Adaptive Smoothing Gradient Learning for Spiking Neural Networks

Ziming Wang [1] Runhao Jiang [1] Shuang Lian [1] Rui Yan [2] Huajin Tang [1 3]

Abstract

Spiking neural networks (SNNs) with biologically inspired spatio-temporal dynamics demonstrate superior 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 a solution introduces additional smoothing error on spike firing which leads to the gradients being estimated inaccurately. Thus, how to adaptively adjust the relaxation degree and eliminate smoothing 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, The experiments on static images, dynamic event streams, speech, and instrumental sounds 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 that 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 existence of nondifferentiable spike firing poses a hindrance to the direct application of backpropagation in SNNs. To tackle 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. Additionally, 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 adoption of a smooth curve causes the gradient of a single spike to be distributed into a group of analog items in temporal neighbors (Zhang & Li, 2020), resulting in a discrepancy with the inherent dynamics of spiking neurons. This discrepancy is referred to as *gradient mismatching* in this paper. As a result, most parameters are updated in a biased manner within surrogate gradient learning, thus impeding the performance of SNNs. Besides, the level of smoothness exhibited by surrogate functions can significantly impact the network performance (Hagenaars et al., 2021; Li et al., 2021b; Leng et al., 2022).

The time-based method is the other appealing approach. By estimating the gradients based on the exact spike times, the time-based method naturally circumvents the issues of gradient mismatching encountered in surrogate gradient learning. However, to obtain the exact expression of spike time, most works (Mostafa, 2017; Zhang & Li, 2020) assume that the firing count of each neuron remains unchanged during training (Yang et al., 2021), which is challenging

[1]College of Computer Science, Zhejiang University [2]College of Computer Science, Zhejiang University of Technology [3]Research Center for Intelligent Computing Hardware, Zhejiang Lab. Correspondence to: Huajin Tang <htang@zju.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

to establish in practice. Besides, it is difficult to adapt the customized backward flows of the time-based method with auto-differential frameworks such as PyTorch, MXNet, and TensorFlow. Moreover, certain special tricks are necessary to avoid the phenomenon of dead neurons (Bohte et al., 2002). Consequently, obtaining deep SNNs using the time-based method lacks flexibility.

To address the aforementioned issues, this work 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 prototype 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 can be updated against mismatched gradients adaptively. In addition, compared to the time-based method, ASGL backpropagates errors in both spatial and temporal domains, devoid of special constraints and restart mechanisms. Notably, the method is evaluated based on 1-bit spike event packets without real-valued activations. Consequently, after training, an entirely spike-based network is yielded without sacrificing asynchronous event-based calculation on neuromorphic hardware implementations.

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 on smooth factors. Experiments demonstrate 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 learned smooth factors.

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; Yin et al., 2020). 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. (2021b) optimized the shape of the surrogate gradient function with the finite difference method to compensate for this problem. However, their approach necessitates empirical initialization of the finite difference update step and is impeded by the computational complexity associated with the method. As a result, only information from the proceeding layers is utilized in the update of surrogate functions. Other approaches have been proposed to circumvent this difficulty without employing surrogate gradients. Zhang & Li (2020) handled error backpropagation across inter-neuron and intra-neuron dependencies based on the typical time-based scheme. Additionally, the unifying of surrogate gradient and time-based method was suggested by (Kim et al., 2020) to fuse the gradients obtained from both spike generation and spike time shift. Generally, these methods are constrained by specified assumptions or coupling techniques during training. Wunderlich & Pehle (2021) first proposed to compute the exact gradients in an event-driven manner, thereby avoiding smoothing operations by solving ODEs about adjoint state variables. Nevertheless, the approach has only been validated on simple datasets with shallow networks. Yang et al. (2021) developed a novel method that propagates errors with neighborhood aggregation and updates weights in the desired direction. However, the method based on the finite difference incurs high computational costs. Severa et al. (2019) proposed progressively sharpening the bounded ReLU activation in ANN into the Heaviside function in SNN. Although yielding a similar effect with ASGL, it utterly depends on hand-craft sharpening schedulers and faces challenges in adaptive updates considering the evolution of the entire network. Different from the previous works, ASGL directly incorporates 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 $f(\boldsymbol{x}) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$, we use $D^k \boldsymbol{f}[\boldsymbol{x}]$ instead of $\frac{\partial f^{(k)}(\boldsymbol{x})}{\partial \boldsymbol{x}}$ to represent the k-th derivative of f with respect to the variable $\boldsymbol{x}$ in the absence of ambiguity. Let $< \boldsymbol{U}, \boldsymbol{V} >$ represent the Frobenius inner production between two matrices $\boldsymbol{U}$ and $\boldsymbol{V}$. 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; Yin et al., 2020) 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 $\boldsymbol{u}^l[t]$:

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

where $\gamma = 1 - 1/\tau_m$ is the leaky factor that acts as a constant forget gate through time. τ_m is the corresponding membrane time constant. The term $(1 - \boldsymbol{s}^l[t-1])$ 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 product 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 $\boldsymbol{u}^l[t]$ surpasses 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 Appendix A.3.

Readout and Loss. To perform specific tasks, it is necessary to define a readout method that matches the supervised signal $\boldsymbol{y}$. As done in the recent work (Li et al., 2021a; Rathi et al., 2020), we calculate 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. The SNN prediction is then defined as the one with the maximum average postsynaptic current naturally. Furthermore, we could define the cross-entropy loss by 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 simplification, 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 poses a significant challenge in the development of spike-based backpropagation. Formally, we usually need to backpropagate the credit for the state at a specified time step t, denoted

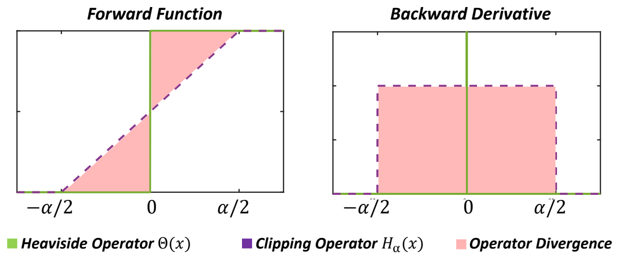

Figure 1. The forward and backward propagation of the Heaviside function (depicted by the green solid line) and the surrogate clipping function (illustrated by the purple dashed line). The red region signifies the mismatching between the Heaviside function and surrogate function, taking into account single spike emission.

as $\boldsymbol{\delta}^L[t] = \frac{\partial \boldsymbol{c}^L[t]}{\partial \boldsymbol{W}}$, across both spatial domain and temporal domain (refer to Appendix A.1 for detailed derivatives):

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

In the aforementioned equations, the partial derivative of the Heaviside function $\Theta(x)$ takes the form of the Dirac-Deta function, which assumes a value of zero for almost all x. Moreover, the Dirac-Delta function tends toward infinity when x equals zero, as depicted by the green line in Figure 1. Such characteristics directly impede the gradient backpropagation in SNNs. Consequently, many researchers resort to approximating the gradient of the Heaviside function using a predefined differentiable curve (Neftci et al., 2019; Shrestha & Orchard, 2018; Fang et al., 2021a; Zenke & Vogels, 2021). In this paper, 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. It is worth noting that the presented idea can be adapted to other alternative approximation functions. Formally, the rectangular function is defined as:

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

Here, the smooth degree of $h_\alpha(x)$ is governed by the width α while the relative steepness is determined by the temperature $\kappa = 1/\alpha$.

