021)	STBP-tdBN	ResNet-19	6	71.72
	(Deng et al., 2022)	TET	ResNet-19	6	74.72
	(Guo et al., 2022)	Rec-Dis	ResNet-19	6	74.10
	(Meng et al., 2022)	DSR	pre-ResNet-18	20	78.50
	ours		ResNet-19	**4**	**80.10**
ImageNet	(Han et al., 2020)	ANN2SNN	ResNet-34	4096	69.89
	(Li et al., 2021)	ANN2SNN	ResNet-34	32	64.54
	(Bu et al., 2021)	ANN2SNN	ResNet-34	32	67.37
	(Zheng et al., 2021)	STBP-tdBN	ResNet-34	6	63.72
	(Fang et al., 2021a)	SEW	SEW-ResNet-50	4	67.78
	(Deng et al., 2022)	TET	ResNet-34	6	64.79
			SEW-ResNet-34	4	68.00
	(Guo et al., 2022)	Rec-Dis	ResNet-19	6	67.33
	(Meng et al., 2022)	DSR	pre-ResNet-18	20	67.74
	ours		ResNet-18	**4**	**68.04**
			ResNet-34		**70.04**
			ResNet-50		**71.71**
DVS-CIFAR10	(Zheng et al., 2021)	STBP-tdBN	ResNet-19	10	67.80
	(Fang et al., 2021a)	SEW	Wide-7B-Net	16	74.40
	(Deng et al., 2022)	TET	VGG-11	10	**83.17**
	(Meng et al., 2022)	DSR	VGG-11	20	77.27
	(Guo et al., 2022)	Rec-Dis	ResNet-19	10	72.42
	ours		VGG-11	**4**	81.50
			ResNet-19		**80.30**

4.1 Comparison with the other method

As presented in Table 3, the proposed framework is compared to previous works on four datasets.

CIFAR-10/100. The ResNet-19 architecture is employed in our experiments on CIFAR-10 and CIFAR-100. We can achieve the top-1 accuracy of 96.06% and 80.10% on CIFAR-10 and CIFAR-100, respectively which exceeds all the other compared methods. Especially, on CIFAR-100, the proposed method outperforms the current best method by 5.38%.

ImageNet. We choose the representative ResNet-18/34/50 to verify our algorithm on ImageNet. Our method outperforms all the other compared methods and achieves the best performance with only 4 timesteps. Especially, the ResNet-50 trained with our method can achieve 71.71% top-1 accuracy with timesteps of only 4.

DVS-CIFAR10. We adopt the VGG-11 and ResNet-19 architecture on DVS-CIFAR10, achieving the top-1 accuracy of 81.50% and 80.30% with 4 timesteps, respectively. Although the TET method adopts the VGG-11 and can achieve an accuracy of 83.17%, its timesteps are more than twice those of ours. It is worth noting that our approach can bring 6.80% and 6.30% improvement in baseline performance of VGG-11 (*i.e.*, 75.20% w/o KD) and ResNet-19 (*i.e.*, 73.50% w/o KD) respectively.

4.2 DEEPER SPIKING RESNET UNDER SAKD

In this part, we adopt a series of experiments with ResNet of different depths on CIFAR to verify the effect of distillation on the deeper network. For each structure, the result of ANNs, SNNs trained with SG, and SNNs trained with our SAKD are listed in Table 4, it can be seen that for deep nets (ResNet-101/152), the traditional SNN suffers a severe degradation problem, while our SAKD converges normally with the help of ANNs. As stated by Fang et al. (2021a), the spiking activation cannot achieve the identity mapping, and the residual connection in SNNs cannot solve the gradient problem. However, knowledge distillation can significantly alleviate this problem by transferring the knowledge of ANNs to optimize the shallow, middle and final layers of deep SNNs.

Table 4: Performance of SNNs with different depths on CIFAR.

