ADAPTIVE SMOOTHING GRADIENT LEARNING FOR SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs) with biologically inspired spatio-temporal dynamics show higher energy efficiency on neuromorphic architectures. Error backpropagation in SNNs is prohibited by the all-or-none nature of spikes. The existing solution circumvents this problem by a relaxation on the gradient calculation using a continuous function with a constant relaxation degree, so-called surrogate gradient learning. Nevertheless, such solution introduces additional smoothness error on spiking firing which leads to the gradients being estimated inaccurately. Thus, how to adjust adaptively the relaxation degree and eliminate smoothness error progressively is crucial. Here, we propose a methodology such that training a prototype neural network will evolve into training an SNN gradually by fusing the learnable relaxation degree into the network with random spike noise. In this way, the network learns adaptively the accurate gradients of loss landscape in SNNs. The theoretical analysis further shows optimization on such a noisy network could be evolved into optimization on the embedded SNN with shared weights progressively. Moreover, we conduct extensive experiments on static images, dynamic event streams, speech, and instrumental sounds. The results show the proposed method achieves state-of-the-art performance across all the datasets with remarkable robustness on different relaxation degrees.

1 INTRODUCTION

Spiking Neural Networks (SNNs), composed of biologically plausible spiking neurons, present high potential for fast inference and low power consumption on neuromorphic architectures (Akopyan et al., 2015; Davies et al., 2018; Pei et al., 2019). Instead of the expensive multiply-accumulation (MAC) operations presented in ANNs, SNNs operate with binary spikes asynchronously and offer sparse accumulation (AC) operations with lower energy costs. Additionally, existing research has revealed SNNs promise to realize machine intelligence especially on sparse spatio-temporal patterns (Roy et al., 2019). Nevertheless, such bio-mimicry with the all-or-none firing characteristics of spikes brings inevitably difficulties to supervised learning in SNNs.

Error backpropagation is the most promising methodology to develop deep neural networks. However, the nondifferentiable spike firing prohibits the direct application of backpropagation on SNNs. To address this challenge, two families of gradient-based training methods are developed: (1) surrogate gradient learning (Shrestha & Orchard, 2018; Wu et al., 2018; Neftci et al., 2019) and (2) Time-based learning (Mostafa, 2017; Zhang & Li, 2020). For surrogate gradient learning, it adopts a smooth curve to estimate the ill-defined derivative of the Heaviside function in SNNs. The backpropagation, in this way, could be tractable at both spatial and temporal domains in an iterative manner. Meanwhile, surrogate gradient learning could substantially benefit from the complete ecology of deep learning. It has been widely used to solve complex pattern recognization tasks (Zenke & Vogels, 2021; Neftci et al., 2019). However, the smooth curve distributes the gradient of a single spike into a group of analog items in temporal neighbors (Zhang & Li, 2020), which is mismatched with the inherent dynamics of spiking neurons. So we identify the problem as *gradient mismatching* in this paper. As a result, most parameters are updated in a biased manner in surrogate gradient learning, which limits the performance of SNNs. Besides, different smoothness of surrogate functions may greatly affect the network performance (Hagenaars et al., 2021; Li et al., 2021c).

The time-based method is the other appealing approach. By estimating the gradients on the exact spike times, the time-based method circumvents the gradient mismatching issues in surrogate gradient learning naturally. However, to obtain the exact expression of spike time, most works (Mostafa, 2017; Zhang & Li, 2020) suppose the firing count of each neuron remains unchanged during training (Yang et al., 2021) which is difficult to establish in practice. Besides, it is difficult to adapt the customized backward flows in the time-based method with auto-differential frameworks such as PyTorch, MXNet, and TensorFlow. Moreover, some special tricks are necessary to avoid the phenomenon of dead neurons (Bohte et al., 2002). Therefore, it is not flexible to obtain deep SNNs with the time-based method.

To solve the problems, this paper proposes adaptive smoothing gradient learning (ASGL) to train SNNs directly. In general, we inject spikes as noise in ANN training and force the error surfaces of ANNs into that of SNNs. With the design of dual-mode forwarding, the smoothness factor could be incorporated into training without the need for a specific design of hyperparameter search, which could be computationally expensive. Therefore most parameters could be updated against mismatched gradients adaptively. In addition, compared to the time-based method, ASGL backpropagates errors in both spatial and temporal domains without special constraints and restart mechanism.

We analyze the evolution of the noisy network with dual mode from the perspective of iterative optimization. As a result, the optimization of the noisy network could be converted into minimizing the loss of the embedded SNN with the penalty of smooth factors. Experiments show the proposed method achieves state-of-the-art performance on static images, dynamic visual streams, speech, and instrumental sounds. It is worth noting that the method shows extraordinary robustness for different hyperparameter selections of smooth factors. Finally, we investigate the evolution of such a hybrid network by visualizing activation similarities, network perturbation, and updated width.

2 Related Works

Direct Training. To circumvent the difficulties from non-differential spikes, surrogate gradient learning approximates spike activities with a pre-defined curve (Wu et al., 2019; Shrestha & Orchard, 2018; Gu et al., 2019; Zenke & Vogels, 2021; Fang et al., 2021b). Wu et al. (2018) proposed to backpropagate errors in both spatial and temporal domains to train SNNs directly with surrogate gradient. Similarly, Shrestha & Orchard (2018) solved the temporal credit assignment problem in SNNs with the smoothness of a custom probability density function. To suppress gradient vanishing or explosion in deep SNNs, Zheng et al. (2021) further proposed threshold-dependent batch normalization (tdBN) and elaborated shortcut connection in standard ResNet architectures.

Gradient Mismatching. The mismatching problem in surrogate gradient learning has attracted considerable attention. Li et al. (2021c) optimized the shape of the surrogate gradient function with the finite difference method to compensate for this problem. Nevertheless, their method needs to initialize an update step for finite difference empirically. Meanwhile, it is limited by the high computational complexity of the finite difference method. Therefore, only information from the proceeding layers is adopted in the update of surrogate functions. There are still other works bypassing the difficulty without using surrogate gradient. Zhang & Li (2020) handled error backpropagation across inter-neuron and intra-neuron dependencies based on the typical time-based scheme. Furthermore, the unifying of surrogate gradient and time-based method was suggested by (Kim et al., 2020) to fuse the gradient from both spike generation and time shift. In general, those methods are constrained by specified assumptions or coupling tricks during training. Wunderlich & Pehle (2021) first proposed to compute the exact gradients in an event-driven manner and so avoid smoothing operations by solving ODEs about adjoint state variables. Nevertheless, the approach is only verified on simple datasets with shallow networks. Yang et al. (2021) developed a novel method to backpropagate errors with neighborhood aggregation and update weights in the desired direction. However, the method based on the finite difference is computationally expensive. Severa et al. (2019) propose to sharpen the bounded ReLU activation function in ANN into the Heaviside function in SNN progressively. Although yielding a similar effect with *ASGL*, it utterly depends on hand-craft sharpening schedulers with difficulties in adaptive update considering whole network evolution. Different from the previous works, *ASGL* incorporates directly learnable width factors into the end-to-end training of a noisy network.

3 PRELIMINARY

Notation. We follow the conventions representing vectors and matrix with bold italic letters and bold capital letters respectively, such as $\boldsymbol{s}$ and $\boldsymbol{W}$. For matrix derivatives, we use a consistent numerator layout across the paper. For a function $f(\boldsymbol{x}) : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$, we use $D^k \boldsymbol{f}[\boldsymbol{x}]$ instead of $\frac{\partial \boldsymbol{f}^{(k)}(\boldsymbol{x})}{\partial \boldsymbol{x}}$ to represent the k-th derivative of f with respect to the variable $\boldsymbol{x}$ in the absence of ambiguity. Let $< \boldsymbol{M}_1, \boldsymbol{M}_2 >$ represent the Frobenius inner production between two matrices. For two vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, we use $\boldsymbol{u}_1 \odot \boldsymbol{u}_2$ to represent the entrywise production. Similarly, the notation of $\boldsymbol{u}^{\odot 2}$ refers to $\boldsymbol{u} \odot \boldsymbol{u}$ for simplification. We use $f \circ g$ to denote the composition of f with g.

Leaky Integrate-and-Fire (LIF) Model. To capture the explicit relation between input current $\boldsymbol{c}$ and output spikes $\boldsymbol{s}$, we adopt the iterative form of the LIF neuron model (Wu et al., 2018) in most experiments. At each time step t, the spiking neurons at l-th layer will integrate the postsynaptic current $\boldsymbol{c}^l[t]$ and update its membrane potential $\boldsymbol{u}^l[t]$:

$$\boldsymbol{u}^l[t] = \gamma \boldsymbol{u}^l[t-1] \odot \left(1 - \boldsymbol{s}^l[t-1]\right) + \boldsymbol{c}^l[t] \tag{1}$$

where $\gamma = 1 - 1/\tau_m$ is the leaky factor that acts as a constant forget gate through time. The term of $(1 - \boldsymbol{s}^l[t-1])$ indicates the membrane potential will be reset to zero when a spike is emitted at the last time step. As done in (Wu et al., 2018; Fang et al., 2021b), we use simply the dot production between weights $\boldsymbol{W}^l$ and spikes from the preceding layer $\boldsymbol{s}^{l-1}[t]$ with a shift $\boldsymbol{b}^l$ to model the synaptic function:

$$\boldsymbol{c}^l[t] = \boldsymbol{W}^l \boldsymbol{s}^{l-1}[t] + \boldsymbol{b}^l \tag{2}$$

The neurons will emit spikes $\boldsymbol{s}^l[t]$ whenever $\boldsymbol{u}^l[t]$ crosses the threshold ϑ with enough integration of postsynaptic currents:

$$\boldsymbol{s}^l[t] = \Theta\left(\hat{\boldsymbol{u}}^l[t]\right) = \Theta\left(\boldsymbol{u}^l[t] - \vartheta\right) \tag{3}$$

where $\Theta(x)$ is the Heaviside function:

$$\Theta(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

A C-LIF variant is also applied in our experiments to make a fair comparison. And we provide its iterative equations in the Appendix A.2.

