EnOF-SNN: Training Accurate Spiking Neural Networks via Enhancing the Output Feature

Yufei Guo[*], **Weihang Peng**[*], **Xiaode Liu, Yuanpei Chen, Yuhan Zhang, Xin Tong, Zhou Jie, Zhe Ma**[†]
Intelligent Science & Technology Academy of CASIC
yfguo@pku.edu.cn, pengweihang812@163.com, mazhe_thu@163.com

Abstract

Spiking neural networks (SNNs) have gained more and more interest as one of the energy-efficient alternatives of conventional artificial neural networks (ANNs). They exchange 0/1 spikes for processing information, thus most of the multiplications in networks can be replaced by additions. However, binary spike feature maps will limit the expressiveness of the SNN and result in unsatisfactory performance compared with ANNs. It is shown that a rich output feature representation (*i.e.*, the feature vector before classifier) is beneficial to training an accurate model in ANNs for classification. We wonder if it also does for SNNs and how to improve the feature representation of the SNN. To this end, we materialize this idea in two special designed methods for SNNs. First, inspired by some ANN-SNN methods that directly copy-paste the weight parameters from trained ANN with light modification to homogeneous SNN can obtain a well-performed SNN, we use rich information of the weight parameters from the trained ANN counterpart to guide the feature representation learning of the SNN. In particular, we present the SNN's and ANN's feature representation from the same input to ANN's classifier to product SNN's and ANN's outputs respectively and then align the feature with the KL-divergence loss as in knowledge distillation methods, called $\mathcal{L}_{\mathrm{AF}}$ loss. It can be seen as a novel and effective knowledge distillation method specially designed for the SNN that comes from both the knowledge distillation and ANN-SNN methods. Second, we replace the last Leaky Integrate-and-Fire (LIF) activation layer as the ReLU activation layer to generate the output feature, thus a more powerful SNN with full-precision feature representation can be achieved but with only a little extra computation. Experimental results show that our method consistently outperforms the current state-of-the-art algorithms on both popular non-spiking static and neuromorphic datasets.

1 Introduction

Recently, convolutional neural networks (CNNs) have become extremely popular due to their performing more and more well in diverse fields including pattern recognition He et al. (2016); Simonyan & Zisserman (2014), object detection Girshick (2015); Ren et al. (2016); Ming et al. (2023), language processing Chen et al. (2016), robotics Levine et al. (2015), and so on. However, these full-precision CNN models make them power-hungry and tedious for real-world deployment. The spiking neural network (SNN), which aims to mimic the behavior of the human brain, has become a promising energy-efficient architecture to substitute for the CNN in some specific scenarios Guo et al. (2022c); Rathi et al. (2020); Zhu et al. (2022); Wu et al. (2019a); Zhang et al. (2020); Jeffares et al. (2021);

[*] Equal Contributions.
[†] Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

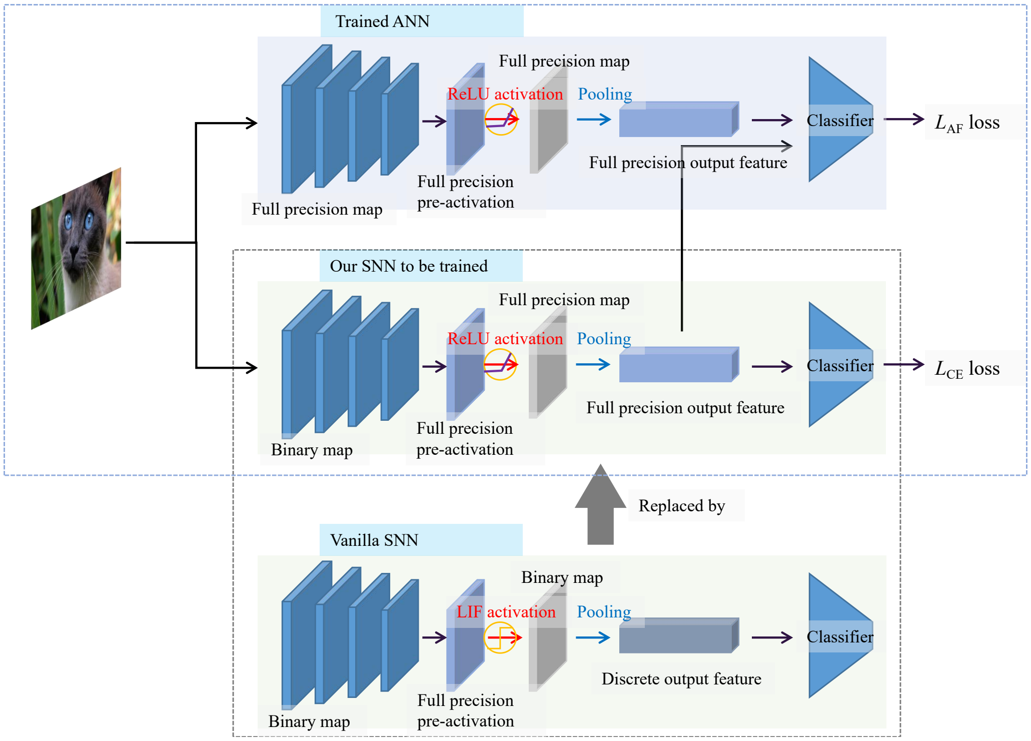


Figure 1: The overall workflow of the proposed method. To enhance the output feature representation, the $\mathcal{L}_{AF}$ loss is used to transmit the important information of output feature of the ANN to that of the SNN counterpart and the last LIF activation neuron layer is replaced by ReLU activation in our SNN structures.

Guo et al. (2022b, 2024b); Wang et al. (2023, 2022); Zhou et al. (2023). It adopts 0/1 spikes to transmit information. Benefitting from such an information processing paradigm, the multiplications of activations and weights of SNNs can be replaced by additions, thus enjoying low power consumption Guo et al. (2023a). Furthermore, on neuromorphic hardwares Akopyan et al. (2015), when there is no spike coming, the SNN neuron will remain inactive thus requiring significantly lower energy.