4.2. Design of ASGL

In surrogate gradient learning, the estimation of $\Theta^{'}(x)$, denoted by a smooth function like $h_\alpha(x)$, serves to predict the rate of loss change within a relatively larger neighborhood (Li et al., 2021b). However, utilizing a constant-width esti-

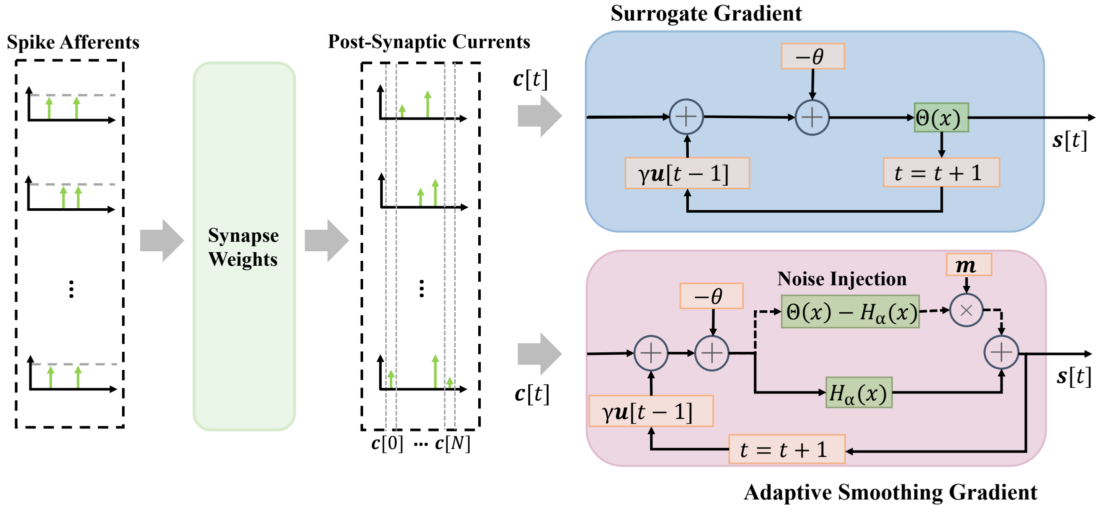

Figure 2. Comparison of the computation flow between ASGL and surrogate gradient learning. The dashed line in the purple area represents those operations that are detached with the error backpropagation while the solid line represents the matching forward and backward flow incorporated with learnable width α.

mation leads to progressive deviation from the correct direction during network training. This not only poses challenges to the network convergence but also impacts the generalization ability due to disturbed underlying neuron dynamics. Specifically, when the smoothing error decreases with reduced α , the phenomenon of dead neurons is more likely to occur throughout the network (refer to Appendix B.7 for further analysis). Essentially, this issue arises from the mismatch of $||\Theta^{'}(x) - h_\alpha(x)||$, as illustrated in the red part of Figure 1. Therefore, how to adjust the width α and eliminate such mismatching adaptively is an important problem for surrogate gradient learning. For ASGL[1], we try to address this problem without explicitly defining the surrogate gradient. The method is straightforward yet highly effective. Initially, we derive the antiderivative function of surrogate function $h_\alpha(x)$:

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

Whetstone (Severa et al., 2019) utilizes $H_\alpha(x)$ for forward calculations and $h_\alpha(x)$ for backward propagation. In this way, although there is no mismatching problem, it becomes challenging to guarantee the network dynamics evolve into that of SNNs eventually. In contrast, surrogate gradient learning employs $\Theta(x)$ and $h_\alpha(x)$ for forward calculations and backward propagation, respectively. While this approach ensures full spike communication, it introduces the problem of gradient mismatching. Consequently, **it becomes logical to seek an approach that combines the advantages of both perspectives, enabling the training of deep SNNs with matching gradients.**

[1]Code is available at https://github.com/Windere/ASGL-SNN

To implement this, the fundamental concept of ASGL entails the coupling of analog activation $H_\alpha(\boldsymbol{x})$ and binary activation $\Theta(\boldsymbol{x})$ employing a random mask, denoted as $\boldsymbol{m}$, during the forward calculation:

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

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 circumvent the gradient mismatching stemming from the surrogate function, we use function Φ to detach the gradients from spikes. Mathematically, Φ corresponds to an identical mapping that possesses a derivative $\frac{\partial \Phi(\boldsymbol{x})}{\partial \boldsymbol{x}} = \mathbf{0}$. In this way, the Heaviside function $\Theta(\boldsymbol{x})$ is taken as the spike noise without error backpropagation. To guarantee the continuous flow of gradients even when $p = 1$, we recouple both modes and only inject the difference $\Theta(\boldsymbol{x}) - H_\alpha(\boldsymbol{x})$ as noise:

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

The operator pipeline is depicted in Figure 2. Surprisingly, the core idea of ASGL is remarkably simple: replace $\Theta(\boldsymbol{u}^l[t] - \vartheta)$ with $\hat{H}_\alpha(\boldsymbol{u}^l[t] - \vartheta)$ in Equation (3) during training and retain $\Theta(\boldsymbol{u}^l[t] - \vartheta)$ for validation. In practice, minor alterations, as shown in Algorithm 1, are required in comparison to the surrogate gradient learning. Notably, the hard reset described in Equation (1) transforms into a soft variant with a certain probability when the analog activations $H_\alpha(x)$ are propagated, despite the fact that the equations describing neuron dynamics remain unchanged. Assuming $H_\alpha(x)$ models the function mapping from the expected membrane potential to spike probability (rate) within a short timeframe (corresponding to one time step). The neurons have a $(1 - H_\alpha(\boldsymbol{u}[t]))$ chance of not emitting a spike

Algorithm 1 Core function in ASGL

```
1: 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
2: Ensure: $\alpha$ is the learnable width parameter
3: if $T$ is true then
4:     generate mask $\boldsymbol{m}$ with noise probability $p$
5:     $\boldsymbol{s}^l[t] = H_\alpha(\hat{\boldsymbol{u}}^l[t]) + \boldsymbol{m} \odot \Phi\left(\Theta(\hat{\boldsymbol{u}}^l[t]) - H_\alpha(\hat{\boldsymbol{u}}^l[t])\right)$
       {The only line needed to update compared to the surrogate learning}
6: else
7:     $\boldsymbol{s}^l[t] = \Theta(\hat{\boldsymbol{u}}^l[t])$
8: end if
9: return $\boldsymbol{s}^l[t]$
```

and maintaining the previous state of membrane potential. In terms 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. Consequently, the information of membrane potential that exceeds the firing threshold is retained according to a certain probability. The temporal interpolating behavior in *soft reset* could be smoothed further by designing the time-invariant masks and adaptively adjusting α.

The only remaining challenge is to guarantee the noisy network trained with $\hat{H}_\alpha(\boldsymbol{x})$ could be evolved into an SNN with $\Theta(x)$ as activation finally. Fortunately, with the perspective of mixture feedforward, it can be accomplished by simply setting the width α as a learnable parameter and incorporating it into the spatial-temporal gradient backpropagation. Detailed analysis can be found in Section 4.3. The gradient calculation concerning width α could be expressed as follows:

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

$$\frac{\partial \boldsymbol{c}^L[t^*]}{\partial \boldsymbol{\alpha}^l} = \sum_{t=1}^{t^*} \boldsymbol{\delta}^{l+1}[t] \boldsymbol{W}^{l+1} \frac{\partial \boldsymbol{s}^l[t]}{\partial \boldsymbol{\alpha}^l[t]} - \sum_{t=1}^{t^*} \gamma \boldsymbol{\delta}^l[t+1] \mathrm{diag}(\boldsymbol{u}^l[t]) \frac{\partial \boldsymbol{s}^l[t]}{\partial \boldsymbol{\alpha}^l[t]}$$

$$\frac{\partial \boldsymbol{s}^l[t]}{\partial \boldsymbol{\alpha}^l[t]} = \frac{\partial H_{\boldsymbol{\alpha}}(\hat{\boldsymbol{u}}^l[t])}{\partial \boldsymbol{\alpha}} = \begin{cases} 0, & \text{if } \left|\hat{\boldsymbol{u}}^l[t]\right| > \frac{1}{2}\boldsymbol{\alpha} \\ -\mathrm{diag}(\frac{1}{\boldsymbol{\alpha}^2} \odot \hat{\boldsymbol{u}}^l[t]), & \text{otherwise} \end{cases} \tag{11}$$

In this way, the gradient mismatching will gradually diminish when adaptive α decreases 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). Moreover, this idea provides valuable insights into surrogate gradient learning. For instance, if we use full spike ($p = 1$) for forwarding computation, ASGL will transform into surrogate gradient while still allowing the adaptive width learning. Furthermore, ASGL naturally benefits from the pretrained ANNs by increasing gradually p from 0. Therefore, both ANN-SNN conversion and surrogate gradient learning could be implemented within the framework of ASGL, with different settings of noise probability p.