	CIFAR-10			CIFAR-100		
Model	ANN	SNN	SAKD-SNN	ANN	SNN	SAKD-SNN
ResNet-18	95.65	94.33	94.99	78.35	75.03	77.85
ResNet-34	95.69	93.50	95.15	79.30	71.72	78.50
ResNet-50	95.85	93.54	95.55	80.49	70.34	79.58
ResNet-101	95.92	55.83	95.13	80.83	35.72	79.72
ResNet-152	96.38	10.00	95.36	81.36	10.00	79.21

4.3 ABLATION EXPERIMENT

Features-based Distillation. To select the best setting for feature-based distillation, we take a lot of experiments to analyze the influence of features transformation, distance function, and distillation position (in Section.A.3) on features distillation according to Section.3.4.

Table 5 shows the necessity of processing the original features map. Conv represents a convolutional layer to project SNNs features into continuous space (parametric). Norm (Liu et al., 2022) represents that the original features of ANNs and SNNs are normalized (non-parametric). Both methods can significantly improve distillation performance, and we default to using the method with a convolutional layer to project SNNs features.

Table 6 shows the effect of different distance functions on distillation. We choose the loss coefficient from { β = 1, 10, 100 } to control the loss value obtained by different distance functions. The results show that different distance functions can have good performance, and we adopt $\mathcal{L}_2$ by default.

Table 5: Effect of features transform on CIFAR-100.

	Baseline	None	Conv	Norm
ResNet-18	75.03	76.00	**77.78**	76.93
ResNet-34	71.72	75.63	**78.05**	77.88

Table 6: Effect of distance function on CIFAR-100.

Function	$\mathcal{L}_2$	$\mathcal{L}_1$	$\mathcal{L}_{KL}$
ResNet-18	**77.78**	77.00	77.47
ResNet-34	78.05	**78.14**	76.04

Logits-based Distillation. Although historically the best distillation methods have tended to be features distillation (Zhao et al., 2022), some recent studies have shown that logits distillation can also provide significant performance improvements (Beyer et al., 2022; Ridnik et al., 2022) and even be more efficient than features-based methods (Hsu et al., 2022). As shown in Table 7, on the large dataset (*i.e.*, ImageNet), logits distillation alone is better than features distillation alone in most cases and the best performance can be achieved when both are used together.

4.4 HYPERPARAMETER.

According to Ridnik et al. (2022); Kim et al. (2021), we set the logits loss hyper-parameter in Sec.3.2 as: { τ = 20, α = 0, γ = 1 } for CIFAR, { τ = 1, α = 1, γ = 10 } for ImageNet. We consider the influence of features distillation factor β on the performance of the CIFAR-100, as shown in Table 8. We set { β = 100 } for CIFAR, { β = 10 } for ImageNet (In general, the loss factor β on ImageNet is 0.1 times of that on CIFAR, which will perform well). It is worth noting that we find it unnecessary or even harmful to force SNNs to make efforts to simulate ANN's features on ImageNet. As Kim et al. (2021) found out, the teacher's information contains too much noise because the model is

Table 7: Comparison of ImageNet classification performance between different distillation modes and weight-initialization.

Initiation	Logit	features	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	**70.04**	**71.71**

difficult to fit the large dataset completely. Cho & Hariharan (2019) also pointed out that when the capacity gap between the teacher model and the student model is too large, the continuous imitation of the teacher's knowledge will limit the ability of students. But this problem can be avoided by ending the distillation process early. Interestingly, Chen et al. (2022) also found that matching ANN's features in the early training stage and canceling it in the later stage would better improve the performance of ViT. Therefore, we greatly reduce the features loss coefficient β in the middle of training on ImageNet (*i.e.*, set β to 0.01 after 10 epochs during the entire 120 epoch training cycle).

Table 8: Ablation study results on the hyperparameter β in Equation 5.

β	0	1	5	10	**100**	500	1000
ResNet-18	75.04	75.98	76.73	77.22	**77.85**	unstable	unstable

4.5 Visualization

Figure 4 shows the features in the intermediate layer and the gradient-weighted class activation mapping (GradCAM) of ANNs and SNNs. The first row is the features of ANNs. The second row represents SNNs without KD, which is difficult to produce rich features. The last row shows that SNN's features are more informative by mimicking ANN's features, and GradCAM also shows that SNNs are more focused on the core region after distillation.

