STRUCTURAL REGULARIZATION FOR SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking Neural Networks (SNNs) have gained increasing popularity as a biologically plausible alternative to traditional Artificial Neural Networks (ANNs). SNNs are both cost-efficient and deployment-friendly, as they process input in a spatiotemporal manner using binary spikes. It is widely believed that increasing the number of timesteps enables SNNs to capture more features, thereby improving accuracy. However, we observe that as the number of timesteps grows, the risk of overfitting also increases, leading to diminishing returns in accuracy improvement. To handle this issue, we propose a structural regularization technique for SNNs, called DropTime. Specifically, during training, we randomly remove information from certain timesteps. We also provide a theoretical justification showing that this simple operation effectively mitigates overfitting. Experimental results demonstrate that our method consistently improves SNN accuracy on both widely used static non-spiking and neuromorphic datasets. We offer a straightforward yet effective approach for training high-accuracy SNNs.

1 INTRODUCTION

Spiking Neural Networks (SNNs) have emerged as a popular class of neural networks, gaining significant attention for their efficiency. They have been successfully applied in various domains, such as object recognition Li et al. (2021a); Xiao et al. (2021); Xu et al. (2025b); Shen et al. (2025a), object detection Kim et al. (2019); Qu et al. (2023); Miao et al. (2025); Xu et al. (2025a), and pose tracking Zou et al. (2023). Unlike traditional neural networks, SNNs transmit information using binary spike signals. A spiking neuron generates a spike (represented by 1) when its membrane potential exceeds a predefined threshold, otherwise, it remains silent (represented by 0). This event-driven information processing mechanism offers substantial energy efficiency, as it replaces the multiplication of weights and activations with simple additions. Furthermore, this paradigm can be implemented efficiently on neuromorphic hardware Ma et al. (2017); Akopyan et al. (2015); Davies et al. (2018); Pei et al. (2019), where computational units are only activated when a spike occurs, conserving energy by remaining idle when no spike is generated. Studies have demonstrated that SNNs can achieve orders of magnitude more energy efficiency compared to their Artificial Neural Network (ANN) counterparts Akopyan et al. (2015); Davies et al. (2018); Shen et al. (2025b); Xu et al. (2024).

Despite the superior energy efficiency of SNNs, they often underperform in tasks compared to ANNs. This limitation arises from the binary nature of spike processing in SNNs, which restricts their expressiveness and leads to significant accuracy degradation when compared to the full-precision feature maps used in ANNs Guo et al. (2023a; 2022a); Shen et al. (2024); Zhang et al. (2024a); Yao et al. (2025; 2024). The capacity of SNNs to process information is primarily influenced by the number of timesteps Guo et al. (2023a); Li et al. (2023); Luo et al. (2024). Increasing the number of timesteps allows SNNs to capture more features, potentially enhancing accuracy. However, we have observed that as the number of timesteps increases, the risk of overfitting also grows, resulting in diminishing returns in accuracy improvements. Surprisingly, we have found no prior research addressing this issue.

In this paper, we propose a novel structural regularization technique to address the overfitting problem associated with increasing timesteps in SNNs. The core idea of our approach is to randomly remove information from certain timesteps during training, in a manner similar to the dropout tech-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