4.3. Theoretical Analysis

In this section, we demonstrate how the dynamics of noisy networks with learnable α can evolve into those of embedded SNNs. Let $\boldsymbol{F}_{\text{noise}}$ represent the noisy network used in training with $H_\alpha(x)$ across all layers, while $\boldsymbol{F}_{\text{snn}}$ denotes the target SNN embedded within $\boldsymbol{F}_{\text{noise}}$ using fully spike-based calculation. To estimate the real loss $\boldsymbol{\ell}_{\text{noise}}(\boldsymbol{F}, \boldsymbol{s})$ of the hybrid network $\boldsymbol{F}_{\text{noise}}$, we take the expectation over the mask matrix into account. The expression can be formulated as:

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

Here, $\hat{\boldsymbol{m}} = \boldsymbol{m}/p$ represents the normalized mask gathered from all layers, and $\boldsymbol{s}$ denotes the input spike pattern. By employing perturbation analysis (refer to Appendix A.2), we have the following proposition:

Approximation 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 $\boldsymbol{C}^l = D^2\left(\boldsymbol{\ell} \circ \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{G}^l]\right)[\boldsymbol{s}^l]$ represents the second derivative of loss function $\boldsymbol{\ell}$ with respect to the l-th layer spike activation $\boldsymbol{s}^l$ through 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 the l-th layer, with full spikes being adopted in the preceding l layers. To provide an intuitive explanation for the approximation, we analyze the non-trivial case when $p \neq 1$ from the perspective of iterative alternate optimization, involving two steps: (1) fixing weights $\boldsymbol{W}$, optimizing width α, and (2) fixing width α, optimizing weights $\boldsymbol{W}$. In the first steps, as $\boldsymbol{\ell}_{\text{snn}}(\boldsymbol{F}, \boldsymbol{s})$ remains constant with fixed weights, the width α tends to minimize the distance between $H_\alpha(\hat{\boldsymbol{u}}^l)$ and $\Theta(\hat{\boldsymbol{u}}^l)$. Consequently, the penalty term diminishes, and $\boldsymbol{\ell}_{\text{noise}}(\boldsymbol{F}, \boldsymbol{s})$ approaches $\boldsymbol{\ell}_{\text{snn}}(\boldsymbol{F}, \boldsymbol{s})$ during this step. In the second step, the noisy network, subject to global task-related loss $\boldsymbol{\ell}_{\text{noise}}(\boldsymbol{F}, \boldsymbol{s})$, is optimized under a specified smooth degree. Through iterative and alternating execution of these

Table 1. Performance Comparison 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., 2021b)	Surrogate Gradient	ResNet-18	10	75.45
TET (Deng et al., 2022)*	Surrogate Gradient	VGGSNN	10	83.17 ± 0.15
Ours	**ASGL**	VGGSNN	10	$\mathbf{84.50 \pm 0.08}$

Table 2. Ablation Study on Image Classification.

Width (α)	CIFAR-10		CIFAR-100	
	SG	ASGL	SG	ASGL
0.5	93.19	**94.11**	75.76	**76.54**
1.0	93.78	**94.30**	65.19	**76.09**
2.5	90.68	**94.09**	15.12	**76.18**
5.0	62.34	**93.61**	8.04	**76.68**
10.0	30.85	**93.53**	6.14	**76.00**

Table 3. Ablation Study with Different Noisy Probabilities p

Noisy Prob.	Width $\alpha = 2$		Width $\alpha = 1$		Width $\alpha = 0.5$	
	Fixed	Adapt.	Fixed	Adapt.	Fixed	Adapt.
$p = 1.0$	91.75	**92.74**	92.80	**92.94**	91.04	**91.68**
$p = 0.8$	**91.97**	**93.11**	**93.08**	**93.32**	**91.12**	**92.26**

Table 4. Classification Performance on SHD dataset

Method	Type	#Param.	Acc. (%)
(Yin et al., 2020)	SG	14.13w	84.4
(Cramer et al., 2020)	SG	178.59w	83.2 ± 1.3
(Perez-Nieves et al., 2021)	SG	**10.85w**	82.7 ± 0.8
(Zenke & Vogels, 2021)	SG	24.99w	82.0 ± 2.0
(Perez-Nieves & Goodman, 2021)	Sparse SG	28.80w	77.5
(Nowotny et al., 2022)	EventProp	24.99w	84.8 ± 1.5
Ours	**ASGL**	23.04w	$\mathbf{86.9} \pm 1.0$

two steps, a high-performance SNN can be achieved by training a noisy network, even without explicitly increasing p during training. The theoretical findings are further supported by the empirical results presented in Figure 3b, where training with fixed p is demonstrated. Notably, both trainable width and random noise injection with spikes play crucial roles in ensuring the validity of the first step. The spike noise could be transformed into the penalty on layerwise activations, while the learnable α facilitates local optimization by enforcing $H_\alpha(\boldsymbol{x})$ to align with $\Theta(\boldsymbol{x})$.

5. Experiments

To validate the effectiveness of the proposed method, we perform experiments on various types of data, including static images and spatio-temporal patterns such as dynamic event streams, spoken digital speech, and instrumental music. Additionally, we investigate the evolutions of network dynamics and the impact of noise probability to understand how injecting noise with spikes provides real observation about the loss landscape of target SNNs. More specific implementation details and energy estimation could be found in Appendix B.1 and Appendix B.2, respectively.

5.1. Ablation Study

In Table 2, we compare ASGL with *Surrogate Gradient* (SG) on both CIFAR datasets under timestep $N = 3$ using ResNet-19 (Zheng et al., 2021) and ResNet-18 architecture for the ablation study, respectively. For a fair comparison, the same optimizer setting, seed, and weight initialization are employed. In particular, we use the SGD optimizer with the weight decay of 5e-4 over 100 epochs (CIFAR-10) and 300 epochs (CIFAR-100). For CIFAR-100, in addition to the shifted clipping function depicted in Equation (8), we use $h_\alpha(x) = 1/(1 + e^{-\alpha x})$ as a surrogate forwarding function to show ASGL could be also applied to other alternative functions. The results demonstrate that ASGL outperforms SG across a range of width initialization on both datasets. It is catastrophic damage for SG when width α is selected inappropriately. For instance, even a small adjustment of 0.5 to the width in SG can lead to a significant drop in performance of nearly 10% on CIFAR-100 (from 75.76% to 65.19%). In contrast, ASGL exhibits surprising robustness for different width α as well as performance improvements.

To further validate the effectiveness of adaptive α, we conduct the four sets of experiments on the CIFAR-10 dataset using spiking ResNet-19 under 2 time steps. The experiments involve different settings of p and α. The results in Table 3 reveal that the adaptive α introduced by ASGL leads to notable performance improvement compared to the fixed α setting in both cases ($p = 1$ and $p = 0.8$), across all the well-selected initial width settings in SG. In addition, dual-mode switching leads to a certain performance improvement when comparing the results of $p = 0.8$ and $p = 1$.

5.2. Performance on Static Images

In Table 5, we compare our method with state-of-the-art approaches on the CIFAR and Tiny ImageNet datasets. For

Table 5. Classification Performance on Static Image Benchmarks.

Dataset	Method	Type	Architecture	Time steps	Accuracy(%)
CIFAR-10	Opt. (Deng & Gu, 2021)	Conversion		400-600	90.61
	TSSL-BP (Zhang & Li, 2020)	Time-based Gradient	CifarNet	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
	IM-Loss (Guo et al., 2022)	Surrogate Gradient	CifarNet	4	92.20 ± 0.12
	Ours	**ASGL**	**CifarNet**	**4**	**94.74 ± 0.10**
				2	**93.80 ± 0.11**
	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
	SpikeDHS (Leng et al., 2022)	Surrogate Gradient	SpikeDHS-CLA (n4s1)	6	94.68 ± 0.05
	Dspike (Li et al., 2021b)	Surrogate Gradient	ResNet-18	4	93.66 ± 0.05
				2	93.13 ± 0.07
	GLIF (Yao et al., 2022)	Surrogate Gradient	ResNet-18	4	94.67 ± 0.05
				2	94.15 ± 0.04
	Ours	**ASGL**	**ResNet-18**	**4**	**95.35** ± 0.25
				2	**95.27** ± 0.06
CIFAR-100	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., 2021b)	Surrogate Gradient	ResNet-18	4	73.35 ± 0.14
				2	71.68 ± 0.12
	IM-Loss (Guo et al., 2022)	Surrogate Gradient	VGG-16	5	70.18 ± 0.09
	SpikeDHS (Leng et al., 2022)	Surrogate Gradient	SpikeDHS-CLA (n4s1)	6	76.03 ± 0.20
	Ours	**ASGL**	**CifarNet**	**4**	**74.59 ± 0.07**
				2	**74.31** ± 0.15
			ResNet-18	**4**	**77.74 ± 0.07**
				2	**76.59 ± 0.05**
Tiny-ImageNet	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., 2021a)†	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**	**8**	**56.81**
				4	**56.57**

* The results are reproduced through the publicly available code.