Despite the SNN being more energy-efficient compared with the ANN, it suffers unsatisfactory task performance. Binary spike feature maps of the SNN will result in limited expressiveness and large accuracy degradation compared with full-precision feature maps of the ANN Guo et al. (2023a, 2022a). Although the accuracy degradation problem can be mitigated by increasing the timesteps of the SNN, the inference burden will be increased too with the timestep increasing. It is shown crucial that output feature representation is rich and powerful for training a highly accurate model for the classification in recent studies in ANNs Chen et al. (2020); He et al. (2019); Kang et al. (2019). Here, we also expect that improving the output feature representation is beneficial for the accuracy of the SNN. To this end, we advocate using trained ANN's rich output feature to guide the learning of output feature representation of SNN. However, different from other SNN distillation methods that align the output feature of the SNN with that of the ANN counterpart directly, we adopt the trained ANN's classifier to guide the information transfer. Using the ANN's classifier directly is inspired by some ANN-SNN methods that directly copy-paste the weight parameters from trained ANN with light modification to homogeneous SNN can obtain a well-performed SNN Han et al. (2020); Li et al. (2021a). Therefore, we expect that utilizing the homogeneous ANN's some modules to guide the learning of SNN will be useful. In particular, for the same input, we wish the SNN's and ANN's feature representation to produce the same output when passed through the same ANN's classifier and the idea can be achieved by a simple Kullback-Leibler (KL)-divergence loss.

Furthermore, we also suggest that the last firing activation neuron layer in SNN could be replaced by ReLU, called RepAct, thus the output feature representation will be full precision and the expression ability will be improved but with a trivial additional cost compared with the vanilla SNN. It is worth noting that spiking neural networks do have a certain number of non-binary feature maps. For example, many works use original images as the input of SNNs, thus the multiplications of the first layer can not be replaced by additions in SNNs Rathi & Roy (2020); Guo et al. (2022b); Baltes et al. (2023); Li & Zeng (2022), but the accuracy would boost to a high level. With a trivial additional cost, the RepAct can also improve the SNN greatly, hence it could be seen as a practical choice for many tasks too.

To sum up, we find that the expression ability of the output feature of the SNN is limited. Under this understanding, we focus on enhancing the output feature representation of the SNN in the paper. To this end, we propose two very simple but effective methods. First, we utilize the trained ANN to guide the learning of the output feature representation of the SNN. In specific, since the feature representation of the ANN is more powerful, we propose to align the output feature layer of SNN with that of ANN to learn its rich capabilities. Because direct feature representation matching does not take into account the classification task at hand and ignores the high-level representation information, thus will adversely affect SNN classification results if the matching strategy design is not good enough, we utilize the ANN's pre-trained classifier to align the output feature of the SNN with that of the ANN counterpart to enrich the SNN's feature representation, called $\mathcal{L}_{\mathrm{AF}}$ loss. This idea is based on the observation that directly copying-pasting the weight parameters from trained ANN to homogeneous SNN is always enough to obtain a well-performed SNN in ANN-SNN methods. In this sense, it can be seen as a simple and effective knowledge distillation method specially designed for SNN born in ANN-SNN. In this way, the feature representation and classification information can both be considered. Second, we replace the last LIF activation neuron layer with the ReLU activation layer to produce the full precision output feature with only a little extra computation, named RepAct. The overall workflow can be seen in Fig.1. Our contributions can be summarized as follows:

- We propose to use the well-trained ANN to guide and improve the learning of the output feature representation of the SNN. Along with taking into account the classification task, the $\mathcal{L}_{\mathrm{AF}}$ loss is proposed to weigh the discrepancy of the two outputs that come from the ANN's and SNN's feature representations for the same image by feeding in the same trained ANN's classifier. It can be seen as a simple and effective knowledge distillation method specially designed for SNN derived from a combination of advantages of knowledge distillation and ANN-SNN. With the $\mathcal{L}_{\mathrm{AF}}$ loss, the SNN's feature representation can be extremely enhanced and results in high accuracy.
- We also advise replacing the last LIF activation neuron layer in the SNN with ReLU activation layer to obtain the full precision output feature representation. This improvement will increase the expression ability of the output feature layer thus improving accuracy greatly, but with a trivial extra computation.
- We evaluate our methods on both famous static and spiking datasets. Extensive experimental results show that our method consistently outperforms state-of-the-art methods in various experimental settings.

2 Related Work

In view of the success of deep CNNs, the early development of deep SNNs benefits from the experience of CNNs. Using the trained ANN to guide the generation of high-precision SNN is one of the mainstream research at present. There are three main routes to realizing this idea.

First, a simple and effective way to use ANNs is converting a well-trained ANN model to an SNN model Han et al. (2020); Sengupta et al. (2018); Li et al. (2021a); Ding et al. (2021); Hao et al. (2023a,b). This method trains an accurate ANN with ReLU activation first and then uses the ANN network parameters to construct a special SNN network that enjoys the same structure as the ANN and adopts the average output through multiple timesteps as the final result. The converted SNN usually performs worse than the original ANN, since there are more or fewer conversion errors between the ANN and SNN. To bridge this gap as much as possible, many effective methods have been proposed, such as long inference time Girshick (2015), threshold rescaling Sengupta et al. (2018), soft reset Han et al. (2020), threshold shift Li et al. (2021a), and the quantization clip-floor-shift activation function Ding et al. (2021). However, the conversion method is restricted to rate-coding and the Integrate-and-Fire (IF) model, and needs long timesteps to obtain a satisfied accuracy.

Second, using the well-trained ANN model's parameters to initial the nascent SNN has been another fundamental line of research. For example, in Rathi et al. (2020) and Rathi & Roy (2020), a trained ANN is converted to an SNN first and then the weights of the converted SNN are fine-tuned further with the training method. Initialed with the trained ANN, the SNN shows a faster convergence than that from random initialization. Though this method usually needs fewer timesteps than the conversion method, it does not show obvious advantages for deep models.