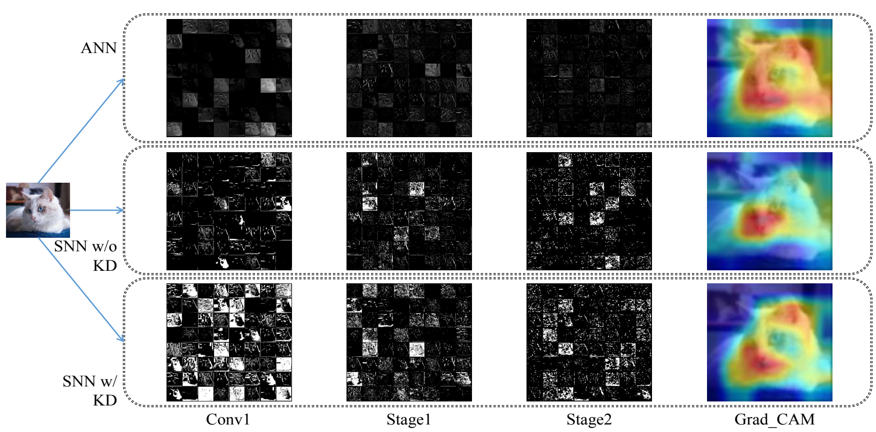


Figure 4: Visualization of features and GradCAM of ANNs and SNNs.

5 Conclusion

In this paper, we propose a self-architected knowledge distillation strategy to train SNNs with ultra-low latency and high performance by migrating the knowledge from ANNs to SNNs with the same architecture. Extensive experimental results with different models on different datasets show that our method can significantly improve the SNNs performance. Since most existing SNNs are based on ANNs' architecture, we believe that SAKD can provide a new training paradigm for SNNs.

6 Reproducibility Statement.

We describe the experimental details in the Appendix.A. The experimental results in this paper are reproducible. We explain the details of model training and experimental setting in the main text and supplement it in the appendix. Our codes of Neural Architecture Search and Knowledge Distillation are uploaded as supplementary material and will be available on GitHub after review.

References

Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9163–9171, 2019.

Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. 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, 2022.

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. In *International Conference on Learning Representations*, 2021.

Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5008–5017, 2021.

Xianing Chen, Qiong Cao, Yujie Zhong, Jing Zhang, Shenghua Gao, and Dacheng Tao. 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, 2022.

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

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*, pp. 248–255. Ieee, 2009.

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

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.

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, 2021a.

Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. 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, 2021b.

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

Yufei Guo, Xinyi Tong, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Zhe Ma, and Xuhui Huang. 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, 2022.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European conference on computer vision*, pp. 544–560. Springer, 2020.

Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. 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, 2020.

Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1921–1930, 2019.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

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

Dongsung Huh and Terrence J Sejnowski. Gradient descent for spiking neural networks. *Advances in neural information processing systems*, 31, 2018.

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

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

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)*, pp. 4536–4543. IEEE, 2021.

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

Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in neuroscience*, pp. 119, 2020.

Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1989–1998, 2020.

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.

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

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, pp. 367–377. PMLR, 2020.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, pp. 6316–6325. PMLR, 2021.

Tao Liu, Xi Yang, and Chenshu Chen. Normalized feature distillation for semantic segmentation. *arXiv preprint arXiv:2207.05256*, 2022.

Yu Liu, Xuhui Jia, Mingxing Tan, Raviteja Vemulapalli, Yukun Zhu, Bradley Green, and Xiaogang Wang. 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, 2020.

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

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*, pp. 12444–12453, 2022.

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.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Nitin Rathi and Kaushik Roy. 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*, 2021.

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

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

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

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

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.

Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31, 2018.

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)*, pp. 1–3. IEEE, 2021.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1365–1374, 2019.

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

Hao Wu, Yueyi Zhang, Wenming Weng, Yongting Zhang, Zhiwei Xiong, Zheng-Jun Zha, Xiaoyan Sun, and Feng Wu. Training spiking neural networks with accumulated spiking flow. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10320–10328, 2021a.

Jibin Wu, Yansong Chua, Malu Zhang, Guoqi Li, Haizhou Li, and Kay Chen Tan. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021b.

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.

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

Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*, pp. 702–717. Springer, 2020.

Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3903–3911, 2020.

Kaiyu Yue, Jiangfan Deng, and Feng Zhou. Matching guided distillation. In *European Conference on Computer Vision*, pp. 312–328. Springer, 2020.

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

Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11953–11962, 2022.

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*, volume 35, pp. 11062–11070, 2021.

A Appendix

A.1 Datasets and Implementation Details.

CIFAR. It contains 60,000 images with the resolution of 32×32. The training and test sets contain 50,000 and 10,000 images respectively. CIFAR-10 and CIFAR-100 contain 10 and 100 categories, respectively. We apply standard data augmentation to the input image: random cropping and flipping, normalization. For shallow networks, such as ResNet-18/19/34, we used SGD optimizer to train 200 epochs, and the learning rate was 0.1 cosine decayed to 0. 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. We set the hyper-parameters according to Section.4.4, and the distillation position is selected according to Section.A.3.

ImageNet. It contains a total of about 1.2 million training images, 50,000 validation images, and 1000 categories. We apply standard data augmentation to the input image: random cropping and flipping, normalization. We used the AdamW optimizer to train 120 epochs for all models. 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 then decays cosinely to 0. We choose to match the features after each stage and set the hyperparameters according to Section.4.4 for all models.

DVS-CIFAR10. It contains 10,000 images with the resolution of 128×128, which are generally divided into 9,000 images for the training set and 1,000 images for the test set. Similar to the Meng et al. (2022) processing method. We simply reduced the resolution of the input image from 128×128 to 48×48 and apply random cropping. For all models, we used the SGD optimizer to train 200 epochs, and the learning rate was 0.1 cosine decayed to 0.

A.2 Weight-initialization Training on ImageNet.

Some recent studies (Rathi et al., 2020; Rathi & Roy, 2021; Meng et al., 2022) show that loading the weight of pre-trained ANNs before the start of SNNs training can accelerate convergence and improve performances. We consider the influence of this weight initialization on distillation and investigate the influence of ANN's weights with different performances on training: PyTorch (Paszke et al., 2017) and Timm (Wightman et al., 2021), respectively. The results are shown in Table 9. Loading the pre-trained weight of ANNs can significantly improve the performance of shallow SNNs (*i.e.*, Resnet-18/34), but it cannot guarantee that deeper SNNs will benefit from it (*i.e.*, ResNet-50). It is worth noting that our method achieves the best performance under weight initialization. Unfortunately, we observed that weight initialization hurts performance on small datasets (*e.g.*, CIFAR) under SAKD. Hence, like Meng et al. (2022), we only use this method on ImageNet.

Table 9: Comparison on ImageNet classification between the different pre-trained weights.

	Model	ANN	SNN	weight	weight+SAKD
torch	ResNet-18	69.8	61.13	63.49	65.37
	ResNet-34	73.3	64.40	65.05	67.59
	ResNet-50	76.1	67.45	67.99	70.71
timm	ResNet-18	71.5	61.13	65.00	**66.09**
	ResNet-34	76.4	64.40	66.57	**70.04**
	ResNet-50	80.4	67.45	66.21	**71.71**

A.3 Ablation study on distillation position.

Table 10 shows the performance of different distillation positions. Because the SNNs share the same architecture as ANNs, we are free to choose any layer of them to align their features. We compare two strategies to determine the alignment position. *Stage* means we align features after each entire stage, while *Block* means features after each single block (*e.g.*, bottleneck or basicblock) are aligned. Besides, we also explore whether the features-based distillation should be put before or after the spike in experiments. The results show that in shallow networks, distillation for each stage and before-spike may be a better choice. However, in deeper network architectures (such as ResNet-101/152), only matching the features after the block and activation function can ensure the normal convergence of SNNs (*e.g.*, ResNet-101:79.41% (Block and After-spike) v.s. unstable (Block and Before-spike) v.s. unstable (Stage)).

Table 10: Effect of different distillation positions on CIFAR-100. C and S respectively represent the baseline performance of ANNs and SNNs. Stage and Block denote whether we align features after each stage or each block. For example, the ResNet-18 structure contains four Stage modules, and each Stage contains two Block modules.

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