CIFAR datasets, we use the widely-used CifarNet (Wu et al., 2019) and a modified ResNet-18 structure (Li et al., 2021b). As done in (Li et al., 2021b; 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., 2021b; Rathi et al., 2020; Rathi & Roy, 2020) to initialize weights and Time Inheritance Training (TIT) (Li et al., 2021b; Deng et al., 2022) to improve performance under low time steps. Nonetheless, ASGL outperforms the state-of-the-art surrogate gradient and conversion methods with the same or fewer time steps on both datasets. Moreover, 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. For Tiny-ImageNet, 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 5, ASGL still achieves competitive results compared to other methods using only 4 time steps which further verifies the effectiveness of ASGL.

5.3. Performance on Spatio-temporal Patterns.

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 B.4.

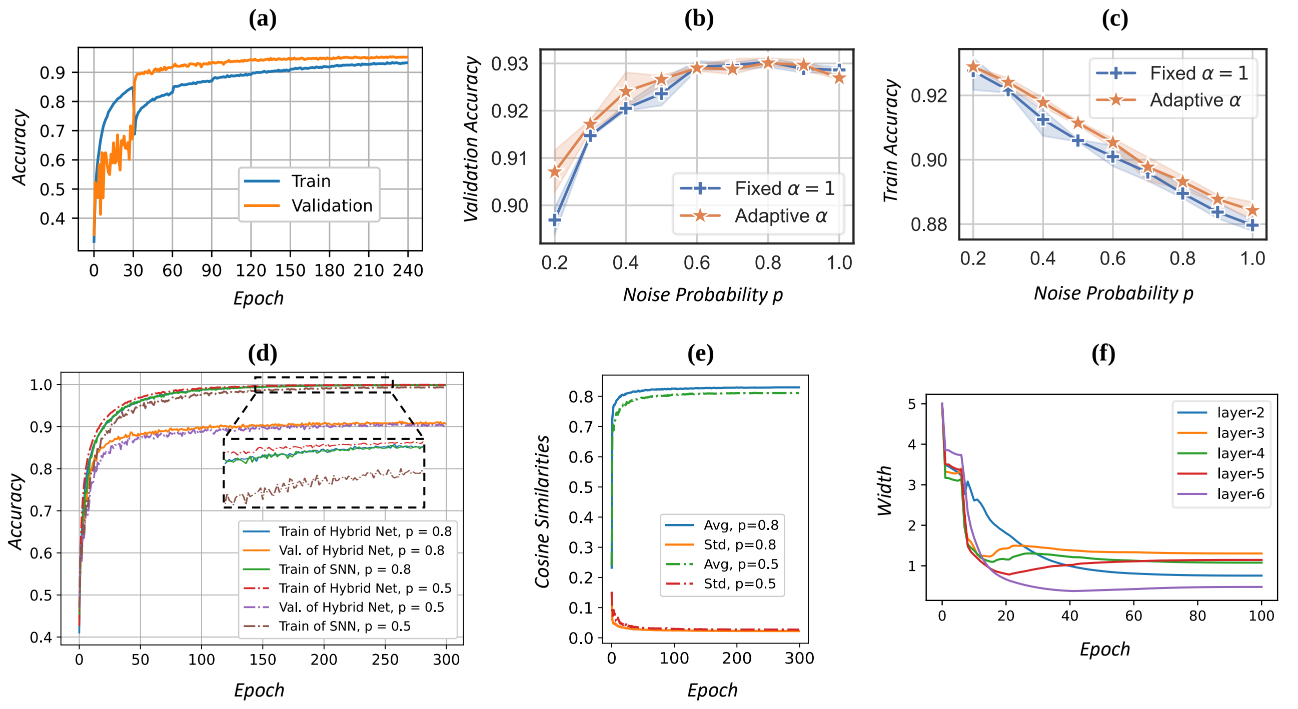

Figure 3. (**a**) explores the evolution procedure by observing accuracy change with increasing spike noise. (**b**) and (**c**) examine the accuracy for different fixed p selections during the test and training, respectively. (**d**) and (**e**) shows the similarities between the noisy network and embedded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

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 the VGGSNN architecture and data augmentation method in (Deng et al., 2022). As shown in Table 1, our method achieves state-of-the-art performance (84.50% $\pm$ 0.08%) without a larger network (e.g., ResNet-19, DenseNet), which is competitive to 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 (700-240-20) with recurrent synaptic connections to identify the keywords in utterances (More details about recurrent connections could be found in Appendix A.4). As shown in Table 4, the proposed method achieves 2% accuracy improvement at least without any data augmentation introduced in (Cramer et al., 2020). Notably, 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 spiking neurons, respectively. The number of parameters listed in Table 4 is the minimum number of parameters inferred from the corresponding network structure described in the original paper.

5.4. Effect of Noise Probability

In this section, we aim to analyze how noise probability affects the performance of SNNs. Firstly, we conduct experiments by increasing noise probability p from 0 to 0.8 in increments of 0.1 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 exhibits erratic growth in the first 30 epochs. This behavior is expected since there is no noise injection ($p = 0$) initially, and the network operates purely in the analog domain without any spike injections. However, when 10% spikes are injected ($p = 0.1$), the validation accuracy of the embedded SNN starts to increase rapidly around the 30th epoch at the cost of a drop in training accuracy for the noisy network. This indicates that the noisy network begins to transform into the target SNNs. Interestingly, for the noise injection at the 60-th and 90-th epoch, the training accuracy is actually improved, suggesting that a small number of spikes in the first injection is sufficient for the dynamics of SNNs and the presence of analog activations may hinder fast convergence instead.

Next, we investigate the effect of different choices of fixed p in Figure 3**b** and Figure 3**c**. Generally, selecting a low value of p blocks the evolution toward SNNs due to high analog activations. On the other hand, choosing an excessively high value of p does not achieve the best generalization performance. There exists an optimal range of p that balances the analog and spiking dynamics, leading to better generalization.

5.5. Network Evolution

To understand the evolution of the noisy network, we visualize accuracy changes of the noisy network and the embedded SNN with shared weights during training under two noise probability settings $p = 0.8$ 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, indicating that the noisy network consistently predicts outputs similar to 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 across all layers after each training epoch with ASGL. Figure 3(**e**) reports the mean and standard deviation of S across all samples in the CIFAR-10 training set. According to the results, even with shared weights, the hybrid network initially has relatively low overall similarities, but as training progresses with ASGL, the hybrid network gradually shifts toward an SNN, resulting in similarities reaching approximately 0.8. The decreasing standard deviation also validates the effectiveness of ASGL. Additionally, we evaluate the evolution of the noisy network by monitoring the change of learnable width α (Figure 3**f**) in the image reconstruction task. The width αs consistently decline throughout training and converge to specific values across all layers, indicating a convergence towards the desired SNN behavior.

6. Conclusion

In this paper, we introduce ASGL, a novel training method to develop deep SNNs. Different from the typical surrogate gradient learning, our method circumvents the gradient mismatching problem naturally and updates weights adaptively against 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 gradient calculation. This allows the hybrid network to learn the optimal shapes of the activation functions against the spike noise and evolve into SNNs. To validate the effectiveness and generalization of the proposed method, we study theoretically and practically the evolution from hybrid networks to SNNs further. 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 selections of smooth factors.

Acknowledgements

This work is supported by the National Key Research and Development Program of China under Grant 2020AAA0105900 and the National Natural Science Foundation of China under Grant 62236007, Grant U2030204. We sincerely thank the anonymous reviewers and AC for their valuable suggestions on this work.

References

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G.-J., 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.

Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., et al. A low power, fully event-based gesture recognition system. In *CVPR*, pp. 7243–7252, 2017.

Bi, Y., Chadha, A., Abbas, A., Bourtsoulatze, E., and Andreopoulos, Y. Graph-based spatio-temporal feature learning for neuromorphic vision sensing. *IEEE Transactions on Image Processing*, 29:9084–9098, 2020.