Third, there are also some works that apply the knowledge distillation technique in the SNN field that to use ANN to distill the SNN Kushawaha et al. (2021); Takuya et al. (2021); Xu et al. (2023b,a); Guo et al. (2023c); Zhang et al. (2024). However, as has been comprehensively and systematically studied in Tian et al. (2020), most of the knowledge distillation methods perform poorly without well-designed hyper-parameters. These knowledge distillation methods for SNNs have also verified this consensus. They all adopt complex multi-stage distillation manner and only report the results on small-scale datasets. Providing a simple but effective knowledge distillation for the SNN is difficult.

Despite the above methods providing us with a lot of valuable insights, we still think there are some other improvements worth further consideration. First, these prior methods merely use the parameters of the trained ANN to guide the designing of the parameters of the SNN or adopt a complex framework to transfer knowledge from ANN to SNN individually, designing a simple and effective knowledge distillation method specifically for SNNs combining these useful experience of SNN knowledge distillation and ANN-SNN methods simultaneously are ignored. Second, these SNN methods all adopt similar architectures as ANNs that only modify the ANN activation layers as spike neuron layers simply but without considering the advantages and disadvantages of the SNNs comprehensively. In the paper, aiming at improving the output feature representation of the SNN, we will take care of these two problems well.

3 Preliminary and Methodology

This section first introduces the proposed method of aligning the SNN's output feature with the ANN's output feature. Then the Leaky Integrate-and-Fire (LIF) neuron model, why its output feature's expression ability is limited, and how to replace the LIF activation layer with the ReLU activation layer will be introduced in detail. Finally, some key details for training the SNN will be given.

3.1 L_{AF} Loss: Output Feature Alignment Loss

To improve the output feature representation ability, we advocate using the well-trained ANN to guide and improve the learning of the output feature representation of the SNN. A natural idea is to align the output feature of the SNN to that of its ANN counterpart directly like knowledge distillation methods, given by

$$\arg\min_{\mathbf{W}_{\mathrm{s}}}(||\boldsymbol{h}_{\mathrm{a}} - \boldsymbol{h}_{\mathrm{s}}||). \tag{1}$$

Where $\mathbf{W}_{\mathrm{s}}$, $\boldsymbol{h}_{\mathrm{a}}$, and $\boldsymbol{h}_{\mathrm{s}}$ represent the parameters of the SNN, the output feature of the ANN, and the output feature of the SNN.

However, we found that this natural idea does not always work, which has also been validated by Tian et al. (2020). The work has investigated most of the famous knowledge distillation methods and found that they perform poorly without well-designed hyper-parameters.

The reason why direct distillation performs worse in SNNs comes from that: 1) discrete output feature of the SNN makes its representation space much different from that of the full precision ANN, and directly aligning their output features simply will ignore the difference of their representation spaces and induce too much constraint in the SNN optimization, which may lead to a configuration far from the global optimal; 2) this design also treats each dimension in the feature space independently, and ignores the high-level interdependencies of the feature representation; and 3) it does not take into account the classification task at all, which conflicts with the final optimization goal.

To consider the above problems comprehensively, we present a simple but effective method, that uses the ANN's classifier to guide the learning of the SNN's output feature. This is inspired by some ANN-SNN methods that directly copy-paste the weight parameters from trained ANN to homogeneous SNN can obtain a well-performed SNN. Hence, we guess that utilizing a homogeneous ANN's some elements to guide the learning of SNN may be useful. In specific, we fed the SNN's output feature to the ANN's pre-trained classifier to generate the output, $\boldsymbol{P}_{\rm s}$. At the same time, we obtain the ANN's output, $\boldsymbol{P}_{\rm a}$ from the same input image as the SNN. Then we utilize the KL-divergence loss to align the student output to the teacher output as follows:

$$\mathcal{L}_{\rm AF} = \sum \texttt{SoftMax}(\boldsymbol{P}_{\rm a})\log(\frac{\texttt{SoftMax}(\boldsymbol{P}_{\rm a})}{\texttt{SoftMax}(\boldsymbol{P}_{\rm s})}). \tag{2}$$

In this situation, since $\boldsymbol{P}_{\rm s}$ and $\boldsymbol{P}_{\rm a}$ come from the same ANN's pre-trained classifier, if $\boldsymbol{h}_{\rm s}$ is same as $\boldsymbol{h}_{\rm a}$, $\texttt{SoftMax}(\boldsymbol{P}_{\rm s})$ will become same as the $\texttt{SoftMax}(\boldsymbol{P}_{\rm a})$, then the $\mathcal{L}_{AF}$ will become zero and the smallest. At the same time, there are still other chances that can make $\mathcal{L}_{AF}$ small too. That is, $\mathcal{L}_{AF}$ will not lose the consideration that $\boldsymbol{h}_{\rm s} = \boldsymbol{h}_{\rm a}$, but will also take care of other alignment situations of $\boldsymbol{h}_{\rm s}$ and $\boldsymbol{h}_{\rm a}$. Furthermore, Eq. 2 also takes the classification task into the consideration. Hence, Eq. 2 can be seen as a more general alignment strategy than Eq. 1 designed specifically for SNNs. Finally, taking classification loss together, the total loss can be written as

$$\mathcal{L}_{\rm Total} = \mathcal{L}_{\rm CE} + \lambda\mathcal{L}_{\rm AF}, \tag{3}$$

where $\mathcal{L}_{\rm CE}$ is the cross-entropy loss, and λ is a coefficient to balance the classification loss and the alignment loss.

3.2 RepAct: Replace the Last Activation Layer

As abovementioned, we argue that the last spike neuron layer will limit the performance of the SNN. To better depict this, we give the specific form of a spiking neuron first. Here, we adopt the well-known Leaky Integrate-and-Fire (LIF) neuron model for an SNN. The LIF neuron adjusts the membrane potential based on the input and its membrane potential at the previous moment as follows,

$$\boldsymbol{U}(t, \mathrm{pre}) = \tau_{\rm decay}\boldsymbol{U}(t-1) + \mathbf{W}\boldsymbol{X}(t), \tag{4}$$

where $\boldsymbol{U}(t, \mathrm{pre})$ is the pre-membrane potential at t-th timestep, $\boldsymbol{U}(t)$ is the membrane potential at t-th timestep, $\tau_{\rm decay}$ is the membrane time constant to describe the membrane potential decaying which is set as 0.25 in the paper, $\mathbf{W}$ is the weight, and $\boldsymbol{X}(t)$ is the binary map comes from the previous layer at t-th timestep. Since the $\boldsymbol{X}(t)$ is a binary tensor, $\mathbf{W}\boldsymbol{X}(t)$ can be realized by additions, which is more energy-efficient than multiplications.