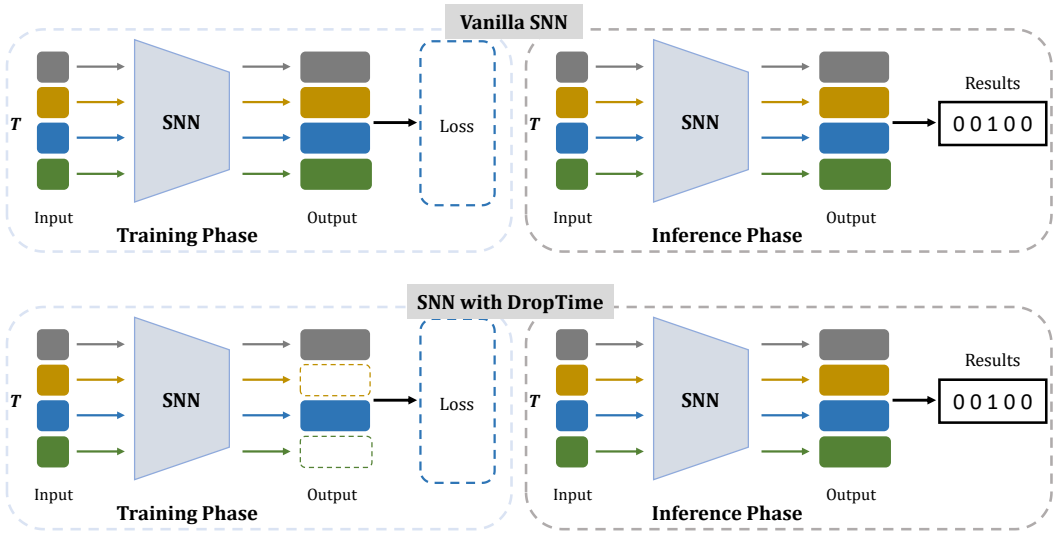


Figure 1: The difference between SNN with DropTime and the vanilla SNN. We randomly remove information from certain timesteps during training.

nique used in conventional neural networks; however, DropTime differs significantly from dropout in key aspects, and these differences will be elaborated in detail later in this paper. During inference, however, all temporal features are retained. We refer to this method as DropTime. The DropTime technique effectively mitigates the overfitting risk induced by the increase in timesteps.

The difference between SNN with DropTime and the vanilla SNN is illustrated in Figure 1. Our contributions can be summarized as follows:

- First, we introduce a simple yet powerful regularization method called DropTime for SNNs. By randomly removing information from multiple timesteps during training, DropTime reduces the overfitting risk caused by an increased number of timesteps in SNNs.
- Second, we provide theoretical justifications to demonstrate that DropTime effectively mitigates the overfitting problem associated with increasing timesteps in SNNs.
- Finally, we evaluate the effectiveness of our method on both static (CIFAR-10, CIFAR-100, ImageNet) and spiking (CIFAR10-DVS) datasets, using widely adopted network architectures. Our experimental results show that SNNs trained with DropTime achieve better accuracy.

2 Related Work

2.1 Learning of Spiking Neural Networks

Three commonly employed learning paradigms for SNNs are unsupervised learning Diehl & Cook (2015); Hao et al. (2020); Deng et al. (2025), ANN-to-SNN conversion (ANN2SNN) Sengupta et al. (2019); Hao et al. (2023a;b); Deng & Gu (2021), and supervised learning Li et al. (2021b); Guo et al. (2022c); Hu et al. (2024); Zhang et al. (2025a;a); Deng et al. (2024). In unsupervised learning, the spike-timing-dependent plasticity (STDP) rule Lobov et al. (2020) is used to adjust the SNN model, which is considered more biologically plausible. However, without explicit task guidance, this method is typically constrained to small-scale networks and datasets. The ANN-SNN conversion approach Han & Roy (2020); Li et al. (2021a); Bu et al. (2021); Ho & Chang (2021); Bu et al. (2022); Hao et al. (2023a); Lan et al. (2023) involves initially training an ANN and subsequently converting it into an SNN by transferring the trained weights and replacing the ReLU activation function with a spiking neuron. However, this method struggles to capture the rich temporal dynamics required for neuromorphic datasets, as the ReLU neuron in the ANN cannot effectively model sequential information. On the other hand, supervised learning Fang et al. (2021a); Wu et al. (2018);

Liang et al. (2025); Wang et al. (2025a); Zeng et al. (2025) adapts the backpropagation algorithm by replacing the spiking process with a differentiable surrogate function, enabling direct training of the SNN as if it were an ANN. This gradient-based optimization technique allows SNNs to achieve strong performance with just a few timesteps, even on large-scale datasets. Moreover, supervised learning can handle temporal data efficiently, making it an increasingly popular approach in SNN research. Our work is also situated within this paradigm.

2.2 Regularization Techniques for Spiking Neural Networks

Several regularization techniques have been proposed to enhance the performance of SNNs. In RecDis-SNN Guo et al. (2022c), a novel approach was introduced to address the gradient explosion and vanishing problems in SNNs by classifying the undesired shifts in the membrane potential distribution. To mitigate these shifts, a membrane potential regularization loss was proposed, which does not introduce additional operations during SNN inference. In TET Deng et al. (2022), a temporal regularization loss was introduced to address the loss of momentum in gradient descent methods, enabling the model to converge to flatter minima and improving generalization. A correlation-based regularizer Han & Lee (2022) was incorporated into the loss function to reduce feature redundancies across layers, promoting structural sparsity and benefiting energy efficiency. Superspike Zenke & Ganguli (2018) proposed adding a heterosynaptic regularization term to the hidden layer weights to prevent excessively high firing rates. In Pellegrini et al. (2021), sparse spiking activity was enforced by applying l_1 or l_2 regularization to the total spike count in each layer. The Moderate Dropout was proposed in the LTMD Wang et al. (2022), which addresses the issue of "sub-model consistency in the spatial dimension" and enhances training stability through dynamic probabilities and distribution constraints, whereas DropTime solves the problem of "temporal-dimensional overfitting" by randomly masking timesteps to learn robust temporal features. In conclusion, none of these methods specifically addresses the overfitting issue caused by increasing the number of timesteps in SNNs.