Bittner, R. M., Salamon, J., Tierney, M., Mauch, M., Cannam, C., and Bello, J. P. Medleydb: A multitrack dataset for annotation-intensive mir research. In *ISMIR*, volume 14, pp. 155–160, 2014.

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

Bu, T., Fang, W., Ding, J., Dai, P., Yu, Z., and Huang, T. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2021.

Cramer, B., Stradmann, Y., Schemmel, J., and Zenke, F. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

Deng, S. and Gu, S. Optimal conversion of conventional artificial neural networks to spiking neural networks. *ICLR*, 2021.

Deng, S., Li, Y., Zhang, S., and Gu, S. Temporal efficient training of spiking neural network via gradient reweighting. *arXiv preprint arXiv:2202.11946*, 2022.

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

Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., and Tian, Y. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 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. In *ICCV*, pp. 2661–2671, 2021b.

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

Gu, P., Xiao, R., Pan, G., and Tang, H. Stca: Spatio-temporal credit assignment with delayed feedback in deep spiking neural networks. In *IJCAI*, pp. 1366–1372, 2019.

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

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

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

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015.

He, W., Wu, Y., Deng, L., Li, G., Wang, H., Tian, Y., Ding, W., Wang, W., and Xie, Y. Comparing snns and rnns on neuromorphic vision datasets: similarities and differences. *Neural Networks*, 132:108–120, 2020.

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

Kim, J., Kim, K., and Kim, J.-J. Unifying activation-and timing-based learning rules for spiking neural networks. *NeurIPS*, 33:19534–19544, 2020.

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, 2020.

Kundu, S., Datta, G., Pedram, M., and Beerel, P. A. 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.

Leng, L., Che, K., Zhang, K., Zhang, J., Meng, Q., Cheng, J., Guo, Q., and Liao, J. Differentiable hierarchical and surrogate gradient search for spiking neural networks. In *Advances in Neural Information Processing Systems*, 2022.

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

Li, H., Liu, H., Ji, X., Li, G., and Shi, L. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017.

Li, Y., Deng, S., Dong, X., Gong, R., and Gu, S. A free lunch from ANN: towards efficient, accurate spiking neural networks calibration. In *ICML*, volume 139, pp. 6316–6325, 2021a.

Li, Y., Guo, Y., Zhang, S., Deng, S., Hai, Y., and Gu, S. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *NeurIPS*, 34, 2021b.

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

Nagel, M., Amjad, R. A., Van Baalen, M., Louizos, C., and Blankevoort, T. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pp. 7197–7206. PMLR, 2020.

Neftci, E. O., Mostafa, H., and Zenke, F. 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.

Nowotny, T., Turner, J. P., and Knight, J. C. Loss shaping enhances exact gradient learning with eventprop in spiking neural networks. *arXiv preprint arXiv:2212.01232*, 2022.

Pei, J., Deng, L., Song, S., Zhao, M., Zhang, Y., Wu, S., Wang, G., Zou, Z., Wu, Z., He, W., et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

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

Perez-Nieves, N., Leung, V. C., Dragotti, P. L., and Goodman, D. F. Neural heterogeneity promotes robust learning. *Nature communications*, 12(1):1–9, 2021.

Pons, J., Slizovskaia, O., Gong, R., Gómez, E., and Serra, X. Timbre analysis of music audio signals with convolutional neural networks. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 2744–2748. IEEE, 2017.

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.

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

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

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

Severa, W., Vineyard, C. M., Dellana, R., Verzi, S. J., and Aimone, J. B. Training deep neural networks for binary communication with the whetstone method. *Nature Machine Intelligence*, 1(2):86–94, 2019.

Shrestha, S. B. and Orchard, G. Slayer: Spike layer error reassignment in time. *arXiv preprint arXiv:1810.08646*, 2018.

Wang, Q., Zhang, Y., Yuan, J., and Lu, Y. 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.

Wang, Z., Lian, S., Zhang, Y., Cui, X., Yan, R., and Tang, H. Towards lossless ann-snn conversion under ultra-low latency with dual-phase optimization. *arXiv preprint arXiv:2205.07473*, 2022.

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

Wu, J., Chua, Y., Zhang, M., Li, G., Li, H., and Tan, K. C. 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.

Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12: 331, 2018.

Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., and Shi, L. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1311–1318, 2019.

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*, 2021b.

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

Yang, Q., Wu, J., Zhang, M., Chua, Y., Wang, X., and Li, H. Training spiking neural networks with local tandem learning. *NeurIPS*, 2022.

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

Yao, M., Gao, H., Zhao, G., Wang, D., Lin, Y., Yang, Z., and Li, G. Temporal-wise attention spiking neural networks for event streams classification. In *ICCV*, pp. 10221–10230, 2021.

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.

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

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

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

Zheng, H., Wu, Y., Deng, L., Hu, Y., and Li, G. Going deeper with directly-trained larger spiking neural networks. In *AAAI 2021*, pp. 11062–11070, 2021.

A. Detailed Derivation and Neuron Dynamics

A.1. Spike-based Backpropagation incorporated with Adaptive Width

Firstly, we decompose the gradients of aggregate loss over the entire time window into gradients corresponding to different target time steps t^*:

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

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

$$\frac{\partial \boldsymbol{c}^L[t^*]}{\partial \boldsymbol{W}^l} = \sum_{t=1}^{t^*} \frac{\partial \boldsymbol{c}^L[t^*]}{\partial \boldsymbol{u}^l[t]} \frac{\partial \boldsymbol{u}^l[t]}{\partial \boldsymbol{c}^l[t]} \frac{\partial \boldsymbol{c}^l[t]}{\partial \boldsymbol{W}^l} = \sum_{t=1}^{t^*} \frac{\partial \boldsymbol{c}^L[t^*]}{\partial \boldsymbol{u}^l[t]} \frac{\partial \boldsymbol{c}^l[t]}{\partial \boldsymbol{W}^l} \tag{15}$$

Here, $\frac{\partial \boldsymbol{c}^l[t]}{\partial \boldsymbol{W}}$ is a three-dimensional tensor scattered with the afferent spikes $\boldsymbol{s}^{l-1}[t]$. For simplicity, we denote the credit $\frac{\partial \boldsymbol{c}^L[t^*]}{\partial \boldsymbol{u}^l[t]}$ assigned to $\boldsymbol{u}^l[t]$ as $\boldsymbol{\delta}^l[t]$, which can be calculated as follows when $t < t^*$ and $l < L-1$:

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

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

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

Furthermore, by replacing $\Theta(x)$ with $\hat{H}_\alpha(x)$ and incorporating the smooth factor α into the computational flow of SNNs as shown in Equation (10), ASGL can compute the gradient with respect to α:

$$\begin{aligned}
\nabla \boldsymbol{\alpha}^l &= \frac{\partial \mathcal{L}(\overline{\boldsymbol{c}}^L, \boldsymbol{y})}{\partial \overline{\boldsymbol{c}}^L} \frac{\partial \overline{\boldsymbol{c}}^L}{\partial \boldsymbol{\alpha}^l} = -\frac{1}{N}(\boldsymbol{y}^T - \hat{\boldsymbol{y}}^T) \sum_{t^*=1}^{N} \frac{\partial \boldsymbol{c}^L[t^*]}{\partial \boldsymbol{\alpha}^l} \\
\frac{\partial \boldsymbol{c}^L[t^*]}{\partial \boldsymbol{\alpha}^l} &= \sum_{t=1}^{t^*} \left(\frac{\partial \boldsymbol{c}^L[t^*]}{\partial \boldsymbol{c}^{l+1}[t]} \frac{\partial \boldsymbol{c}^{l+1}[t]}{\partial \boldsymbol{s}^l[t]} + \frac{\partial \boldsymbol{c}^L[t^*]}{\partial \boldsymbol{u}^l[t+1]} \frac{\partial \boldsymbol{u}^l[t+1]}{\partial \boldsymbol{s}^l[t]} \right) \frac{\partial \boldsymbol{s}^l[t]}{\partial \boldsymbol{\alpha}^l[t]} \\
&= \sum_{t=1}^{t^*} \left(\frac{\partial \boldsymbol{c}^L[t^*]}{\partial \boldsymbol{u}^{l+1}[t]} \frac{\partial \boldsymbol{u}^{l+1}[t]}{\partial \boldsymbol{c}^{l+1}[t]} \frac{\partial \boldsymbol{c}^{l+1}[t]}{\partial \boldsymbol{s}^l[t]} + \frac{\partial \boldsymbol{c}^L[t^*]}{\partial \boldsymbol{u}^l[t+1]} \frac{\partial \boldsymbol{u}^l[t+1]}{\partial \boldsymbol{s}^l[t]} \right) \frac{\partial \boldsymbol{s}^l[t]}{\partial \boldsymbol{\alpha}^l[t]} \\
&= \sum_{t=1}^{t^*} \left(\boldsymbol{\delta}^{l+1}[t] \boldsymbol{W}^{l+1} - \gamma \boldsymbol{\delta}^l[t+1] \operatorname{diag}(\boldsymbol{u}^l[t]) \right) \frac{\partial \boldsymbol{s}^l[t]}{\partial \boldsymbol{\alpha}^l[t]}
\end{aligned} \tag{18}$$