Next, if $\boldsymbol{U}(t)$ exceeds a threshold, the LIF neuron will fire a spike and then hard-reset to 0, given by

$$\boldsymbol{O}(t) = \begin{cases} 1, & \text{if } \boldsymbol{U}(t, \mathrm{pre}) \geq V_{\rm th} \\ 0, & \text{otherwise} \end{cases}, \tag{5}$$

$$\boldsymbol{U}(t) = \boldsymbol{U}(t, \mathrm{pre}) \cdot (1 - \boldsymbol{O}(t)), \tag{6}$$

where $V_{\rm th}$ is a given firing threshold and $\boldsymbol{O}(t)$ is the output of the LIF neuron.

Let set $\boldsymbol{O}(t, \mathrm{last})$ as the binary output of the last firing activation layer. It will be presented to the average-pooling layer and then full-connected layer, given by

$$\boldsymbol{y}(t) = \mathbf{W}_f\texttt{Avgpool}(\boldsymbol{O}(t, \mathrm{last})), \tag{7}$$

where $\boldsymbol{y}(t)$ is the output of the network at t-th timestep and $\mathbf{W}_f$ is the weight matrix of the full-connected layer. obviously, the output feature, $\texttt{Avgpool}(\boldsymbol{O}(t, \mathrm{last}))$ will be a discrete vector with a limited number of values. Hence, the representation ability of the output feature will be constrained compared to a full precision vector. We advocate for using the ReLU activation layer to replace the last LIF activation layer. Then the output of the last activation layer will be

$$\boldsymbol{O}(t, \mathrm{last}) = \texttt{ReLU}(\mathbf{W}\boldsymbol{X}(t)). \tag{8}$$

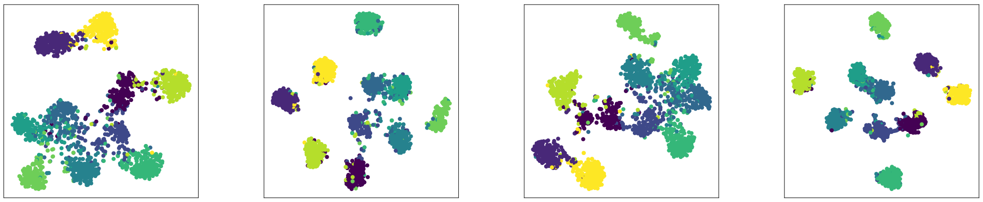
Thus we will obtain a full precision output feature representation with a stronger expression ability. With the rich and powerful output feature, the SNN can converge to a highly accurate model easily but with a trivial cost.

3.3 Training Framework

We train the SNN with our method using the spatial-temporal backpropagation (STBP) algorithm Wu et al. (2019b), which treats the spiking neuron as a self-recurrent neural network thus enabling an

Algorithm 1 Training SNN for one epoch.

```
Input: An SNN to be trained whose last LIF activation layer is replaced by the ReLU activation layer ; a pre-trained ANN; timestep: $T$; training dataset; total training iteration in one epoch $I_{\text{train}}$.
Output: The trained SNN.
for all $i = 1, 2, \ldots, I_{\text{train}}$ iteration do
    Get mini-batch training data and class label: $\boldsymbol{Y}^i$;
    Feed the mini-batch training data into the SNN and the ANN networks;
    Calculate the SNN output $\boldsymbol{O}_s^i(t)$ and the output feature $\boldsymbol{h}_s^i(t)$ of each time step;
    Compute classification loss $L_{\text{CE}} = \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_{CE}(\boldsymbol{O}_s^i(t), \boldsymbol{Y}^i)$;
    Calculate the ANN output $\boldsymbol{P}_a^i$;
    Present the $\boldsymbol{h}_s^i(t)$ to the ANN's classifier to product the output $\boldsymbol{P}_s^i(t)$ of each time step;
    Compute alignment loss $L_{AF} = \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_{\text{AF}}(\boldsymbol{P}_s^i(t), \boldsymbol{P}_a^i)$;
    Compute the total loss $\mathcal{L}_{\text{Total}}$ by Eq. 3;
    Backpropagation using STBP algorithm Wu et al. (2019b) and the SG method (see in Eq. 9),
    then update the SNN model parameters.
end for
```



(a) Vanilla (b) $\mathcal{L}_{\text{AF}}$ (c) RepAct (d) $\mathcal{L}_{\text{AF}}$ + RepAct

Figure 2: t-SNE visualization of the output feature for random 2000 samples in CIFAR-10. Every color represents a different class. It can be clearly seen that our method can learn better representations.

error backpropagation mechanism for SNNs that follows the same principles as in conventional deep networks. As for the non-differentiable firing activity of the spiking neuron, we choose the STE surrogate gradients to solve it as doing in other surrogate gradient (SG) methods Rathi & Roy (2020); Guo et al. (2022b). Mathematically, it is defined as:

$$\frac{d\boldsymbol{O}}{d\boldsymbol{U}} = \begin{cases} 1, & \text{if } 0 \leq \boldsymbol{U} \leq 1 \\ 0, & \text{otherwise} \end{cases}. \tag{9}$$

Then, the SNN model can be trained end-to-end. As described in Zheng et al. (2020); Guo et al. (2022b, 2023b,d), with the SNN going deep, the distribution of membrane potential will shift accumulatively and fall into an inappropriate range, which will cause accuracy to decrease. To deal with this problem, we adopt the threshold-dependent Batch Normalization(tdBN) technique Zheng et al. (2020). The training algorithm of our method for one epoch is detailed in the Algo. 1.