Readout and Loss. For classification tasks, we need to define the readout method matching with the supervised signal $\boldsymbol{y}$. As done in the recent work (Li et al., 2021a; Rathi et al., 2020), we fetch the average postsynaptic current in the last layer $\overline{\boldsymbol{c}}^L = \frac{1}{N}\sum_t \boldsymbol{c}^L[t]$ where $N = T/\Delta t$ is the number of discrete time steps. Then the SNN prediction could be defined as the one with maximum average postsynaptic current naturally. Furthermore, we could define the cross-entropy loss removing the temporal randomness:

$$\mathcal{L}(\overline{\boldsymbol{c}}^L, \boldsymbol{y}) = -\boldsymbol{y}^T \log(\text{softmax}(\overline{\boldsymbol{c}}^L)) \tag{5}$$

For simplication, we denote $\hat{\boldsymbol{y}} = \text{softmax}(\overline{\boldsymbol{c}}^L)$ in the rest part of the paper.

4 METHOD

4.1 SPIKE-BASED BACKPROPAGATION

The nature of all-or-none firing characteristics of spikes blocks the direct utilization of backpropagation which is the key challenge to developing spike-based backpropagation. Formally, we usually need to backpropagate the credit for the state at a specified time step t^* of $\boldsymbol{\delta}^l[t] = \frac{\partial \overline{\boldsymbol{c}}^L[t^*]}{\partial \boldsymbol{W}}$ in both spatial domain and temporal domain (Detailed derivatives are provided in Appendix A.1) as follows:

$$\begin{aligned} \boldsymbol{\delta}^l[t] &= \boldsymbol{\delta}^l[t+1]\frac{\partial \boldsymbol{u}^l[t+1]}{\partial \boldsymbol{u}^l[t]} + \boldsymbol{\delta}^{l+1}[t]\frac{\partial \boldsymbol{u}^{l+1}[t]}{\partial \boldsymbol{u}^l[t]} \\ \frac{\partial \boldsymbol{u}^l[t+1]}{\partial \boldsymbol{u}^l[t]} &= \gamma \,\text{diag}\left(1 - \boldsymbol{s}^l[t] - \boldsymbol{u}^l[t] \odot \Theta'\left(\boldsymbol{u}^l[t]\right)\right) \\ \frac{\partial \boldsymbol{u}^{l+1}[t]}{\partial \boldsymbol{u}^l[t]} &= \boldsymbol{W}^{l+1} \,\text{diag}\left(\Theta'\left(\boldsymbol{u}^l[t]\right)\right) \end{aligned} \tag{6}$$

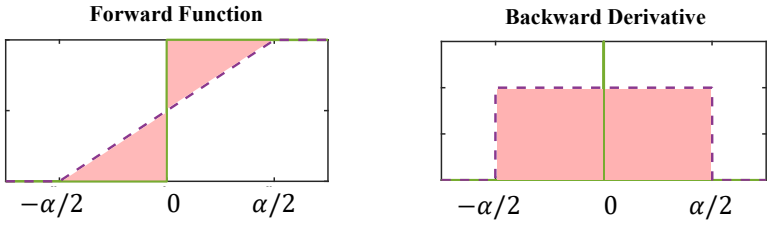

Figure 1: The forward and backward of the Heaviside function (green solid line) and the surrogate clipping function (purple dashed line). The red area represents the mismatching between the Heaviside function and surrogate function considering only one spike emission.

In the above equations, the partial derivative of the Heaviside function $\Theta(x)$ is the Dirac-Deta function which is equal to zero almost everywhere. Moreover, it goes to infinity when $x = 0$ shown with the green line in Figure 1. Such properties prohibit the backward flow in SNNs directly. Thus most researchers approximate the gradient of the Heaviside function with a predefined differentiable curve (Neftci et al., 2019; Shrestha & Orchard, 2018). Here, we use the rectangular function (Wu et al., 2018), one of the most popular approximation functions with low computational complexity, as an example to illustrate the proposed method. The idea could be applied to other alternative approximation functions. Formally, the rectangular function could be defined as:

$$\Theta^{'}(\boldsymbol{u}) \approx h_{\alpha}(\boldsymbol{u}) = \frac{1}{\alpha}\,\text{sign}\left(|\boldsymbol{u} - \vartheta| < \frac{\alpha}{2}\right) \tag{7}$$

where the width α controls the smooth degree of $h_{\alpha}(x)$ and the temperature $\kappa = 1/\alpha$ determines the relative steepness.

4.2 Design of ASGL

In surrogate learning, $\Theta^{'}(x)$ is estimated by a smooth function such as $h_{\alpha}(x)$, which predicts the change rate of loss in a relatively larger neighborhood (Li et al., 2021c). However, such estimation with constant width will deviate from the correct direction progressively during the network training. It not only brings difficulties to the network convergence but also affects the generalization ability with the disturbed underline neuron dynamics. Essentially, this problem comes from the mismatching $||\Theta^{'}(x) - h_{\alpha}(x)||$ shown in the red part of Figure 1. So how to adjust the width α and eliminate such mismatching adaptively is an important problem for surrogate gradient learning. For *ASGL*, we try to solve the problem without defining surrogate gradient. The method is simple but rather effective. Firstly, we derive the antiderivative function of surrogate function $h_{\alpha}(x)$:

$$H_{\alpha}(x) = \int_{-\infty}^{x} h_{\alpha}(u)\,du = \text{clip}(\frac{1}{\alpha}x + \frac{1}{2}, 0, 1) \tag{8}$$

Whetstone (Severa et al., 2019) uses $H_{\alpha}(x)$ for forwarding calculation and $h_{\alpha}(x)$ for backward propagation. In this way, although there is no mismatching problem, it is difficult to guarantee the network dynamics evolving into that of SNNs finally. In contrast, surrogate gradient learning uses $\Theta(x)$ and $h_{\alpha}(x)$ for forwarding calculation and backward propagation respectively. It guarantees fully spike communication but introduces the problem of gradient mismatching. Sequentially, **it makes sense to seek an approach combining advantages from both perspectives and training deep SNNs with matching gradients.**

To implement this, the basic idea of *ASGL* is just to couple the analog activation $H_{\alpha}(\boldsymbol{x})$ and binary activation $\Theta(\boldsymbol{x})$ through a random mask $\boldsymbol{m}$ during forwarding calculation:

$$\hat{H}_{\alpha}(\boldsymbol{x}) = (1 - \boldsymbol{m}) \odot H_{\alpha}(\boldsymbol{x}) + \boldsymbol{m} \odot \Phi(\Theta(\boldsymbol{x})) \tag{9}$$

where $\boldsymbol{m} \sim$ Bernoulli (p) represents independent random masking. The p controlling the proportion of analogs and spikes is referred to as the noise probability. To avoid the gradient mismatching from the surrogate function, we use function Φ to detach the gradients from spikes. Mathematically, Φ standards for the special identical mapping with the derivative $\frac{\partial \Phi(\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{0}$. In this way, the Heaviside function $\Theta(\boldsymbol{x})$ is taken as the spike noise without error backpropagation. To guarantee

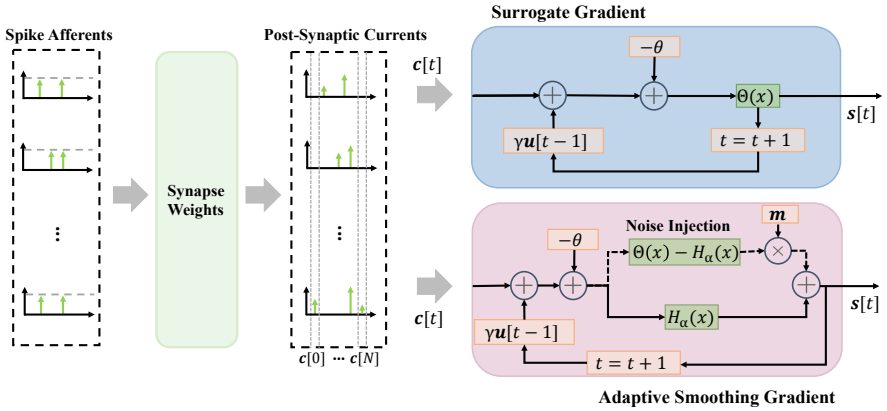

Figure 2: Comparasion of the computation flow between *ASGL* and surrogate gradient learning. The dashed line in purple area represents those operations are detached with the error backpropagation while the solid line represent the matching forward and backward flow.

the gradient still flows even when $p = 1$, we recouple both modes and only inject the difference $\Theta(\boldsymbol{x}) - H_\alpha(\boldsymbol{x})$ as noise:

$$\hat{H}_\alpha(\boldsymbol{x}) = H_\alpha(\boldsymbol{x}) + \boldsymbol{m} \odot \Phi\left(\Theta(\boldsymbol{x}) - H_\alpha(\boldsymbol{x})\right) \tag{10}$$

The operator pipeline is visualized in Figure 2. Surprisingly, the core idea of *ASGL* is so simple: replace $\Theta(\boldsymbol{u}^l[t] - \vartheta)$ with $\hat{H}_\alpha(\boldsymbol{u}^l[t] - \vartheta)$ in Equation (3) during training and still adopt $\Theta(\boldsymbol{u}^l[t] - \vartheta)$ for validation. In practice, minor alterations shown in Algorithm 1 are needed compared to the surrogate gradient learning. Notably, the hard reset given in Equation (1) transforms into a soft version at probability when the analog activations $H_\alpha(x)$ are propagated, despite the fact that the equations describing neuron dynamics remain unchanged. Assume $H_\alpha(x)$ models the function from expectational membrane potential into spike probability (rate) in a short period (corresponding to one time step). The neurons have a $(1 - H_\alpha(\boldsymbol{u}[t]))$ chance of not emitting a spike and maintaining the previous potential state. In the sense of expectation, the *soft reset* should be performed as $\boldsymbol{u}[t] = (1 - H_\alpha(\boldsymbol{u}[t])) \odot \boldsymbol{u}[t]$ rather than resetting into a fixed value.