where $\frac{\partial \boldsymbol{s}^l[t]}{\partial \boldsymbol{\alpha}^l[t]}$ relies on the selection of surrogate forwarding function $H_\alpha(x)$. For the specific case of the surrogate clipping function, it could be obtained as follows:

$$\frac{\partial \boldsymbol{s}^l[t]}{\partial \boldsymbol{\alpha}^l[t]} = \frac{\partial H_{\boldsymbol{\alpha}}(\hat{\boldsymbol{u}}^l[t])}{\partial \boldsymbol{\alpha}} = \begin{cases} 0, & \text{if } \left|\hat{\boldsymbol{u}}^l[t]\right| > \frac{1}{2}\boldsymbol{\alpha} \\ -\operatorname{diag}(\frac{1}{\boldsymbol{\alpha}^2} \odot \hat{\boldsymbol{u}}^l[t]), & \text{otherwise} \end{cases} \tag{19}$$

This derivative captures how changes in α affect the spiking activity $\boldsymbol{s}^l[t]$ when using the surrogate clipping function as the surrogate forwarding function. Subsequently, the complete backward propagation through time of SNNs utilizing the LIF model can be derived from Equations (14) to (19).

A.2. Theory Analysis for Error Decomposition

Table 6. 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 $\Theta(\hat{\boldsymbol{u}}^l)$
$\boldsymbol{F}^l$	$\boldsymbol{f}^l \circ \boldsymbol{f}^{l-1} \circ ... \circ \boldsymbol{f}^1$
$\boldsymbol{s}^l$	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

Approximation 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 $H_\alpha(\hat{\boldsymbol{u}}^l)$.*

$$\boldsymbol{\ell}_{noise}(\boldsymbol{F}, \boldsymbol{s}) \approx \boldsymbol{\ell}_{snn}(\boldsymbol{F}, \boldsymbol{s}) + \frac{1-p}{2p} \sum_{l=1}^{L} \left\langle \boldsymbol{C}^l, diag(H_\alpha(\hat{\boldsymbol{u}}^l) - \Theta(\hat{\boldsymbol{u}}^l))^{\odot 2} \right\rangle \tag{20}$$

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 the l-th layer. Moreover, $\boldsymbol{s}^l$ is the output spikes at the l-th layer across all time steps and neurons. Detailed symbol definitions and descriptions are provided in Table 6. Then we can have:

$$\begin{aligned}
\boldsymbol{\ell}_{\text{noise}}(\boldsymbol{F}, \boldsymbol{s}) &= \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s})) + \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s}, \hat{\boldsymbol{m}})) - \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s}))] \\
&= \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s})) + \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{\ell}(\boldsymbol{G}^0(\boldsymbol{s}^0, \hat{\boldsymbol{m}})) - \boldsymbol{\ell}(\boldsymbol{G}^L(\boldsymbol{s}^L))] \\
&= \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s})) + \mathbb{E}_{\hat{\boldsymbol{m}}}\left[\sum_{l=1}^{L}\Big(\boldsymbol{\ell}(\boldsymbol{G}^{l-1}(\boldsymbol{s}^{l-1}, \hat{\boldsymbol{m}})) - \boldsymbol{\ell}(\boldsymbol{G}^l(\boldsymbol{s}^l, \hat{\boldsymbol{m}}))\Big)\right] \\
&= \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s})) + \sum_{l=1}^{L} \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{\ell}(\boldsymbol{G}^{l-1}(\boldsymbol{s}^{l-1}, \hat{\boldsymbol{m}})) - \boldsymbol{\ell}(\boldsymbol{G}^l(\boldsymbol{s}^l, \hat{\boldsymbol{m}}))] \\
&= \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s})) + \sum_{l=1}^{L} \boldsymbol{R}(\boldsymbol{G}^l, \boldsymbol{s}^l)
\end{aligned} \tag{21}$$

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

$$\begin{aligned}
\boldsymbol{G}^{l-1}\left(\boldsymbol{s}^{l-1}, \hat{\boldsymbol{m}}\right) &= \boldsymbol{G}^l\left((\boldsymbol{1} - \hat{\boldsymbol{m}}^l) \odot H_\alpha(\hat{\boldsymbol{u}}^l) + \hat{\boldsymbol{m}}^l \odot \Theta(\hat{\boldsymbol{u}}^l), \hat{\boldsymbol{m}}\right) \\
&= \boldsymbol{G}^l\left((\boldsymbol{1} - \hat{\boldsymbol{m}}^l) \odot (H_\alpha(\hat{\boldsymbol{u}}^l) - \Theta(\hat{\boldsymbol{u}}^l)) + \boldsymbol{s}^l, \hat{\boldsymbol{m}}\right)
\end{aligned} \tag{22}$$

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

$$\begin{aligned}
\boldsymbol{R}(\boldsymbol{G}^l, \boldsymbol{s}^l) &= \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{\ell}(\boldsymbol{G}^{l-1}(\boldsymbol{s}^{l-1}, \hat{\boldsymbol{m}})) - \boldsymbol{\ell}(\boldsymbol{G}^l(\boldsymbol{s}^l, \hat{\boldsymbol{m}}))] \\
&= \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{\ell}(\boldsymbol{G}^l(\boldsymbol{s}^l + \Delta, \hat{\boldsymbol{m}})) - \boldsymbol{\ell}(\boldsymbol{G}^l(\boldsymbol{s}^l, \hat{\boldsymbol{m}}))] \\
&\approx \mathbb{E}_{\hat{\boldsymbol{m}}}\left[D(\boldsymbol{\ell} \circ \boldsymbol{G}^l)[\boldsymbol{s}^l]\Delta + \frac{1}{2}\Delta^T\left(D^2(\boldsymbol{\ell} \circ \boldsymbol{G}^l)[\boldsymbol{s}^l]\right)\Delta\right]
\end{aligned} \tag{23}$$

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

$$\boldsymbol{R}(\boldsymbol{G}^l, \boldsymbol{s}^l) \approx \mathbb{E}_{\hat{\boldsymbol{m}}}\left[\frac{1}{2}\Delta^T\left(D^2(\boldsymbol{\ell} \circ \boldsymbol{G}^l)[\boldsymbol{s}^l]\right)\Delta\right] \tag{24}$$

Then we could take the expectation over Δ:

$$\begin{aligned}\boldsymbol{R}\left(\boldsymbol{G}^l, \boldsymbol{s}^l\right) &\approx \frac{1}{2}\left\langle D^2\left(\boldsymbol{\ell} \circ \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{G}^l]\right)\left[\boldsymbol{s}^l\right], \mathbb{E}_{\Delta}\left[\Delta\Delta^T\right]\right\rangle \\ &= \frac{1-p}{2p}\left\langle D^2\left(\boldsymbol{\ell} \circ \mathbb{E}_{\hat{\boldsymbol{m}}}[\boldsymbol{G}^l]\right)\left[\boldsymbol{s}^l\right], \text{diag}\left(\left(H_\alpha(\hat{\boldsymbol{u}}^l) - \Theta(\hat{\boldsymbol{u}}^l)\right)^{\odot 2}\right)\right\rangle\end{aligned} \tag{25}$$