4 Ablation Study

To understand how our method works in practice, a series of ablative studies for $\mathcal{L}_{\text{AF}}$ and RepAct with spiking ResNet20 architecture along with different timesteps were conducted on the CIFAR-10 dataset. In the experiment, we find that when $\lambda = 0.1$, $\mathcal{L}_{\text{AF}}$ can lead to a relatively better result and we adopt this setting. The top-1 accuracy of these models with $\mathcal{L}_{\text{AF}}$ and RepAct alone and their combinations are shown in Tab. 1. It's can be seen that both $\mathcal{L}_{\text{AF}}$ and RepAct can greatly improve the models' performances alone. furthermore, the performances can be still improved by combining these two methods. Take the first ablation experiment (*i.e.* timestep = 1) for example, the standard training of SNN is 90.40%. If we choose to use $\mathcal{L}_{\text{AF}}$ loss and RepAct alone, the performance would boost to 92.28% and 92.50%, which are huge improvements (more than 2.0%). Using the joint $\mathcal{L}_{\text{AF}}$

Table 1: Ablation experiments for RepAct and $\mathcal{L}_{AF}$.

Timestep	Methods	Accuracy
1	baseline	90.40%
	w/ $\mathcal{L}_{\mathrm{AF}}$	92.28%
	w/ RepAct	92.50%
	w/ $\mathcal{L}_{\mathrm{AF}}$ & RepAct	**92.66%**
2	baseline	92.80%
	w/ $\mathcal{L}_{\mathrm{AF}}$	93.53%
	w/ RepAct	93.62%
	w/ $\mathcal{L}_{\mathrm{AF}}$ & RepAct	**93.86%**
4	baseline	93.85%
	w/ $\mathcal{L}_{\mathrm{AF}}$	94.39%
	w/ RepAct	94.66%
	w/ $\mathcal{L}_{\mathrm{AF}}$ & RepAct	**94.74%**

and RepAct loss, we get a 2.26% accuracy improvement. Moreover, we also show the output feature representation ability of these SNN models with 1 timestep using t-SNE method van der Maaten & Hinton (2008) in Fig. 2. It can be observed that $\mathcal{L}_{\mathrm{AF}}$ and RepAct are both able to learn more discriminative features, which also correlates with quantitative accuracy gains.

5 Experiments

In this section, we conducted extensive experiments using widely-used spiking ResNet20 Rathi & Roy (2020); Sengupta et al. (2018), VGG16 Rathi & Roy (2020), ResNet18 Fang et al. (2021a), ResNet19 Zheng et al. (2020), and ResNet34 Fang et al. (2021a) to verify the effectiveness of the proposed methods on both static and neuromorphic datasets including CIFAR-10 Krizhevsky et al., CIFAR-100 Krizhevsky et al., ImageNet Deng et al. (2009), and CIFAR10-DVS Li (2017). The firing threshold V_{th} and the membrane potential decaying τ_{decay} were set as 0.5 and 0.25 respectively. For static image datasets, we fed the images into the SNN model directly and used the first layer to encode the images to spikes, as in recent works Zheng et al. (2020); Rathi & Roy (2020). For the neuromorphic image dataset, we used the $0/1$ spike format directly. We report the top-1 accuracy results with the mean accuracy and standard deviation of 3 trials.

CIFAR-10. The CIFAR-10 dataset consists of 60K 32×32 images in 10 classes with 50K training images and 10K test images respectively. The data normalization, random horizontal flipping, cropping, AutoAugment Cubuk et al. (2019), and Cutout DeVries & Taylor (2017) were used for data augmentation as in Li et al. (2021b); Guo et al. (2022b). We utilized the SGD optimizer to train our models for 400 epochs with the 0.9 momentum and a learning rate of 0.1 cosine decayed to 0. We found that $\lambda = 0.1$ can lead to a relatively better result on the CIFAR dataset and we adopt this setting in the paper. From Tab. 3, it can be seen that our models achieve better performances than other SoTA methods for three commonly used networks in this field and can greatly reduce the timesteps, corresponding to inference time. Our VGG16 model with only 2 timesteps even outperforms the Diet-SNN Rathi & Roy (2020) with 10 timesteps by 1.31% accuracy. With only 2 timesteps, our method based on Resnet19 can also outperform the RecDis-SNN Zhang & Li (2020), the TET Deng et al. (2022), and the STBP-tdBN Zheng et al. (2020) with 6 timesteps by 0.64%, 1.69%, and 3.03% accuracy, respectively. The same superiority is also shown with the ResNet20 backbone. These comparison results clearly show the efficiency and effectiveness of our method.

CIFAR-100. We also verified our method on CIFAR-100. The CIFAR-100 dataset is similiar with the CIFAR-100 but in 100 classes. With the same training pipeline and the architectures on CIFAR-10 here, we found that our method can also achieves better accuracy than other prior works with fewer timesteps. Especially, the ResNet19 trained with our method can achieve 82.43% top-1 accuracy with timesteps of only 2, which outperforms the current methods, TET Deng et al. (2022) and TEBN Duan et al. (2022) by 7.71% and 6.02% but with 6 timesteps.

ImageNet. The ImageNet dataset consists of more than 1,250K training and 50K test images in 1000 classes with 224×224 respectively. We used standard data normalization, random horizontal flipping, and cropping for data augmentation. The SGD optimizer with the 0.9 momentum and a learning

Table 2: Comparison with SoTA methods on CIFAR-10.