2.3 Dropout Techniques

Dropout Srivastava et al. (2014); Hinton et al. (2012) revolutionized regularization in deep learning by randomly deactivating neurons during training, which prevents the co-adaptation of features and mitigates overfitting. This technique is particularly effective in data-scarce scenarios, as it forces the network to learn redundant, robust representations. Initially, dropout was applied to fully connected models, where all neurons in one layer are connected to all neurons in the next layer. Over time, numerous variations of dropout have been proposed to address different challenges, such as DropConnect Wan et al. (2013), DropBlock Ghiasi et al. (2018), Skipout Moayed & Mansoori (2022), MaxDropout do Santos et al. (2021), DropPath Larsson et al. (2016), selective CNN dropout, cutout DeVries & Taylor (2017), adversarial dropout Park et al. (2018), AutoDropout Pham & Le (2021), R-Drop Wu et al. (2021b), Fraternal Dropout Zolna et al. (2017), DropGNN Papp et al. (2021), and spectral dropout Khan et al. (2019). These variants have explored regularization from different perspectives and applied dropout to a variety of network architectures. In this paper, we introduce DropTime, a novel method to address the overfitting problem caused by an increase in timesteps in SNNs.

3 Methodology

In this section, we begin by elucidating the challenges posed by the increase in timesteps in SNNs. Subsequently, we introduce our technique, referred to as DropTime, which is based on robust theoretical justifications. Finally, we demonstrate how DropTime effectively mitigates the overfitting risk associated with increasing timesteps in SNNs.

3.1 Spiking Neural Networks

SNNs aim to replicate the functioning of the human brain using spiking neurons. In our study, we utilize the well-established spiking Leaky-Integrate-and-Fire (LIF) neuron model, which accurately reflects the dynamics of biological neurons by accounting for the interplay between the membrane potential and input current. The variable controlling the LIF dynamics, known as the membrane potential, is denoted by $\boldsymbol{u}$. At each time step, the membrane potential is updated according to the

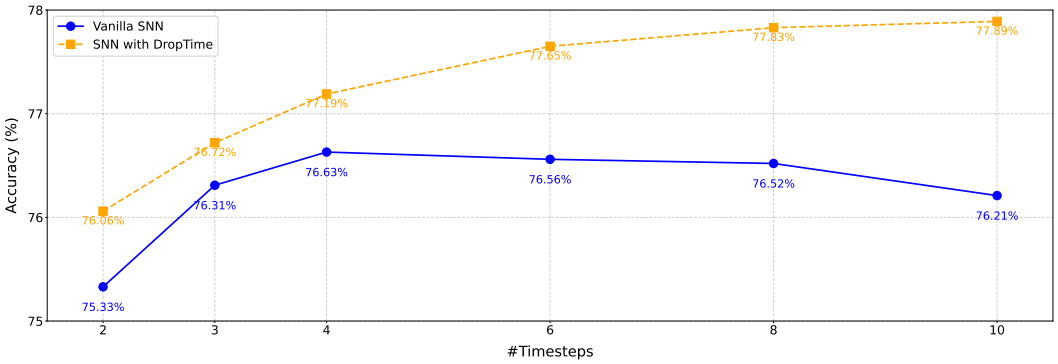


Figure 2: Accuracy of spiking ResNet18 across various timesteps on CIFAR100.

following equation:

$$\boldsymbol{u}^{(t+1),\text{pre}} = \tau \boldsymbol{u}^{(t)} + \boldsymbol{c}^{(t+1)}, \quad \text{where } \boldsymbol{c}^{(t+1)} = \mathbf{W}\boldsymbol{x}^{(t+1)}, \tag{1}$$

where τ is a constant in the interval $(0, 1)$ that governs the leakage of the membrane potential. The term $\boldsymbol{c}^{(t+1)}$ represents the pre-synaptic input at timestep $t+1$, which accumulates input current. The input current is computed as the dot product between the weights and the spikes from the preceding layer.