Algorithm 1 Core function in ASGL

Require: The difference between membrane voltage and threshold $\hat{\boldsymbol{u}}^l[t] = \boldsymbol{u}^l[t] - \vartheta$;
The sign T indicates training or validation
Ensure: α is the learnable width parameter
if T **is** true **then**
  generate mask $\boldsymbol{m}$ with noise probability p
  $\boldsymbol{s}^l[t] = H_\alpha(\hat{\boldsymbol{u}}^l[t]) + \boldsymbol{m} \odot \Phi\left(\Theta(\hat{\boldsymbol{u}}^l[t]) - H_\alpha(\hat{\boldsymbol{u}}^l[t])\right)$
  {The only line needed to update compared to the surrogate learning}
else
  $\boldsymbol{s}^l[t] = \Theta(\hat{\boldsymbol{u}}^l[t])$.detach()
end if
return $\boldsymbol{s}^l[t]$

The only remaining problem is to guarantee the noisy network trained with $\hat{H}_\alpha(\boldsymbol{x})$ could be involved into an SNN with $\Theta(x)$ as activation finally. Fortunately, with the perspective of mixture feedforward, it could be achieved by simply setting the width α as learnable (see Section 4.3 for detailed analysis):

$$\frac{\partial H_\alpha(x)}{\partial \alpha} = \begin{cases} -\frac{1}{\alpha^2}x, & \text{if } -\frac{1}{2}\alpha \le x \le \frac{1}{2}\alpha \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

In this way, the gradient mismatching will gradually diminish when adaptive α approaches 0 against spike noise. Besides, it avoids tricky adjustments for width α which usually has a significant impact on performance (Wu et al., 2018; Hagenaars et al., 2021). The idea could also bring insights into

surrogate gradient learning. For example, *ASGL* will demote to surrogate gradient but still enable the adaptive width learning if we use fully spike ($p = 1$) for forwarding computation. Moreover, *ASGL* benefits naturally from the pretrained ANNs by increasing gradually p from 0. Therefore, both ANN-SNN conversion and surrogate gradient learning could be implemented in the framework of *ASGL* with different settings of noise probability p.

4.3 Theoretical Analysis

In this section, we show network dynamics of noisy networks with learnable α could be evolved into that of embedded SNNs . Suppose $\boldsymbol{F}_{\text{noise}}$ represents the noisy network used in training with $H_\alpha(x)$ across all layers while $\boldsymbol{F}_{\text{snn}}$ denotes the target SNN embedded in $\boldsymbol{F}_{\text{noise}}$ with fully spike-based calculation. The expectation over mask matrix is adopted to estimate the real loss $\boldsymbol{\ell}_{\text{noise}}(\boldsymbol{F}, \boldsymbol{s})$ of the hybrid network $\boldsymbol{F}_{\text{noise}}$:

$$\boldsymbol{\ell}_{\text{noise}}(\boldsymbol{F}, \boldsymbol{s}) \triangleq \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{\ell}(\boldsymbol{F}_{\text{noise}}(\boldsymbol{s}))] = \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s}, \hat{\boldsymbol{m}}))] \tag{12}$$

Here, $\hat{\boldsymbol{m}} = \boldsymbol{m}/p$ represents the normalized mask gathered from all layers and $\boldsymbol{s}$ denotes the input spike pattern. With the perturbation analysis in Appendix A.9, we have the following proposition:

Proposition 4.1 *Minimizing the loss of noisy network $\boldsymbol{\ell}_{noise}(\boldsymbol{F}, \boldsymbol{s})$ can be approximated into minimizing the loss of the embedded SNN $\boldsymbol{\ell}_{snn}(\boldsymbol{F}, \boldsymbol{s})$ regularized by the layerwise distance between $\Theta(\hat{\boldsymbol{u}}^l)$ and $H_\alpha(\hat{\boldsymbol{u}}^l)$.*

$$\boldsymbol{\ell}_{noise}(\boldsymbol{F}, \boldsymbol{s}) \approx \boldsymbol{\ell}_{snn}(\boldsymbol{F}, \boldsymbol{s}) + \frac{1-p}{2p} \sum_{l=1}^{L} \left\langle \boldsymbol{C}^l, diag(H_\alpha(\hat{\boldsymbol{u}}^l) - \Theta(\hat{\boldsymbol{u}}^l))^{\odot 2} \right\rangle \tag{13}$$

where $\boldsymbol{C}^l = D^2\left(\boldsymbol{\ell} \circ \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{G}^l]\right)[\boldsymbol{s}^l]$ is the second derivative of loss function $\boldsymbol{\ell}$ with respect to the l-th layer spike activation $\boldsymbol{s}^l$ in the constructed network $\boldsymbol{G}^l$, which could be treated as a constant (Nagel et al., 2020). $\boldsymbol{G}^l$ denotes the network using mixed activations after l-th layer and full spikes are adopted in the front l layers. To explain the proposition intuitively, we analyze the non-trivial case of $p \neq 1$ from the perspective of iterative alternate optimization. There are two steps: (1) fix weights $\boldsymbol{W}$, optimize width α. (2) fix width α, optimize weights $\boldsymbol{W}$. For the first case, as $\boldsymbol{\ell}_{\text{snn}}(\boldsymbol{F}, \boldsymbol{s})$ is constant with fixed weights, the width α tends to minimize the distance between $H_\alpha(\hat{\boldsymbol{u}}^l)$ and $\Theta(\hat{\boldsymbol{u}}^l)$. So the penalty term diminishes and $\boldsymbol{\ell}_{\text{noise}}(\boldsymbol{F}, \boldsymbol{s})$ approaches $\boldsymbol{\ell}_{\text{snn}}(\boldsymbol{F}, \boldsymbol{s})$ in this step. In the second step, the noisy network with global task-related loss $\boldsymbol{\ell}_{\text{noise}}(\boldsymbol{F}, \boldsymbol{s})$ is optimized under a constant smooth degree. Therefore, by applying the two steps iteratively and alternately, a high-performance SNN could be obtained through training a noisy network even if we do not increase p explicitly during training. The theoretical results have also been validated further in Fig 3b of training with fixed p. Notably, both trainable width and random noise injection with spikes are important to guarantee the first step holds on. The spike noise could be converted into the penalty on layerwise activations while the learnable α enables local optimization on it by forcing $H_\alpha(\boldsymbol{x})$ into $\Theta(\boldsymbol{x})$.

5 Experiments

To validate the effectiveness of the proposed method, we conduct extensive experiments on static images and spatio-temporal patterns such as dynamic event streams, spoken digital speech, and instrumental music. Specially, we study the evolutions of network dynamics and the effect of noise rate to explore whether and how injecting noise with spikes obtains the real observation about the loss landscape of target SNNs. More implementation details and energy estimation could be found in Appendix A.3 and Appendix A.6 respectively.

5.1 Performance on Static Images

In Table 1, we compare our work with state-of-the-art methods on the CIFAR datasets. We use the widely-used CifarNet (Wu et al., 2019) and a modified ResNet-18 structure (Li et al., 2021c). As done in (Li et al., 2021c; Deng et al., 2022), AutoAugment (Cubuk et al., 2018) and Cutout (DeVries & Taylor, 2017) are used for data augmentation. However, we do not adopt a pretrained ANN (Li et al., 2021c; Rathi et al., 2020; Rathi & Roy, 2020) to initialize weights and Time Inheritance

Table 1: Classification Performance on Static Image Benchmarks.

Dataset	Method	Type	Architecture	Time steps	Accuracy(%)
CIFAR-10	Opt. (Deng & Gu, 2021)	Conversion	CifarNet	400-600	90.61
	STBP NeuNorm (Wu et al., 2019)	Surrogate Gradient		12	90.53
	TSSL-BP (Zhang & Li, 2020)	Time-based Gradient		5	91.41
	TL (Wu et al., 2021a)	Tandem Learning		8	90.98
	PLIF-SNN (Fang et al., 2021b)	Surrogate Gradient	CifarNet-B	8	93.50
	Ours	**ASGL**	**CifarNet**	**4**	**94.74 ± 0.10**
				2	**93.80 ± 0.11**
				1	**92.84 ± 0.21**
	Hybrid (Rathi et al., 2020)	Hybrid	ResNet-20	250	92.22
	STBP-tdBN (Zheng et al., 2021)	Surrogate Gradient	ResNet-19	4	92.92
				2	92.34
	TET (Deng et al., 2022)	Surrogate Gradient	ResNet-19	4	94.44 ± 0.08
				2	94.16 ± 0.03
	SEW-ResNet (Fang et al., 2021a)*	Surrogate Gradient	SEW-ResNet-18	4	94.39 ± 0.18
				2	91.22 ± 0.14
	Dspike (Li et al., 2021c)	Surrogate Gradient	ResNet-18	4	93.66 ± 0.05
				2	93.13 ± 0.07
	Ours	**ASGL**	**ResNet-18**	**4**	**95.35** ± 0.25
				2	**95.27 ± 0.06**
				1	**94.20 ± 0.01**
CIFAR-100	Hybrid (Rathi et al., 2020)	Hybrid	VGG-11	125	67.87
	BinarySNN (Lu & Sengupta, 2020)	Conversion	VGG-15	62	63.20
	Hybrid (Rathi & Roy, 2020)	Hybrid	ResNet-20	5	64.07
	TET (Deng et al., 2022)	Surrogate Gradient	ResNet-19	4	74.47 ± 0.15
				2	72.87 ± 0.10
	Dspike (Li et al., 2021c)	Surrogate Gradient	ResNet-18	4	73.35 ± 0.14
				2	71.68 ± 0.12
	Ours	**ASGL**	**CifarNet**	**4**	**74.59 ± 0.07**
				2	**74.31** ± 0.15
				1	**72.81** ± 0.35
			ResNet-18	**4**	**74.48 ± 0.06**
				2	**73.19 ± 0.04**
				1	**70.73 ± 0.07**

* The results are reproduced through the publicly available code.