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 $\boldsymbol{C}^l$(Nagel et al., 2020). Furthermore, by substituting Equation (25) into Equation (21), we get:

$$\boldsymbol{\ell}_{\text{noise}}(\boldsymbol{F}, \boldsymbol{s}) \approx \boldsymbol{\ell}(\boldsymbol{F}_{\text{snn}}(\boldsymbol{s})) + \frac{1-p}{2p}\sum_{l=1}^{L}\left\langle \boldsymbol{C}^l, \text{diag}(H_\alpha(\hat{\boldsymbol{u}}^l) - \Theta(\hat{\boldsymbol{u}}^l))^{\odot 2}\right\rangle \tag{26}$$

A.3. 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 can be represented as:

$$\begin{aligned}\boldsymbol{s}^l[t] &= \Theta\left(\boldsymbol{u}^l[t] - \vartheta\right) \\ \boldsymbol{c}^l[t] &= \boldsymbol{u}_0 \odot \boldsymbol{W}^l\boldsymbol{s}^{l-1}[t] \\ \boldsymbol{u}^l[t] &= \boldsymbol{m}^l[t] - \boldsymbol{v}^l[t] - \boldsymbol{e}^l[t] \\ \boldsymbol{v}^l[t] &= \beta_v\boldsymbol{v}^l[t-1] + \boldsymbol{c}^l[t] \\ \boldsymbol{m}^l[t] &= \beta_m\boldsymbol{m}^l[t-1] + \boldsymbol{c}^l[t] \\ \boldsymbol{e}^l[t] &= \beta_m\boldsymbol{m}^l[t-1] + \vartheta\boldsymbol{s}^l[t-1] \\ \Theta(x) &= \begin{cases} 1, & x \geq \vartheta \\ 0, & x < \vartheta \end{cases}\end{aligned} \tag{27}$$

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. The variable $\boldsymbol{e}^l[t]$ simulates the refractory period in spiking neurons. The decay factors $\beta_{v(m)} = \exp(-\frac{\Delta t}{\tau_{v(m)}})$, where $\beta_{v(m)} < 1$, represents the rate at which the input response and output response decay over time. If no spike occurs, the input and output responses will gradually decrease to 0 by recursively multiplying the decay factors. All other symbols keep consistent with the standard LIF model.

A.4. Reccurent Connections

In order to enhance the memory capacity of SNNs in the temporal domain, synaptic recurrence is widely adopted distinguish from the internal dynamics with decay mechanism in spiking neurons. The fundamental equation for this external recurrence can be described as follows:

$$\frac{d\boldsymbol{c}^l}{dt} = \underbrace{-\frac{1}{\tau_{\text{syn}}}\boldsymbol{c}^l(t)}_{\text{exp. decay}} + \underbrace{\boldsymbol{W}^l\boldsymbol{s}^{l-1}(t)}_{\text{feedforward}} + \underbrace{\boldsymbol{V}^l\boldsymbol{s}^l(t)}_{\text{recurrent}}, \tag{28}$$

where the terms of decay τ_{syn} and τ_m in Equation (1) both contribute to the internal recurrence, influencing the dynamics of the spike-based system. Furthermore, synaptic recurrence, characterized by the weight $\boldsymbol{V}^l$, plays a role in enhancing temporal memory capabilities.

Table 7. 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)

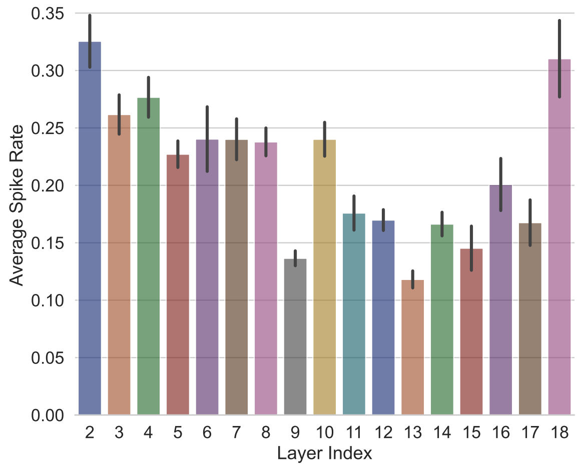

Figure 4. The average spike rate of each layer on the CIFAR-10 dataset .

B. Supplemental Experiments and Details

B.1. Experimental Details

We utilize the ADAM optimizer with an initial learning rate $\lambda = 0.001$ for the CIFAR100 dataset when adjusting p, while for the CIFAR10 dataset, we employ the SGD optimizer with an initial learning rate of $\lambda = 0.1$. To augment the data in both static image datasets, we use AutoAugment (Cubuk et al., 2018) and Cutout (DeVries & Taylor, 2017), as done in (Li et al., 2021a; Bu et al., 2021; Wang et al., 2022). Additionally, a cyclic cosine annealing learning rate scheduler is adopted. For the SHD dataset, we discretize the time into 250 time steps and initiate a decrease in noise probability starting from 0.2. The corresponding network architecture is $700 - 240 - 20$, with the middle layer neurons connected with recurrent synapses. For the MedlyDB dataset, we increase the noise probability at 30-th, 70-th, 90-th, 95-th epoch with a discretization of 500 time steps. Specifically, we update p as $1 - (1 - p) \cdot \zeta$ at each milestone and attenuate the ratio of analog activations at the rate of ζ. All the p and ζ values we use for each dataset are shown in Table 7 unless otherwise specified. For the CIFAR-10 results of Table 8, we provide detailed statistics and configurations in Table 12. For the CIFAR-100 results of Table 8, we set p and ζ as 0.8 and 1 for ASGL, respectively.

B.2. Energy Estimation

In this section, we visualize the spike rate of each layer in the spiking ResNet-18, as shown in Figure 4. Additionally, we follow the convention of the neuromorphic computing community by counting the total synaptic operations (SOP) to estimate the computation overhead of SNN models and compare it to the energy consumption of 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 spike sparsity. For SNNs, the total synaptic operation with accumulation N_{AC} is defined as:

$$N_{\text{AC}} = \sum_{t=1}^{T} \sum_{l=1}^{L-1} \sum_{i=1}^{N^l} f_i^l s_i^l[t] \tag{29}$$

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

$$N_{\text{MAC}} = \sum_{l=1}^{L-1} \sum_{i=1}^{N^l} f_i^l \tag{30}$$

Specially, we use MAC to estimate the energy cost of the first layer in SNNs as the direct current input without explicit encoding is adopted in our experiments on static images. Here, we randomly select 1024 samples 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 the CIFAR-10 dataset with only 8.96% energy consumption compared to the ANN with the same architecture.

B.3. Ablation Study on Image Reconstruction

Table 8. Comparison of 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**

In addition to evaluating the performance of ASGL on classification tasks, 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 8 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 shows higher performance.

B.4. Experiments on MedlyDB, DVS128 Gesture

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**

Performance on MedlyDB. In this experiment, we explore the instrument recognization task with various music pieces in different melodies and styles. Specifically, we follow the approach of (Gu et al., 2019) and subtract a subset of MedlyDB that 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, network structure (384-700-10), and C-LIF neuron as the previous work for a fair comparison. The results show that our method exhibits superior performance compared to other approaches across most metrics, with the exception of the Recall rate achieved by the specialized CNN developed by (Pons et al., 2017). This CNN utilizes specially designed convolutional kernels that are optimized for capturing temporal information in music.

Performance on DVS128 Gesture. The DVS128 Gesture dataset 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

Table 10. 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**

an average duration of 6.5 ± 1.7 s and all samples are split into a training set (1208 samples) and a test set (134 samples). Given the long sample duration and the limited sample size, we follow the RCS approach (Yao et al., 2021) and randomly select 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 (He et al., 2020). In Table 10, our method has achieved 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.

B.5. Ablation Study on ResNet-19

Table 11. Comparison with Surrogate Gradient Learning with VGG-19 architecture on the CIFAR-10 dataset.

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	**89.62 ± 0.20**	**89.69 ± 0.17**	**88.83 ± 0.09**

We have included the results of ResNet-19 (Zheng et al., 2021) under time-steps $N = 3$ in Table 12 (rectangular function) and Table 13 (Dspike function (Li et al., 2021b)). Specifically, we test the effect of different width initializations while using the same optimizer settings and weight initialization for fairness. Notably, ResNet-19 has approximately 10× operations than ResNet-18 (Li et al., 2021b). 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. From Table 13, we could find that the Dspike function shows certain robustness with respect to the rectangular function . However, it could still be improved greatly by ASGL when $\alpha \leq 0.5$ as shown in Table 13.