Dataset	Method	Type	Architecture	Timestep	Accuracy
CIFAR-10	SpikeNorm Sengupta et al. (2018)	ANN2SNN	VGG16	2500	91.55%
	Hybrid-Train Rathi et al. (2020)	Hybrid training	VGG16	200	92.02%
	Spike-basedBP Lee et al. (2020)	SNN training	ResNet11	100	90.95%
	STBP Wu et al. (2019b)	SNN training	CIFARNet	12	90.53%
	TSSL-BP Zhang & Li (2020)	SNN training	CIFARNet	5	91.41%
	PLIF Fang et al. (2021b)	SNN training	PLIFNet	8	93.50%
	GLIF Yao et al. (2022)	SNN training	ResNet19	2	94.44%
	DSR Meng et al. (2023)	SNN training	ResNet18	20	95.40%
	KDSNN Xu et al. (2023b)	SNN training	ResNet18	4	93.41%
	TAB Jiang et al. (2024)	SNN training	ResNet19	4	94.76%
	Diet-SNN Rathi & Roy (2020)	SNN training	VGG16	5	92.70%
				10	93.44%
			ResNet20	5	91.78%
				10	92.54%
	Dspike Li et al. (2021b)	SNN training	ResNet20	2	93.13%
				4	93.66%
				6	94.25%
	STBP-tdBN Zheng et al. (2020)	SNN training	ResNet19	2	92.34%
				4	92.92%
				6	93.16%
	TET Deng et al. (2022)	SNN training	ResNet19	2	94.16%
				4	94.44%
				6	94.50%
	RecDis-SNN Guo et al. (2022b)	SNN training	ResNet19	2	93.64%
				4	95.53%
				6	95.55%
	Our method	SNN training	ResNet19	1	**95.37%**±0.08
				2	**96.19%**±0.10
			ResNet20	1	**92.66%**±0.08
				2	**93.86%**±0.07
				4	**94.74%**±0.08
			VGG16	2	**94.07%**±0.10
				4	**94.75%**±0.09

Table 3: Comparison with SoTA methods on CIFAR-100.

Dataset	Method	Type	Architecture	Timestep	Accuracy
CIFAR-100	RMP Han et al. (2020)	ANN2SNN	ResNet20	2048	67.82%
	BinarySNN Lu & Sengupta (2020)	ANN2SNN	VGG15	62	63.20%
	Hybrid-Train Rathi et al. (2020)	Hybrid training	VGG11	125	67.90%
	T2FSNN Park et al. (2020)	ANN2SNN	VGG16	680	68.80%
	TAB Jiang et al. (2024)	SNN training	ResNet19	4	76.81%
	Diet-SNN Rathi & Roy (2020)	SNN training	ResNet20	5	64.07%
			VGG16	5	69.67%
	Dspike Li et al. (2021b)	SNN training	ResNet20	2	71.68%
				4	73.35%
				6	74.24%
	TET Deng et al. (2022)	SNN training	ResNet19	2	72.87%
				4	74.47%
				6	74.72%
	RecDis-SNN Guo et al. (2022b)	SNN training	ResNet19	4	74.10%
			VGG16	5	69.88%
	Real Spike Guo et al. (2022c)	SNN training	ResNet20	5	66.60%
			VGG16	5	70.62%
	TEBN Duan et al. (2022)	SNN training	ResNet19	2	75.86%
				4	76.13%
				6	76.41%
	Our method	SNN training	VGG16	5	**72.53%**±0.11
			ResNet19	1	**77.08%**±0.08
				2	**82.43%**±0.09
			ResNet20	2	**71.55%**±0.12
				4	**73.01%**±0.10

rate of 0.1 cosine decayed to 0 is adopted. Except that we set $\lambda = 1$ and epochs as 320, the other training settings are consistent with CIFAR dataset. Our results are presented in Tab. 4. Again, it can

Table 4: Comparison with SoTA methods on ImageNet.

Dataset	Method	Type	Architecture	Timestep	Accuracy
ImageNet	STBP-tdBN Zheng et al. (2020)	SNN training	ResNet34	6	63.72%
	TET Deng et al. (2022)	SNN training	ResNet34	6	64.79%
	MS-ResNet Hu et al. (2021)	SNN training	ResNet18	6	63.10%
	OTTT Xiao et al. (2022)	SNN training	ResNet34	6	63.10%
	RecDis-SNN Guo et al. (2022b)	SNN training	ResNet34	6	67.33%
	SlipReLU Jiang et al. (2023)	ANN-SNN	ResNet34	32	66.61%
	Real Spike Guo et al. (2022c)	SNN training	ResNet18	4	63.68%
			ResNet34	4	67.69%
	SEW ResNet Fang et al. (2021a)	SNN training	ResNet18	4	63.18%
			ResNet34	4	67.04%
	OTTT Xiao et al. (2022)	SNN training	ResNet34	6	65.15%
	GLIF Yao et al. (2022)	SNN training	ResNet34	4	67.52%
	DSR Meng et al. (2023)	SNN training	ResNet18	50	67.74%
	Our method	SNN training	ResNet18	4	**65.31%**±0.09
			ResNet34	4	**67.40%**±0.10

Table 5: Comparison with SoTA methods on CIFAR10-DVS.

Dataset	Method	Type	Architecture	Timestep	Accuracy
CIFAR10-DVS	Rollout Kugele et al. (2020)	Rollout	DenseNet	10	66.80%
	LIAF-Net Wu et al. (2022)	Conv3D	LIAF-Net	10	71.70%
	LIAF-Net Wu et al. (2022)	LIAF	LIAF-Net	10	70.40%
	STBP-tdBN Zheng et al. (2020)	SNN training	ResNet19	10	67.80%
	RecDis-SNN Guo et al. (2022b)	SNN training	ResNet19	10	72.42%
	Real Spike Guo et al. (2022c)	SNN training	ResNet19	10	72.85%
			ResNet20	10	78.00%
	Ternary Spike Guo et al. (2024a)	SNN training	ResNet19	10	78.40%
			ResNet20	10	78.70%
	Our method	SNN training	ResNet19	10	**80.10%**±0.20
			ResNet20	10	**80.50%**±0.20

be observed that our method also reaches the SoTA. Our ResNet18 and ResNet34 achieve 65.31% and 67.40% top-1 accuracy with only 4 timesteps, which is just a little worse than GLIF Yao et al. (2022). However, GLIF adopts the complex spiking neuron, which will increase the inference cost. It's worth noting that our models even outperform better than these SEW ResNet-based models Fang et al. (2021a); Guo et al. (2022c), which transmits information with arbitrary integers. This shows the ability of our method to handle these large-scale datasets.