Training (TIT) (Li et al., 2021c; Deng et al., 2022) to improve performance under low time steps. Even though, *ASGL* outperforms the state-of-the-art surrogate gradient and conversion methods with the same or fewer time steps on both datasets. Remarkably, we also compare two special training methods TSSL (Zhang & Li, 2020) and TL (Wu et al., 2021a) without the definition of surrogate gradient functions. Our method also achieves a significant remarkable on the tradeoff between accuracy and latency which indicates the effectiveness of adaptive learning employed in *ASGL*.

5.2 Performance on Spatio-temporal Patterns.

Table 2: Comparasion on DVS-CIFAR10 dataset.

Method	Type	Architecture	Time steps	Accuracy
Streaming Rollout (Kugele et al., 2020)	Conversion	DenseNet	10	66.8
STBP-tdBN (Zheng et al., 2021)	Surrogate Gradient	ResNet-19	10	67.8
Conv3D (Wu et al., 2021b)	Surrogate Gradient	LIAF-Net	10	71.70
LIAF (Wu et al., 2021b)	Surrogate Gradient	LIAF-Net	10	70.40
SEW ResNet (Fang et al., 2021a)	Surrogate Gradient	Wide-7B-Net	16	74.40
PLIF-SNN (Fang et al., 2021b)	Surrogate Gradient	CifarNet-C	20	74.80
Dspike (Li et al., 2021c)	Surrogate Gradient	ResNet-18	10	75.45
TET (Deng et al., 2022)	Surrogate Gradient	VGGSNN	10	77.40
ours	**ASGL**	VGGSNN	10	**78.90**

To validate that our method handles spatio-temporal error backpropagation properly, we conduct experiments on different datasets of spatio-temporal patterns such as DVS-CIFAR10 (Li et al., 2017), and Spiking Heidelberg Dataset (SHD) (Cramer et al., 2020). More results of MedlyDB (Bittner et al., 2014) and DVS128 Gesture (Amir et al., 2017) could be found in Appendix A.4.

Performance on DVS-CIFAR10. DVS-CIFAR10 (Li et al., 2017) is a challenging benchmark neuromorphic dataset, where each sample is a record of an image of CIFAR10 scanning with repeated closed-loop motion in front of a DVS. DVS-CIFAR10 has the same number of categories (10) and samples (1k/class) as CIFAR10, but its recording process generates more noise, thus making classification more difficult. To alleviate the overfitting problem caused by data size and noise, we adopt

Table 3: Classification Performance on SHD

Method	Type	#Param.	Acc.(%)
RSNN with Adaption (Yin et al., 2020)	Surrogate Gradient	14.13w	84.4
RSNN with Data Augmentation (Cramer et al., 2020)	Surrogate Gradient	178.59w	83.2 ± 1.3
Heterogeneous RSNN (Perez-Nieves et al., 2021)	Surrogate Gradient	**10.85w**	82.7 ± 0.8
RSNN (Zenke & Vogels, 2021)	Surrogate Gradient	24.99w	82.0 ± 2.0
Fully-connected SNN (Perez-Nieves & Goodman, 2021)	Sparse Gradient	28.80w	77.5
Ours	**ASGL**	23.04w	**86.9**$\pm$ **1.0**

Table 4: Comparison on Image Classification.

Width (α)	SG	ASGL
0.5	93.19	**94.11**
1.0	93.78	**94.30**
2.5	90.68	**94.09**
5.0	62.34	**93.61**
10.0	30.85	**93.53**

Table 5: Comparison on Image Reconstruction.

Width (α)	PSNR		SSIM	
	SG	ASGL	SG	ASGL
0.1	11.91	**17.36**	0.21	**0.78**
0.5	**17.75**	**17.75**	**0.80**	**0.80**
1.0	16.55	**16.79**	0.73	**0.74**
2.5	15.27	**16.09**	0.65	**0.70**
5.0	14.66	**15.79**	0.59	**0.68**

the VGGSNN architecture and data augmentation method in (Deng et al., 2022). As shown in Table 2, our method achieves state-of-the-art performance (78.90%) without a larger network (e.g., ResNet-19, DenseNet), which outperforms existing surrogate-gradient based approaches.

Performance on Sound Datasets. The SHD dataset is a spiking dataset containing 10k spoken digits generated through an encoding model simulating auditory bushy cells in the cochlear nucleus. For training and evaluation, the dataset is split into a training set (8156 samples) and a test set (2264 samples). In this experiment, we train a three-layer SNN (800-240-20) with recurrent synaptic connections to identify the keywords in utterances (More details about recurrent connections could be found in Appendix A.5). As shown in Table 3, the proposed method achieves 2.5% accuracy improvement at least without any data augmentation introduced in (Cramer et al., 2020) compared to the latest results. Remarkably, we use standard LIF neurons shown in Equations (1) to (3) while the adaptive LIF model (Yin et al., 2020) and the heterogenous LIF model (Perez-Nieves et al., 2021) are adopted to enhance the dynamics of neurons respectively.

5.3 Ablation Study

In Table 4, we compare *ASGL* with *Surrogate Gradient* (*SG*) on CIFAR-10 with ResNet-19 architecture (Zheng et al., 2021) under $N = 3$ for the ablation study. The rectangular function is adopted with the same optimizer setting, seed, and weight initialization for a fair comparison. Specially, we train 100 epochs with SGD optimizer and the weight decay of 5e-4. The results show *ASGL* outperforms *SG* across a large range of width initialization. It is catastrophic damage for *SG* when width α is selected inappropriately ($\alpha \geq 5$). In contrast, *ASGL* exhibits surprising robustness for different width α. Furthermore, image reconstruction, a challenging regression task for SNNs, is also conducted to verify the effectiveness of *ASGL*. Here, we use $h_\alpha(x) = \frac{1}{2}\tanh(\alpha x) + \frac{1}{2}$ as a surrogate to show *ASGL* could be also applied to other functions. The fully-connected autoencoder is adopted for evaluation with the architecture of 784-128-64-32-64-128-784. Table 5 reports the peak signal-to-noise ratio (PSNR) and Structural Similarity (SSIM) of reconstructed MNIST images under 8 time steps. We could find the adaptive mechanism in *ASGL* reduces sensitivity for width in *SG* and so shows higher performance.

5.4 Effect of noise Probability

In this section, we aim to analyze how noise probability affects the performance of SNN. Firstly, we increase noise probability p from 0 to 0.8 by 0.1 with every 30 epochs during the training of ResNet-19 on the CIFAR-10 dataset. As shown in Figure 3**a**, the training accuracy of the noisy network is extremely stable while the validation accuracy of the target SNN grows erratically in the first 30 epochs. It is reasonable as the noise probability is zero in the first 30 epochs and the network is purely analog without spike injections. With the noise injection of 10% spikes, the validation accuracy of SNN increases rapidly around 30-th epoch at the expense of the training accuracy drop of the noisy network. That means the noisy network begins to transform into target

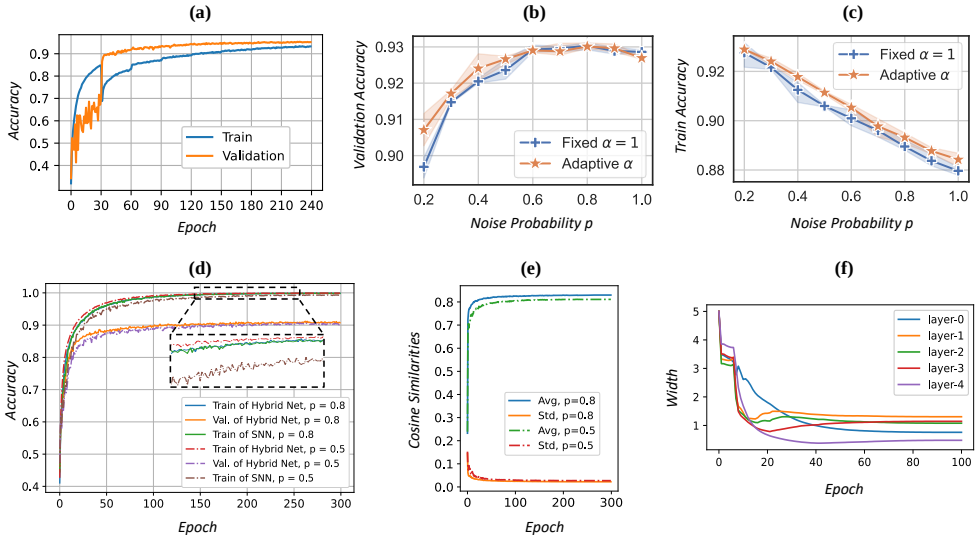

Figure 3: **(a)** explores the evolution procedure by observing accuracy change with increasing spike noise. **(b)** and **(c)** examine the accuracy for different fixed p selections during the test and training, respectively. **(d)** and **(e)** shows the similarities between the noisy network and embedded SNN when training with fixed p. **(f)** manifests the width change in the image reconstruction task.