Table 12. Comparison between ASGL and *Surrogate Gradient* with rectangular functions under different width initializations.

Width	SG	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 13. Comparison between ASGL and *Surrogate Gradient* with Dspike functions under different width initializations.

Width	SG	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

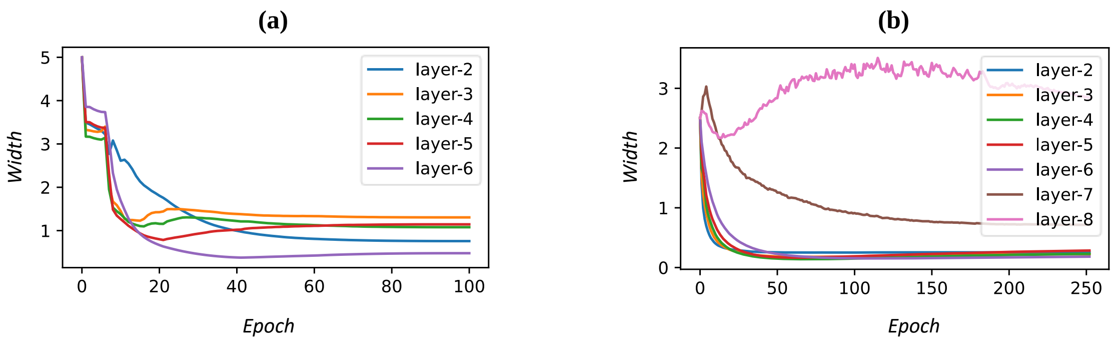

Figure 5. Fig. (**a**) and (**b**) visualize the width update in image reconstruction and image recognization, respectively.

B.6. 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 decreases progressively across all layers. This indicates the injection of spike noise forces the noisy network to evolve into the desired SNN and simultaneously optimize it in a coupled manner. Additionally, Figure 5**b** presents the variation of α in CIFAR-10 classification. Notably, the width α of the final layer in CIFARNet increases while others consistently decrease. This phenomenon likely stems from the coupled training, which aims to simultaneously minimize the loss of SNN and narrow the gap between the noisy network and SNN. Moreover, it suggests that the dynamically adjusting the width is non-trivial and essential for different network layers and training epochs.

B.7. Effect of fixed α

In this section, we delve into the impact of a fixed α coefficient on the tradeoff between minimizing the propotion of dead neurons and minimizing smoothing error. To investigate this , we utilize a spiking VGG16 architecture on the CIFAR10 dataset, considering 3 discrete time steps. Each sample in the CIFAR-10 training set is treated as a trial, allowing us to ascertain the frequency of neuron death across 50,000 trials, thus approximating the probability of neuron inactivity. Figure 6**a** illustrates the average proportion of inactive neurons at each layer. Generally, as the network width increases, the ratio of inactive neurons gradually diminishes. Furthermore, due to the inability to compute the true gradient of the binary spikes, we estimate the degree of gradient mismatching by evaluating the activation discrepancy between the SNN with the Heaviside function $\Theta(x)$ and the ANN with the surrogate feedforward function $\Theta_\alpha(x)$. In this regard, we employ the metric of $err = 1 - \cos(\boldsymbol{a}^l, \boldsymbol{s}^l)$ to quantitatively assess the level of activation mismatch at each layer. As depicted in Figure 6**b**, with increasing width, the smoothing error at each layer grows larger, demonstrating an opposing trend to the proportion of inactive neurons. Taking into account both scenarios, it becomes crucial to develop a suitable adaptive approach for the smooth factor α.

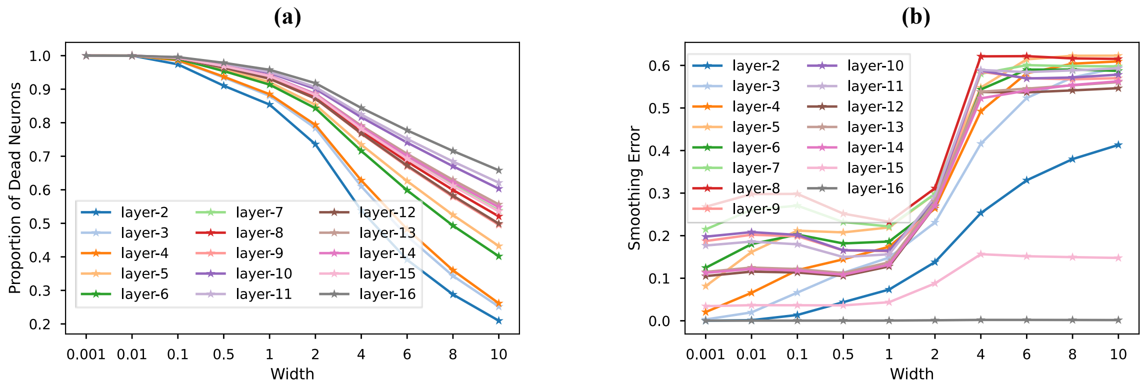

Figure 6. The proportion of dead neurons (**a**) and smoothing error (**b**) in relation to the changing width parameter α on CIFAR-10.

B.8. Effect of Adaptive α ($\alpha \nrightarrow 0$)

In practice, as depicted in Figure 3**f**, the αs have not approached 0 against spike noise. Based on the approximation in Section 4.3 , minimizing the loss of noisy network $\ell_{\text{noise}}(\mathbf{F}, \mathbf{s})$ can be approximated into minimizing the loss of the embedded SNN $\ell_{\text{snn}}(\mathbf{F}, \mathbf{s})$ regularized by the layerwise distance between $\Theta(\hat{\mathbf{u}}^l)$ and $H_\alpha(\hat{\mathbf{u}}^l)$. As the width α approaches 0, minimizing the first term of $\boldsymbol{\ell}_{\text{snn}}\ (\boldsymbol{F}, \boldsymbol{s})$ tends to difficult due to the phenomenon of dead neurons, while the second term decreases with the layer-wise distance between noisy networks and SNNs (refer to the Appendix B.7). Therefore, the alphas tend to be optimized adaptively to make a tradeoff between both losses in ASGL rather than just approaching 0.

To investigate the effect of adaptive alpha on the network evolution, we train a noisy network on the CIFAR-10 dataset under the conditions of adaptive alpha and without adaptive alpha, respectively. After each training epoch, we record layer-wise activations of the noisy network and the embedded SNN with shared weights for each sample, and calculate the average cosine similarities S over all layers. We compare the network similarities of both cases in Figure 7. The results show that, in general, adaptive α achieves higher similarities (over 9%) with the embedded SNN than the case without adaptive α. This provides additional evidence for the effectiveness of adaptive α in helping the noisy network converge to an SNN, even if it does not approach 0.

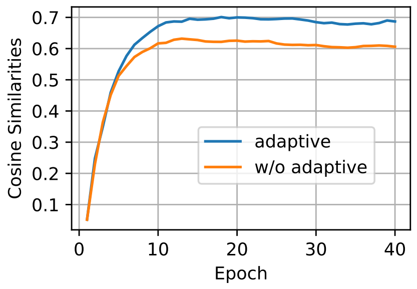

Figure 7. Comparing network similarities under the cases of adaptive α and w/o adaptive α.

B.9. Results without Auto-Augment and Cutout

Table 14. Comparing ASGL with the SG method without using auto-augment and cutout.

Width (α)	**CIFAR-10**		**CIFAR-100**	
	SG	**ASGL**	**SG**	**ASGL**
0.5	92.67	**93.25**	73.15	**73.51**
1.0	93.05	**93.33**	67.49	**74.12**
2.5	90.98	**92.74**	32.01	**73.32**
5.0	78.04	**92.01**	12.86	**73.90**
10.0	36.96	**91.52**	8.90	**73.03**

We have conducted additional experiments under 3 time steps to evaluate the performance of ASGL without using auto-augment and cutout. Similar to Table 2, the results in Table 14 show that ASGL outperforms SG across a wide range of width initialization on both datasets. The experiments demonstrate that catastrophic damage still exists for SG when width α is inappropriately selected, such as $\alpha = 1$ in the CIFAR-100 dataset. In contrast, ASGL exhibits remarkable robustness for different widths α and achieves notable performance improvements.