CIFAR10-DVS. The neuromorphic dataset, CIFAR10-DVS, was also used in the paper. It is the neuromorphic version of a part of the CIFAR-10 dataset which consists of 10K images in 10 classes. We also adopted the principle to split the dataset into 9K training images and 1K test images, and resize them to 48×48 to evaluate the models' performance similar to Wu et al. (2019b); Guo et al. (2022c,b). the random horizontal flip and random roll within 5 pixels are also adopted in the paper for augmentation Guo et al. (2022b). To train the ANN model on the neuromorphic dataset, we change the channel number of the first layer of the ANN to 20 to contain all the temporal input at once. Other training settings are the same as that for CIFAR-10 dataset. Our method reaches to 80.10% and 80.50% for ResNet19 and ResNet20.

6 Conclusion

This work aims at enhancing the output feature representation of SNNs. Then, the $\mathcal{L}_{\text{AF}}$ Loss that improves the learning of the output feature representation of the SNN using the well-trained ANN classifier and the RepAct method which replaces the last LIF activation layer with the ReLU activation layer to generate a more powerful output feature are proposed. A series of ablation studies show that the two proposed methods can greatly increase the SNN's accuracy. The proposed methods can also be combined and will consistently outperform the other state-of-the-art methods.

Acknowledgment

This work is supported by grants from the National Natural Science Foundation of China under contracts No.12202412 and No.12202413.

References

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G.-J., Taba, B., Beakes, M., Brezzo, B., Kuang, J., Manohar, R., Risk, W., Jackson, B., and Modha, D. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 34:1537–1557, 10 2015. doi: 10.1109/TCAD.2015.2474396.

Baltes, M., Abujahar, N., Yue, Y., Smith, C. D., and Liu, J. Joint ann-snn co-training for object localization and image segmentation, 2023.

Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H., and Inkpen, D. Enhanced lstm for natural language inference. 2016.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. 2020.

Cubuk, E., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. Autoaugment: Learning augmentation strategies from data. pp. 113–123, 06 2019. doi: 10.1109/CVPR.2019.00020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Li, F.-F. Imagenet: a large-scale hierarchical image database. pp. 248–255, 06 2009. doi: 10.1109/CVPR.2009.5206848.

Deng, S., Li, Y., Zhang, S., and Gu, S. Temporal efficient training of spiking neural network via gradient re-weighting. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=_XNtisL32jv`.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout, 2017.

Ding, J., Yu, Z., Tian, Y., and Huang, T. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. 2021.

Duan, C., Ding, J., Chen, S., Yu, Z., and Huang, T. Temporal effective batch normalization in spiking neural networks. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=fLIgyyQiJqz`.

Fang, W., Yu, Z., Chen, Y., Huang, T., and Tian, Y. Deep residual learning in spiking neural networks. 2021a.

Fang, W., yu, Z., Chen, Y., Masquelier, T., Huang, T., and Tian, Y. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. 08 2021b.

Girshick, R. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 04 2015. doi: 10.1109/ICCV.2015.169.

Guo, Y., Chen, Y., Zhang, L., Liu, X., Wang, Y., Huang, X., and Ma, Z. Im-loss: information maximization loss for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:156–166, 2022a.

Guo, Y., Tong, X., Chen, Y., Zhang, L., Liu, X., Ma, Z., and Huang, X. Recdis-snn: Rectifying membrane potential distribution for directly training spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 326–335, June 2022b.

Guo, Y., Zhang, L., Chen, Y., Tong, X., Liu, X., Wang, Y., Huang, X., and Ma, Z. Real spike: Learning real-valued spikes for spiking neural networks. In *European Conference on Computer Vision*, pp. 52–68. Springer, 2022c.

Guo, Y., Huang, X., and Ma, Z. Direct learning-based deep spiking neural networks: a review. *Frontiers in Neuroscience*, 17:1209795, 2023a.

Guo, Y., Liu, X., Chen, Y., Zhang, L., Peng, W., Zhang, Y., Huang, X., and Ma, Z. Rmp-loss: Regularizing membrane potential distribution for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17391–17401, 2023b.

Guo, Y., Peng, W., Chen, Y., Zhang, L., Liu, X., Huang, X., and Ma, Z. Joint a-snn: Joint training of artificial and spiking neural networks via self-distillation and weight factorization. *Pattern Recognition*, 142:109639, 2023c.

Guo, Y., Zhang, Y., Chen, Y., Peng, W., Liu, X., Zhang, L., Huang, X., and Ma, Z. Membrane potential batch normalization for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19420–19430, 2023d.

Guo, Y., Chen, Y., Liu, X., Peng, W., Zhang, Y., Huang, X., and Ma, Z. Ternary spike: Learning ternary spikes for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12244–12252, 2024a.

Guo, Y., Chen, Y., and Ma, Z. Neuroclip: Neuromorphic data understanding by clip and snn. *IEEE Signal Processing Letters*, 2024b.

Han, B., Srinivasan, G., and Roy, K. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. pp. 13555–13564, 06 2020. doi: 10.1109/CVPR42600.2020.01357.

Hao, Z., Bu, T., Ding, J., Huang, T., and Yu, Z. Reducing ann-snn conversion error through residual membrane potential, 2023a.

Hao, Z., Ding, J., Bu, T., Huang, T., and Yu, Z. Bridging the gap between anns and snns by calibrating offset spikes, 2023b.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. pp. 770–778, 06 2016. doi: 10.1109/CVPR.2016.90.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. 2019.

Hu, Y., Wu, Y., Deng, L., and Li, G. Advancing residual learning towards powerful deep spiking neural networks. *arXiv preprint arXiv:2112.08954*, 2021.

Jeffares, A., Guo, Q., Stenetorp, P., and Moraitis, T. Spike-inspired rank coding for fast and accurate recurrent neural networks. *CoRR*, abs/2110.02865, 2021. URL `https://arxiv.org/abs/2110.02865`.

Jiang, H., Anumasa, S., De Masi, G., Xiong, H., and Gu, B. A unified optimization framework of ann-snn conversion: Towards optimal mapping from activation values to firing rates. In *International Conference on Machine Learning*, pp. 14945–14974. PMLR, 2023.

Jiang, H., Zoonekynd, V., Masi, G. D., Gu, B., and Xiong, H. TAB: Temporal accumulated batch normalization in spiking neural networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=k1wlmtPGLq`.

Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., and Kalantidis, Y. Decoupling representation and classifier for long-tailed recognition. 2019.

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

Kugele, A., Pfeil, T., Pfeiffer, M., and Chicca, E. Efficient processing of spatio-temporal data streams with spiking neural networks. *Frontiers in Neuroscience*, 14:439, 05 2020. doi: 10.3389/fnins.2020.00439.

Kushawaha, R. K., Kumar, S., Banerjee, B., and Velmurugan, R. Distilling spikes: Knowledge distillation in spiking neural networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 4536–4543. IEEE, 2021.

Lee, C., Sarwar, S., Panda, P., Srinivasan, G., and Roy, K. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 14:119, 02 2020. doi: 10.3389/fnins.2020.00119.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373, 2015.

Li, H. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, 11, 05 2017. doi: 10.3389/fnins.2017.00309.

Li, Y. and Zeng, Y. Efficient and accurate conversion of spiking neural network with burst spikes, 2022.

Li, Y., Deng, S., Dong, X., Gong, R., and Gu, S. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 6316–6325, 06 2021a.

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

Lu, S. and Sengupta, A. Exploring the connection between binary and spiking neural networks. *Frontiers in Neuroscience*, 14:535, 06 2020. doi: 10.3389/fnins.2020.00535.

Meng, Q., Xiao, M., Yan, S., Wang, Y., Lin, Z., and Luo, Z.-Q. Training high-performance low-latency spiking neural networks by differentiation on spike representation, 2023.

Ming, Q., Miao, L., Ma, Z., Zhao, L., Zhou, Z., Huang, X., Chen, Y., and Guo, Y. Deep dive into gradients: Better optimization for 3d object detection with gradient-corrected iou supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5136–5145, 2023.

Park, S., Kim, S., Na, B., and Yoon, S. T2fsnn: Deep spiking neural networks with time-to-first-spike coding. 2020.

Rathi, N. and Roy, K. DIET-SNN: direct input encoding with leakage and threshold optimization in deep spiking neural networks. *CoRR*, abs/2008.03658, 2020. URL `https://arxiv.org/abs/2008.03658`.

Rathi, N., Srinivasan, G., Panda, P., and Roy, K. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. 05 2020.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. pp. 1–10, 01 2016.

Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroscience*, 13, 02 2018. doi: 10.3389/fnins.2019.00095.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

Takuya, S., Zhang, R., and Nakashima, Y. 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.

Tian, Y., Krishnan, D., and Isola, P. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020.

van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL `http://jmlr.org/papers/v9/vandermaaten08a.html`.

Wang, S., Cheng, T. H., and Lim, M.-H. LTMD: Learning improvement of spiking neural networks with learnable thresholding neurons and moderate dropout. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=BbaSRgUHW3`.

Wang, S., Schmutz, V., Bellec, G., and Gerstner, W. Mesoscopic modeling of hidden spiking neurons, 2023.

Wu, J., Pan, Z., Zhang, M., Das, R. K., Chua, Y., and Li, H. Robust sound recognition: A neuromorphic approach. In *Interspeech*, pp. 3667–3668, 2019a.

Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., and Shi, L. Direct training for spiking neural networks: Faster, larger, better. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:1311–1318, 07 2019b. doi: 10.1609/aaai.v33i01.33011311.

Wu, Z., Zhang, H., Lin, Y., Li, G., Wang, M., and Tang, Y. Liaf-net: Leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6249–6262, 2022. doi: 10.1109/TNNLS.2021.3073016.

Xiao, M., Meng, Q., Zhang, Z., He, D., and Lin, Z. Online training through time for spiking neural networks. *arXiv preprint arXiv:2210.04195*, 2022.

Xu, Q., Li, Y., Fang, X., Shen, J., Liu, J. K., Tang, H., and Pan, G. Biologically inspired structure learning with reverse knowledge distillation for spiking neural networks. *arXiv preprint arXiv:2304.09500*, 2023a.

Xu, Q., Li, Y., Shen, J., Liu, J. K., Tang, H., and Pan, G. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. *arXiv preprint arXiv:2304.05627*, 2023b.

Yao, X., Li, F., Mo, Z., and Cheng, J. Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks. *arXiv preprint arXiv:2210.13768*, 2022.

Zhang, M., Wu, J., Belatreche, A., Pan, Z., Xie, X., Chua, Y., Li, G., Qu, H., and Li, H. Supervised learning in spiking neural networks with synaptic delay-weight plasticity. *Neurocomputing*, 409: 103–118, 2020.

Zhang, W. and Li, P. Temporal spike sequence learning via backpropagation for deep spiking neural networks. 02 2020.

Zhang, Y., Liu, X., Chen, Y., Peng, W., Guo, Y., Huang, X., and Ma, Z. Enhancing representation of spiking neural networks via similarity-sensitive contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16926–16934, 2024.

Zheng, H., Wu, Y., Deng, L., Hu, Y., and Li, G. Going deeper with directly-trained larger spiking neural networks. 10 2020.

Zhou, Z., Zhu, Y., He, C., Wang, Y., YAN, S., Tian, Y., and Yuan, L. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=frE4fUwz_h`.

Zhu, Y., Yu, Z., Fang, W., Xie, X., Huang, T., and Masquelier, T. Training spiking neural networks with event-driven backpropagation. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 2022.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes] .

   Justification: We clearly state the claims made and the contributions made in both the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [NA] .

   Justification: We find no limitation which we feel must be specifically highlighted here.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes] .

Justification: We provide the full set of assumptions and complete proofs in the Section 3.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes] .

   Justification: We provide the detail experiment settings in the Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes] .

Justification: We provide open access to the data and code with sufficient instructions in the supplemental material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes] .

Justification: All implementations are described in the experiments section.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes] .

Justification: We report the mean as well as the standard deviation accuracy in experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes] .

   Justification: The computation resources description is provided in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes] .

   Justification: The research conducted with the NeurIPS Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No] .

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA] .

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes] .

    Justification: The original paper for datasets we used are all cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA] .

    Justification: We adopt public datasets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA] .

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA] .

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.