SNNs. Interestingly, for the noise injection at the 60-th and 90-th epoch, the training accuracy is improved actually which indicates the small account of spikes in the first injection is enough for the dynamics of SNNs and the analog activations may become the obstacle for fast convergence instead. Then we explores the effect of different choices of fixed p in Figure 3**b** and Figure 3**c**. Generally, low p will block the evolution into SNNs through high analog activations while excessive p could not achieve the best generalization performance.

5.5 Network Evolution

To reveal the evolution of the noisy network, we visualize accuracy changes of the noisy network and the embedded SNN with shared weights during training under $p = 0.8$ and $p = 0.5$. As shown in Figure 3**d**, the accuracy curves of SNNs and the noisy network are extremely close in both cases. It shows that the noisy network exhibits consistency in network prediction with the embedded SNN. Furthermore, we record layer-wise activations of the noisy network and the embedded SNN for each sample, and calculate the average cosine similarities S over all layers after each training epoch with ASGL. Panel (**e**) reports the mean and standard deviation of S across all samples in the training set of CIFAR-10. According to the results, even with shared weights, the hybrid network initially has relatively low overall similarities, but after training with ASGL, the hybrid network shifts toward SNN, and the similarities increase to about 0.8. The decremental standard deviation also verifies the effectiveness of ASGL. We also evaluate the evolution of such a noisy network by observing the change of learnable width α (Figure 3**f**) in image reconstruction task. The width α declines steadily and converges to respective values across all layers.

6 Conclusion

In this paper, we propose a novel training method called *ASGL* to develop deep SNNs. Different from the typical surrogate gradient learning, our method circumvents the gradient mismatching problem naturally and updates weights adaptively with the random noise injection in spikes. Specifically, we train a special hybrid network with a mixture of spike and analog signals where only the analog part is involved in the calculation of gradients. In this way, the hybrid network will learn the optimal shapes of the activation functions against the spike noise and evolve into SNN. To validate the effectiveness and generalization of the proposed method, we analyze theoretically the evolution from hybrid networks to SNNs. Besides, we conduct extensive experiments on various benchmark datasets. Experimental result shows our method achieves state-of-the-art performance across all the tested datasets. Meanwhile, it exhibits surprising robustness for different width selections.

REFERENCES

Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *CVPR*, pp. 7243–7252, 2017.

Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatze, and Yiannis Andreopoulos. Graph-based spatio-temporal feature learning for neuromorphic vision sensing. *IEEE Transactions on Image Processing*, 29:9084–9098, 2020.

Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. Medleydb: A multitrack dataset for annotation-intensive mir research. In *ISMIR*, volume 14, pp. 155–160, 2014.

Sander M Bohte, Joost N Kok, and Han La Poutre. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1-4):17–37, 2002.

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.

Benjamin Cramer, Yannik Stradmann, Johannes Schemmel, and Friedemann Zenke. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *ICLR*, 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.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

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 *ICCV*, pp. 2661–2671, 2021b.

Isha Garg, Sayeed Shafayet Chowdhury, and Kaushik Roy. Dct-snn: Using dct to distribute spatial information over time for learning low-latency spiking neural networks. *arXiv preprint arXiv:2010.01795*, 2020.

Pengjie Gu, Rong Xiao, Gang Pan, and Huajin Tang. Stca: Spatio-temporal credit assignment with delayed feedback in deep spiking neural networks. In *IJCAI*, pp. 1366–1372, 2019.

Robert Gütig. Spiking neurons can discover predictive features by aggregate-label learning. *Science*, 351(6277), 2016.

Jesse Hagenaars, Federico Paredes-Vallés, and Guido De Croon. Self-supervised learning of event-based optical flow with spiking neural networks. *Advances in Neural Information Processing Systems*, 34:7167–7179, 2021.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015.

Weihua He, YuJie Wu, Lei Deng, Guoqi Li, Haoyu Wang, Yang Tian, Wei Ding, Wenhui Wang, and Yuan Xie. Comparing snns and rnns on neuromorphic vision datasets: similarities and differences. *Neural Networks*, 132:108–120, 2020.

Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 14:424, 2020.

Jinseok Kim, Kyungsu Kim, and Jae-Joon Kim. Unifying activation-and timing-based learning rules for spiking neural networks. *NeurIPS*, 33:19534–19544, 2020.

Alexander Kugele, Thomas Pfeil, Michael Pfeiffer, and Elisabetta Chicca. Efficient processing of spatio-temporal data streams with spiking neural networks. *Frontiers in Neuroscience*, 14:439, 2020.

Souvik Kundu, Gourav Datta, Massoud Pedram, and Peter A Beerel. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3953–3962, 2021.

Michael S Lewicki. Efficient coding of natural sounds. *Nature neuroscience*, 5(4):356–363, 2002.

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.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ANN: towards efficient, accurate spiking neural networks calibration. In *ICML*, volume 139, pp. 6316–6325, 2021a.

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, 2021b.

Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *NeurIPS*, 34, 2021c.

Sen Lu and Abhronil Sengupta. Exploring the connection between binary and spiking neural networks. *Frontiers in Neuroscience*, 14:535, 2020.

Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345 (6197):668–673, 2014.

Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29(7):3227–3235, 2017.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pp. 7197–7206. PMLR, 2020.

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.

Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

Nicolas Perez-Nieves and Dan Goodman. Sparse spiking gradient descent. *Advances in Neural Information Processing Systems*, 34, 2021.

Nicolas Perez-Nieves, Vincent CH Leung, Pier Luigi Dragotti, and Dan FM Goodman. Neural heterogeneity promotes robust learning. *Nature communications*, 12(1):1–9, 2021.

Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra. Timbre analysis of music audio signals with convolutional neural networks. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 2744–2748. IEEE, 2017.

Nitin Rathi and Kaushik Roy. DIET-SNN: direct input encoding with leakage and threshold optimization in deep spiking neural networks. *CoRR*, abs/2008.03658, 2020.

Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In *ICLR 2020,*. OpenReview.net, 2020.

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

Arash Samadi, Timothy P Lillicrap, and Douglas B Tweed. Deep learning with dynamic spiking neurons and fixed feedback weights. *Neural computation*, 29(3):578–602, 2017.

William Severa, Craig M Vineyard, Ryan Dellana, Stephen J Verzi, and James B Aimone. Training deep neural networks for binary communication with the whetstone method. *Nature Machine Intelligence*, 1(2):86–94, 2019.

Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *arXiv preprint arXiv:1810.08646*, 2018.

Qinyi Wang, Yexin Zhang, Junsong Yuan, and Yilong Lu. Space-time event clouds for gesture recognition: From rgb cameras to event cameras. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1826–1835. IEEE, 2019.

Colin Wei, Sham Kakade, and Tengyu Ma. The implicit and explicit regularization effects of dropout. In *International Conference on Machine Learning*, pp. 10181–10192. ICML, 2020.

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*, pp. 1–15, 2021a. doi: 10.1109/TNNLS.2021.3095724.

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.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1311–1318, 2019.

Zhenzhi Wu, Hehui Zhang, Yihan Lin, Guoqi Li, Meng Wang, and Ye Tang. Liaf-net: Leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing. *IEEE Transactions on Neural Networks and Learning Systems*, 2021b.

Timo C Wunderlich and Christian Pehle. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports*, 11(1):1–17, 2021.

Qu Yang, Jibin Wu, Malu Zhang, Yansong Chua, Xinchao Wang, and Haizhou Li. Training spiking neural networks with local tandem learning. *NeurIPS*, 2022.

Yukun Yang, Wenrui Zhang, and Peng Li. Backpropagated neighborhood aggregation for accurate training of spiking neural networks. In *ICML*, pp. 11852–11862, 2021.

Man Yao, Huanhuan Gao, Guangshe Zhao, Dingheng Wang, Yihan Lin, Zhaoxu Yang, and Guoqi Li. Temporal-wise attention spiking neural networks for event streams classification. In *ICCV*, pp. 10221–10230, 2021.

Bojian Yin, Federico Corradi, and Sander M Bohté. Effective and efficient computation with multiple-timescale spiking recurrent neural networks. In *International Conference on Neuromorphic Systems 2020*, pp. 1–8, 2020.

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

Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *NeurIPS 2020, 2020*, 2020.

Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *AAAI 2021*, pp. 11062–11070. AAAI Press, 2021.

A Appendix

A.1 Spike-based Backpropagation

Firstly, we decompose the gradient of aggregate loss over the whole time window into different target time steps t^*:

$$\nabla \boldsymbol{W}^l = \frac{\partial \mathcal{L}(\overline{\boldsymbol{c}}^L, \boldsymbol{y})}{\partial \overline{\boldsymbol{c}}^L} \frac{\partial \overline{\boldsymbol{c}}^L}{\partial \boldsymbol{W}^l} = -\frac{(\boldsymbol{y}^T - \hat{\boldsymbol{y}}^T)}{N} \sum_{t^*=1}^{N} \frac{\partial \overline{\boldsymbol{c}}^L[t^*]}{\partial \boldsymbol{W}^l} \tag{14}$$

To obtain the expression of $\frac{\partial \overline{\boldsymbol{c}}^L[t^*]}{\partial \boldsymbol{W}}$, we assign the credits for $\boldsymbol{c}^L[t^*]$ into the membrane potential $\boldsymbol{u}^l[t]$ at all time steps satisfying $t \leq t^*$:

$$\begin{aligned} \frac{\partial \overline{\boldsymbol{c}}^L[t^*]}{\partial \boldsymbol{W}} &= \sum_{k=0}^{t^*} \frac{\partial \overline{\boldsymbol{c}}^L[t^*]}{\partial \boldsymbol{u}^l[t]} \frac{\partial \boldsymbol{u}^l[t]}{\partial \boldsymbol{c}^l[t]} \frac{\partial \boldsymbol{c}^l[t]}{\partial \boldsymbol{W}^l} \\ &= \sum_{t=0}^{t^*} \frac{\partial \overline{\boldsymbol{c}}^L[t^*]}{\partial \boldsymbol{u}^l[t]} \frac{\partial \boldsymbol{c}^l[t]}{\partial \boldsymbol{W}^l} \end{aligned} \tag{15}$$

where $\frac{\partial \boldsymbol{c}^l[t]}{\partial \boldsymbol{W}}$ is a three-dimensional tensor about the afferent spikes $\boldsymbol{s}^{l-1}[t]$. For simplification, we denote $\frac{\partial \overline{\boldsymbol{c}}^L[t^*]}{\partial \boldsymbol{u}^l[t]}$ as $\boldsymbol{\delta}^l[t]$. When $t < t^*$ and $l < L-1$, $\boldsymbol{\delta}^l[t]$ could be calculated as follows:

$$\begin{aligned} \boldsymbol{\delta}^l[t] &= \boldsymbol{\delta}^l[t+1] \frac{\partial \boldsymbol{u}^l[t+1]}{\partial \boldsymbol{u}^l[t]} + \boldsymbol{\delta}^{l+1}[t] \frac{\partial \boldsymbol{u}^{l+1}[k]}{\partial \boldsymbol{u}^l[k]} \\ \frac{\partial \boldsymbol{u}^l[t+1]}{\partial \boldsymbol{u}^l[t]} &= \gamma \, \text{diag}\left(1 - \boldsymbol{s}^l[t] - \boldsymbol{u}^l[t] \odot \Theta^{'}(\boldsymbol{u}^l[t])\right) \\ \frac{\partial \boldsymbol{u}^{l+1}[t]}{\partial \boldsymbol{u}^l[t]} &= \boldsymbol{W}^{l+1} \text{diag}\left(\Theta^{'}(\boldsymbol{u}^l[t])\right) \end{aligned} \tag{16}$$

where $\Theta^{'}(\boldsymbol{x}) = [\Theta^{'}(x_1), \Theta^{'}(x_2), ..., \Theta^{'}(x_n)]^T$ represents the element-wise partial on the colum vector $\boldsymbol{x}$. As for the boundary condition of the layer $L-1$ and time step t^*, we could obtain the expression of $\boldsymbol{\delta}^l[t]$:

$$\boldsymbol{\delta}^l[t] = \begin{cases} \boldsymbol{\delta}^L[t+1] \frac{\partial \boldsymbol{u}^L[t+1]}{\partial \boldsymbol{u}^L[t]} & \text{if } l = L-1 \text{ and } t < t^* \\ \boldsymbol{\delta}^{l+1}[t^*] \frac{\partial \boldsymbol{u}^{l+1}[t^*]}{\partial \boldsymbol{u}^l[t^*]} & \text{if } t = t^* \text{and } l < L-1 \\ \boldsymbol{W}^L \text{diag}\left(\Theta^{'}(\boldsymbol{u}^l[t])\right) & \text{if } t = t^* \text{and } l = L-1 \end{cases} \tag{17}$$

Then the full backward flow through time of SNNs with the LIF model could be obtained with Equations (14) to (17).

A.2 C-LIF MODEL

For the instrumental recognization, we adopt the current-base LIF model (C-LIF) (Gütig, 2016) as the basic computational unit for a fair comparison. The iterative form of C-LIF could be presented as:

$$\begin{aligned}
\boldsymbol{s}^l[t] &= \Theta\left(\boldsymbol{u}^l[t] - \vartheta\right) \\
\boldsymbol{c}^l[t] &= \boldsymbol{u}_0 \odot \boldsymbol{W}^l \boldsymbol{s}^{l-1}[t] \\
\boldsymbol{u}^l[t] &= \boldsymbol{m}^l[t] - \boldsymbol{v}^l[t] - \boldsymbol{e}^l[t] \\
\boldsymbol{v}^l[t] &= \beta_v \boldsymbol{v}^l[t-1] + \boldsymbol{c}^l[t] \\
\boldsymbol{m}^l[t] &= \beta_m \boldsymbol{m}^l[t-1] + \boldsymbol{c}^l[t] \\
\boldsymbol{e}^l[t] &= \beta_m \boldsymbol{m}^l[t-1] + \vartheta \boldsymbol{s}^l[t-1] \\
\Theta(x) &= \begin{cases} 1, & x \geq \vartheta \\ 0, & x < \vartheta \end{cases}
\end{aligned} \tag{18}$$

where $\boldsymbol{u}_0$ is the normalization factor. $\boldsymbol{m}^l[t] - \boldsymbol{v}^l[t]$ models the current integration of the synapse with the double exponential function. $\boldsymbol{e}^l[t]$ simulates the refractory period in spiking neurons. The other symbols keep consistent with the standard LIF model.

A.3 EXPERIMENTAL DETAILS

Table 6: Different hyperparameters related with p.

Dataset	p	ζ	Milestones (Epochs)
CIFAR-10 / DVS-CIFAR10 / DVS128 Gesture	0.8	1	NA (300)
Tiny-ImageNet	0.9	1	NA (300)
CIFAR-100	0.6	0.8	90-th, 210-th, 270-th, 285-th (300)
SHD	0.8	0.9	30-th, 70-th, 90-th, 95-th (100)
MedlyDB	0.5	0.9	30-th, 70-th, 90-th, 95-th (100)

We use ADAM with the initial learning rate $\lambda = 0.1$ for CIFAR100 and SGD with the initial learning rate of $\lambda = 0.1$ for CIFAR10 dataset. As done in (Li et al., 2021c), we use AutoAugment (Cubuk et al., 2018) and Cutout (DeVries & Taylor, 2017) for data augmentation in both static image datasets. Meanwhile, a cyclic cosine annealing learning rate scheduler is adopted. For the SHD dataset, we discretize the time into 250 time steps and decrease noise probability starting from 0.2. The corresponding network architecture is $700-240-20$ while the neurons in the middle layer are connected with recurrent synapses. For the MedlyDB dataset, we increase the noise probability at 30-th, 70-th, 90-th, 95-th epoch with the discretization of 500 time steps. In particular, we update p as $1-(1-p)\cdot\zeta$ at each milestone and make the ratio of analog activations attenuation at the rate of ζ. All the p and ζ we use for each dataset are shown in Table 6 unless otherwise specified. For the results of Table 5, we provide detailed statistics and configurations in Table 11.

A.4 EXPERIMENTS ON MEDLYDB, DVS128 GESTURE, AND TINY IMAGENET

Performance on Tiny-ImageNet. Tiny-ImageNet contains 200 categories and 100,000 64×64 colored images for training, which is a more challenging static image dataset than CIFAR datasets. Here, we use $h_\alpha(x) = \frac{1}{2}\tanh(\alpha x) + \frac{1}{2}$ as the surrogate forwarding function. The initial width α and decay γ is set as 2.5 and 0.5 respectively. As shown in Table 8, ASGL still achieve competitive results compared to other methods using only 4 time steps which further verify the effectiveness of ASGL.

Performance on MedlyDB. In this experiment, we explore the instruments recognization task with various music pieces in different melodies and styles. Specifically, as done in (Gu et al., 2019), we subtract the subset of MedlyDB which contains the monophonic stems of 10 instruments. To test our algorithm in sparse spike patterns, we adopt the efficient coding scheme based on the sparse representation theory (Lewicki, 2002). Moreover, we use the same metric, the same network structure

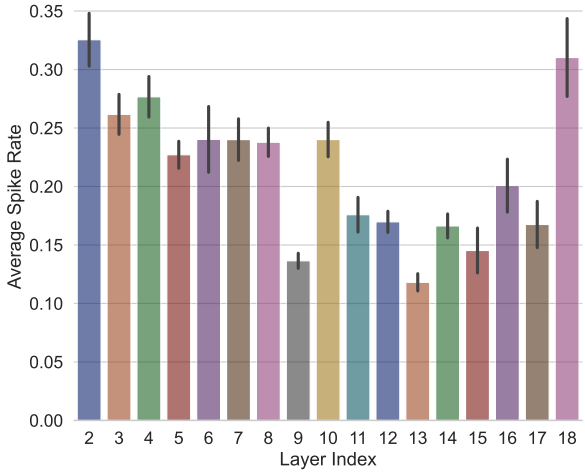

Figure 4: The average spike counts of each layer.

Table 7: Classification Performance on DVS128 Gesture

Method	Type	Architecture	Acc.(%)
SLAYER (Shrestha & Orchard, 2018)	Surrogate Gradient	SNN(8 layers)	93.64
STBP in DVS (He et al., 2020)		SNN(8 layers)	93.40
STBP-tdBN (Zheng et al., 2021)		SNN(ResNet17)	96.87
Temporal-wise Attention (Yao et al., 2021)		SNN(8 layers)	95.49
PLIF-SNN (Fang et al., 2021b)		SNN(8 layers)	97.57
DECOLLE (Kaiser et al., 2020)	Online Local Learning	SNN(4 layers)	95.54
Streaming rollouts (Kugele et al., 2020)	Conversion	SNN (DenseNet)	95.56
PointNet-like ANN (Wang et al., 2019)	Gradient training	DNN	95.32
RG-CNN (Bi et al., 2020)		DNN	97.20
Ours	ASGL	SNN(8 layers)	**97.90**

(384-700-10), and the same C-LIF neuron with the previous work for a fair comparison. The results show our method outperforms the others on most metrics except the Recall rate of the specialized CNN (Pons et al., 2017). which designs special convolutional kernels.

Performance on DVS128 Gesture. DVS128 Gesture is a challenging neuromorphic dataset that records 11 gestures performed by 29 different participants under three lighting conditions. The dataset comprises 1,342 samples with an average duration of 6.5 ± 1.7 s and all samples are split into a training set (1208 samples) and test set (134 samples). Considering the long sample duration and the limited sample size, we follow the RCS approach (Yao et al., 2021) that randomly selects the starting point of the sample to maximize the use of the dataset. The time step N is set to be 60 and the network receives only one slice at each step, where the temporal resolution of each slice is adjusted to 25ms according to the tuning method in (He et al., 2020). In Table 7, our method has achieved

Table 8: Comparasion on Tiny-ImageNet dataset.

Method	Type	Architecture	Time steps	Top-1 Acc.
Spike-thrift (Kundu et al., 2021)	Hybrid	VGG-16	150	51.92
DCT (Garg et al., 2020)	Hybrid	VGG-13	125	56.90
SNN Calibration (Li et al., 2021b)†	Conversion	VGG-16	32	53.96
QCFS (Bu et al., 2021)†	Conversion	VGG-16	32	53.54
Online LTL (Yang et al., 2022)	Tandem Learning	VGG-13	16	54.82
Offline LTL (Yang et al., 2022)	Tandem Learning	VGG-13	16	55.37
Ours	**ASGL**	VGG-13	**4**	**56.57**

† Those results are reproduced by (Yang et al., 2022) through publicly available codes.

the state-of-the-art performance (97.90 $\%$) without a larger network (e.g., ResNet17, DenseNet), outperforming the directly-trained approaches based on surrogate gradient. Even compared with the specially-designed DNN approaches for neuromorphic data, our model also performs better.

A.5 RECCURENT CONNECTIONS

To enhance the memory capacity in SNNs at the temporal domain, synatic recurrence is adopted widely distinguish from the internal dynamics with decay mechanism in spiking neurons. The basic equation for such an external recurrence could be given by:

$$\frac{d\boldsymbol{c}^l}{dt} = -\underbrace{\frac{\boldsymbol{c}^l(t)}{\tau_{\text{syn}}}}_{\text{exp. decay}} + \underbrace{\boldsymbol{W}^l \boldsymbol{s}^{l-1}(t)}_{\text{feedforward}} + \underbrace{\boldsymbol{V}^l \boldsymbol{s}^l(t)}_{\text{recurrent}}, \tag{19}$$

where the terms of decay τ_{syn} and τ_m in Equation (1) both contribute to the internal recurrence. And the synaptic recurrence with weight $\boldsymbol{V}^l$ enhance the temporal memory.

A.6 ENERGY ESTIMATION

In this section, we visualize the spike counts of each layer in spiking ResNet-18 shown in Figure 4 and provide the estimated energy by counting synaptic operations (SOP) compared to the ANN counterpart. Especially, the SOP with MAC presented in ANNs is constant given a specified structure. However, the SOP in SNN is executed by AC with lower power consumption and varies with the spike sparsity. For SNNs, the total synaptic operation with accumulation N_{AC} is defined as:

$$N_{\text{AC}} = \sum_{t=1}^{T} \sum_{l=1}^{L-1} \sum_{i=1}^{N^l} f_i^l s_i^l[t] \tag{20}$$

where fan-out f_i^l is the number of outgoing connections to the subsequent layer. N_i^l is the neuron number of the l-th layer. ANNs, similar synaptic operation N_{MAC} with more exexpensive multiply-accumulate is defined as:

$$N_{\text{MAC}} = \sum_{l=1}^{L-1} \sum_{i=1}^{N^l} f_i^l \tag{21}$$

Specially, we use MAC to estimate the energy cost of the first layer as the direct current input without explicit encoding is adopted in our experiments on static images. For Here, we select 1024 samples randomly and estimate the average SOP for SNNs. Meanwhile, we measure 32-bit floating-point AC by 0.9 pJ per operation and 32-bit floating-point MAC by 4.6 pJ per operation as done in (Han et al., 2015). The experimental result shows the SNN achieves 94.11% classification accuracy under two time steps on CIFAR-10 with only 8.96% energy consumption compared to the ANN with the same architecture. We follow the convention of the neuromorphic computing community by counting the total synaptic operations to estimate the computation overhead of SNN models compared to their ANN counterparts (Merolla et al., 2014).

A.7 ABLATION STUDY ON RESNET-19

Table 9: Classification Performance on Music Instrument Dataset

Method	Type	Encoding Method	#Param.	Recall(%)	Precision(%)	F1 score(%)
CNN (Pons et al., 2017)	Direct BP	Spectrogram	76.9w	**99.21**	95.94	97.51
LSTM	Direct BP	Spectrogram	27.6w	93.31	96.08	94.62
FA (Samadi et al., 2017)	DSNN (Feedback Alignment)	Spikegram	27.6w	86.56	75.62	80.73
STCA (Gu et al., 2019)	DSNN (Surrogate Gradient)	Spikegram	27.6w	97.29	97.23	97.25
Ours	DSNN (ASGL)	Spikegram	**27.6w**	98.58	**98.52**	**98.59**

We have added the results of ResNet-19 (Zheng et al., 2021) under time-steps $N = 3$ in Table 11 (rectangular function) and Table 12 (Dspike function (Li et al., 2021c)). Specifically, we test the effect of different width initializations. For fairness, we use the same optimizer settings and weight initialization. All the experimental details are provided in the Appendix. Notably, ResNet-19 has

Table 10: Comparsion with Surrogate Gradient Learning.

Method	SG ($\alpha = 0.5$)	SG ($\alpha = 1$)	SG ($\alpha = 2.5$)	SG ($\alpha = 5$)	SG ($\alpha = 7.5$)	SG ($\alpha = 10$)
Acc.	10.00 ± 0.00	10.00 ± 0.00	87.64 ± 0.28	81.89 ± 0.57	66.00 ± 1.82	53.65 ± 1.12
Method	ASGL+SG ($p = 0.5, \alpha$=1)	ASGL+SG ($p = 0.5, \alpha$=2.5)	ASGL+SG($p = 0.5, \alpha$=5)	**ASGL(ζ=0.2)**	**ASGL(ζ=0.5)**	**ASGL(ζ=0.8)**
Acc.	10.00 ± 0.00	85.72 ± 0.35	53.41 ± 2.13	$\mathbf{89.62 \pm 0.20}$	$\mathbf{89.69 \pm 0.17}$	$\mathbf{88.83 \pm 0.09}$

approximate $10\times$ operations than ResNet-18 (Li et al., 2021c). Therefore we train 100 epochs for each case considering time cost. **As shown in both tables, ASGL shows robustness surprisingly on different width initializations compared to surrogate gradient with rectangular function and Dspike function.** This is the main advantage of ASGL which could save the cost of hyperparameter selection in practice. From Table 12, we could find that the Dspike function shows certain robustness with respect to the rectangular function considering the result of α = 2.5 from two tables. However, it could be improved further by ASGL when $\alpha \leq 0.5$ shown in Table 12.

Experimental Setting: We train each model in ablation study for 100 epochs with initial learning rate of 0.1. We use SGD with the momentum of 0.9 across all experiments. The weight decay is set as 5e-4. As done in (Li et al., 2021c), we use AutoAugment (Cubuk et al., 2018) and Cutout (DeVries & Taylor, 2017) for data augmentation. Meanwhile, a cyclic cosine annealing learning rate scheduler is adopted. Besides, we use 128 as batch size during training. Both threshold and decay are set to 0.5. Specially, we decrease the noise rate at 20-th, 40-th, 60-th, 80-th and 95-th epoch with different ζ listed in both tables.

Table 11: Comparsion between *ASGL* and *Surrogate Gradient* with rectangular functions under different width initializations.

Width	Surrogate Gradient	ASGL		
		$\xi = 0.2$	$\xi = 0.5$	$\xi = 0.8$
$\alpha = 0.5$	93.19	93.89	93.93	94.11
$\alpha = 1.0$	93.78	93.83	94.30	94.15
$\alpha = 2.5$	90.68	93.71	94.09	93.89
$\alpha = 5.0$	62.34	93.61	93.53	93.08
$\alpha = 10.0$	30.85	92.48	93.53	93.00

Table 12: Comparsion between *ASGL* and *Surrogate Gradient* with Dspike functions under different width initializations.

Width	Surrogate Gradient	ASGL		
		$\xi = 0.2$	$\xi = 0.5$	$\xi = 0.8$
$\alpha = 0.10$	82.48	90.81	91.23	89.24
$\alpha = 0.12$	89.42	90.83	92.65	91.34
$\alpha = 0.14$	91.41	93.20	94.03	93.36
$\alpha = 0.5$	93.93	94.02	94.34	94.28
$\alpha = 1.0$	93.85	93.83	94.30	93.15
$\alpha = 2.5$	93.61	93.60	93.98	93.98

A.8 WIDTH UPDATE

We evaluate the evolution of such a noisy network by observing the change of learnable width α. For the image reconstruction (Figure 5**a**), the width α declines steadily across all layers. It indicates the injection of spike noise will force the noisy network to evolve into the target SNN and then optimize the target SNN in a coupled manner. Furthermore, Figure 5**b** shows the change of α in CIFAR-10 classification. Interestingly, the width α of the last layer in CIFARNet increases while others descend consistently. It may result from the coupling training with both goals of reducing the loss of SNN and minimizing the distance between the noisy network and SNN. Besides, it indicates that the width update should change dynamically according to different layers of the network and different training epochs.

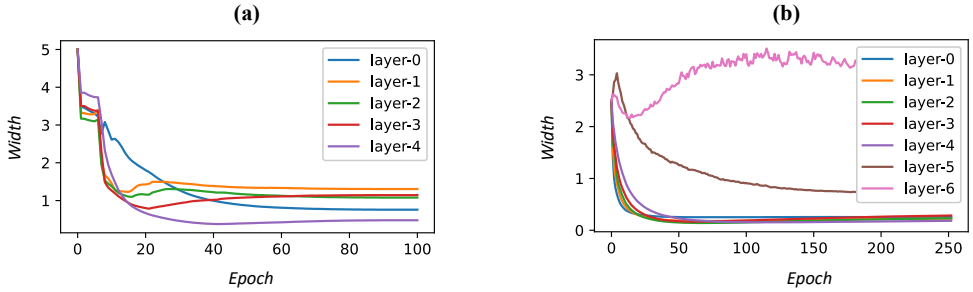

Figure 5: Fig. **(a)** and **(b)** visualize the width update in image reconstruction and image recognization, respectively.

A.9 Theory Analysis

Table 13: The symbols and corresponding definitions (explanations).

Symbol	Defination
$\boldsymbol{\ell}$	loss function
$\boldsymbol{s}$ or $\boldsymbol{s}^0$	input spike pattern
$\boldsymbol{f}^l$	the l-th spiking layer with $\Theta(\hat{\boldsymbol{u}}^l)$
$\boldsymbol{F}^l$	$\boldsymbol{f}^1 \circ \boldsymbol{f}^2 \circ ... \circ \boldsymbol{f}^l$
$\boldsymbol{s}^l$	the output of $\boldsymbol{F}^l$ with fully spike propagation
$\boldsymbol{g}^l$	the l-th noise spiking layer with random mask $\hat{\boldsymbol{m}}^l$
$\boldsymbol{G}^l$	$\boldsymbol{g}^L \circ \boldsymbol{g}^{L-1} \circ ... \circ \boldsymbol{g}^{l+1}$
p	noise probability controlling the percent of spike mode

Proposition A.1 *Minimizing the loss of noisy network $\boldsymbol{\ell}_{noise}(\boldsymbol{F}, \boldsymbol{s})$ can be approximated into minimizing the loss of the embedded SNN $\boldsymbol{\ell}_{snn}(\boldsymbol{F}, \boldsymbol{s})$ regularized by the layerwise distance between $\Theta(\hat{\boldsymbol{u}}^l)$ and $H_\alpha(\hat{\boldsymbol{u}}^l)$.*

$$\boldsymbol{\ell}_{noise}(\boldsymbol{F}, \boldsymbol{s}) \approx \boldsymbol{\ell}_{snn}(\boldsymbol{F}, \boldsymbol{s}) + \frac{1-p}{2p} \sum_{l=1}^{L} \left\langle \boldsymbol{C}^l, diag(H_\alpha(\hat{\boldsymbol{u}}^l) - \Theta(\hat{\boldsymbol{u}}^l))^{\odot 2} \right\rangle \tag{22}$$

Derivation. Suppose $\boldsymbol{G}^l \circ \boldsymbol{F}^l$ as the hybrid network using spike activations $\Theta(x)$ in the preceding l layers and hybrid activations $H_\alpha(x)$ after l-th layer. Morever, $\boldsymbol{s}^l$ is the output spikes at l-th layer across all time steps and neurons. The detailed symbol defination and descriptions are provided in Table 13. Then we can have:

$$\begin{aligned}
\boldsymbol{\ell}_{\text{noise}}(\boldsymbol{F}, \boldsymbol{s}) &= \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s})) + \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s}, \hat{\boldsymbol{m}})) - \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s}))] \\
&= \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s})) + \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{\ell}(\boldsymbol{G}^0(\boldsymbol{s}^0, \hat{\boldsymbol{m}})) - \boldsymbol{\ell}(\boldsymbol{G}^L(\boldsymbol{s}^L))] \\
&= \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s})) + \mathbb{E}_{\hat{\boldsymbol{m}}} \left[\sum_{l=1}^{L} \left(\boldsymbol{\ell}(\boldsymbol{G}^{l-1}(\boldsymbol{s}^{l-1}, \hat{\boldsymbol{m}})) - \boldsymbol{\ell}(\boldsymbol{G}^l(\boldsymbol{s}^l, \hat{\boldsymbol{m}})) \right) \right] \\
&= \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s})) + \sum_{l=1}^{L} \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{\ell}(\boldsymbol{G}^{l-1}(\boldsymbol{s}^{l-1}, \hat{\boldsymbol{m}})) - \boldsymbol{\ell}(\boldsymbol{G}^l(\boldsymbol{s}^l, \hat{\boldsymbol{m}}))] \\
&= \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s})) + \sum_{l=1}^{L} \boldsymbol{R}(\boldsymbol{G}^l, \boldsymbol{s}^l)
\end{aligned} \tag{23}$$

Then we adopt Taylor expansion on $\boldsymbol{s}^l$ inspired from (Wei et al., 2020) to analyze the effect of the perturbation of $\hat{\boldsymbol{m}}$ at l-th layer. From Equation (10) and Equation (3), we could obtain:

$$\begin{aligned}
\boldsymbol{G}^{l-1}\left(\boldsymbol{s}^{l-1}, \hat{\boldsymbol{m}}\right) &= \boldsymbol{G}^l\left((\boldsymbol{1} - \hat{\boldsymbol{m}}^l) \odot H_\alpha(\hat{\boldsymbol{u}}^l) + \hat{\boldsymbol{m}}_b^l \odot \Theta(\hat{\boldsymbol{u}}^l), \hat{\boldsymbol{m}}\right) \\
&= \boldsymbol{G}^l\left((\boldsymbol{1} - \hat{\boldsymbol{m}}^l) \odot (H_\alpha(\hat{\boldsymbol{u}}^l) - \Theta(\hat{\boldsymbol{u}}^l)) + \boldsymbol{s}^l, \hat{\boldsymbol{m}}\right)
\end{aligned} \tag{24}$$

Here, we denote $\Delta = (\mathbf{1} - \hat{\boldsymbol{m}}^l) \odot (H_\alpha(\hat{\boldsymbol{u}}^l) - \Theta(\hat{\boldsymbol{u}}^l))$ for simplication. As the expection of Δ is zero and $|\Delta|$ is bounded in $[0, \max(\frac{1-p}{2p}, \frac{1}{2})]$, we adopt Taylor expansion around $\Delta = \mathbf{0}$ and approximate $\boldsymbol{R}(\boldsymbol{G}^l, \boldsymbol{s}^l)$ here:

$$\begin{aligned}
\boldsymbol{R}(\boldsymbol{G}^l, \boldsymbol{s}^l) &= \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{\ell}(\boldsymbol{G}^{l-1}(\boldsymbol{s}^{l-1}, \hat{\boldsymbol{m}})) - \boldsymbol{\ell}(\boldsymbol{G}^l(\boldsymbol{s}^l, \hat{\boldsymbol{m}}))] \\
&= \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{\ell}(\boldsymbol{G}^l(\boldsymbol{s}^l + \Delta, \hat{\boldsymbol{m}})) - \boldsymbol{\ell}(\boldsymbol{G}^l(\boldsymbol{s}^l, \hat{\boldsymbol{m}}))] \\
&\approx \mathbb{E}_{\hat{\boldsymbol{m}}}\left[D(\boldsymbol{\ell} \circ \boldsymbol{G}^l)[\boldsymbol{s}^l]\Delta + \frac{1}{2}\Delta^T \left(D^2(\boldsymbol{\ell} \circ \boldsymbol{G}^l)[\boldsymbol{s}]\right)\Delta\right]
\end{aligned} \tag{25}$$

As Δ is a zero-mean vector, we discard the first-order term for expectation calculation here:

$$\boldsymbol{R}(\boldsymbol{G}^l, \boldsymbol{s}^l) \approx \mathbb{E}_{\hat{\boldsymbol{m}}}\left[\frac{1}{2}\Delta^T \left(D^2(\boldsymbol{\ell} \circ \boldsymbol{G}^l)[\boldsymbol{s}]\right)\Delta\right] \tag{26}$$

Then we could take the expectation over Δ:

$$\begin{aligned}
&\boldsymbol{R}\left(\boldsymbol{G}^l, \boldsymbol{s}^l\right) \approx \frac{1}{2}\left\langle D^2\left(\boldsymbol{\ell} \circ \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{G}^l]\right)[\boldsymbol{s}], \mathbb{E}_\Delta\left[\Delta\Delta^T\right]\right\rangle \\
&= \frac{1-p}{2p}\left\langle D^2\left(\boldsymbol{\ell} \circ \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{G}^l]\right)[\boldsymbol{s}], \mathrm{diag}\left(\left(H_\alpha(\hat{\boldsymbol{u}}^l) - \Theta(\hat{\boldsymbol{u}}^l)\right)^{\odot 2}\right)\right\rangle
\end{aligned} \tag{27}$$

Here, only the diagonal elements in the covariance matrix $\mathbb{E}_\Delta\left[\Delta\Delta^T\right]$ are non-zero because of the independent sampling strategy presented in $\hat{\boldsymbol{m}}$. For the second order term, we could just take it as a constant like (Nagel et al., 2020). Therefore, by substiting Equation (27) into Equation (23), we get:

$$\boldsymbol{\ell}_{\text{noise}}(\boldsymbol{F}, \boldsymbol{s}) \approx \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s})) + \frac{1-p}{2p}\sum_{l=1}^{L}\left\langle \boldsymbol{C}^l, \mathrm{diag}(H_\alpha(\hat{\boldsymbol{u}}^l) - \Theta(\hat{\boldsymbol{u}}^l))^{\odot 2}\right\rangle \tag{28}$$