Once the membrane potential surpasses the firing threshold, V_{th}, a spike is generated by the LIF neuron, characterized by the following equations:

$$\boldsymbol{y}^{(t+1)} = \begin{cases} 1 & \text{if } \boldsymbol{u}^{(t+1),\text{pre}} > V_{\text{th}} \\ 0 & \text{otherwise} \end{cases}, \quad \boldsymbol{u}^{(t+1)} = \boldsymbol{u}^{(t+1),\text{pre}} \cdot (1 - \boldsymbol{y}^{(t+1)}). \tag{2}$$

After firing, the spike output $\boldsymbol{y}^{(t+1)}$ transmits to the next layer, becoming the input for that layer.

The Classifier in SNN. Typically, in a classification model based on neural networks, the final output is utilized to compute softmax values for predicting the target class. If we solely rely on the count of spikes to determine probabilities, we risk losing critical information, as the network outputs could be both positive and negative. However, spikes can only yield positive values. Consequently, most research opts to integrate network outputs without generating spikes over time, as seen in recent works Deng et al. (2022); Guo et al. (2022a); Yu et al. (2021).

$$\boldsymbol{y}_{\text{net}} = \frac{1}{T}\sum_{t=1}^{T} h \circ g_L \circ g_{L-1} \circ \cdots \circ g_1(\boldsymbol{x}_{\text{in}}^t), \tag{3}$$

where $h(\cdot)$ represents the final linear classifier and $g_l(\cdot)$ refers to the l-th block of the backbone network. Here, L indicates the total number of blocks within the network, and each block comprises a convolutional layer, a normalization layer, and a LIF layer. In this study, we employ a direct encoding method, utilizing $g_1(\boldsymbol{x}_{\text{in}}^t)$ to convert the input tensor into spike trains, as demonstrated in recent SNN research Deng et al. (2022); Guo et al. (2022c); Li et al. (2021b). We then compute the cross-entropy loss based on the true label and $\text{Softmax}(\boldsymbol{y}_{\text{net}})$.

3.2 Introducing DropTime to SNNs

Traditionally, increasing the number of timesteps in SNNs allows for the capture of more temporal features, which can enhance model accuracy. However, as timesteps continue to grow, the risk of overfitting escalates—this not only leads to diminishing returns in accuracy improvements but may even cause performance degradation. Figure 2 illustrates the accuracy versus the number of timesteps for the vanilla spiking ResNet18 on the CIFAR100 dataset, where the model is trained from scratch with timesteps set to 2, 3, 4, 6, 8, and 10 respectively. The performance evaluation reveals a clear trend: **(I)** When timesteps increase from 2 to 3, top-1 accuracy rises significantly by 0.98% (from 75.33% to 76.31%); **(II)** When timesteps further extend from 3 to 4, the accuracy gain shrinks sharply to only 0.32% (reaching 76.63%, the peak in the experiment); **(III)** As timesteps

continue to increase beyond 4 (i.e., 4→6→8→10), accuracy no longer improves but instead shows a steady downward trend—dropping to 76.56% at 6 steps, 76.52% at 8 steps, and finally 76.21% at 10 steps (lower than the accuracy at 4 steps). *This trend fully confirms that with the increase of timesteps in SNNs, the accuracy gain first becomes marginal and then even reverses, which is primarily attributed to the heightened risk of overfitting.*

This observation motivates us to investigate more effective training strategies to address overfitting resulting from increasing timesteps in SNNs. Dropout, a widely-used regularization technique, is designed to prevent neural network models from overfitting during the training phase by randomly deactivating neurons, thereby reducing feature co-adaptation. In this paper, we aim to adapt this principle to alleviate the overfitting issues associated with increasing timesteps in SNNs.

We propose a structural regularization technique for SNNs called DropTime, which involves the random removal of some information from several timesteps during training. Let $\mathcal{T} = \{1, 2, \ldots, T\}$ denote the set of timesteps. For each training iteration, we create a binary mask $\boldsymbol{m} \in \{0, 1\}^T$, with each element m^t independently drawn from a Bernoulli distribution:

$$m^t \sim \text{Bernoulli}(p), \quad \forall t \in \mathcal{T}, \tag{4}$$

where $p \in (0, 1]$ signifies the retention probability of the timesteps.

The modified temporal integration incorporating dropout can be expressed as:

$$\boldsymbol{y}_{\text{net}}^{\text{drop}} = \frac{1}{\|\boldsymbol{m}\|_1} \sum_{t=1}^{T} m^t \cdot \left[h \circ g^L \circ \cdots \circ g^1(\boldsymbol{x}_{\text{in}}^t)\right], \tag{5}$$

where $\|\boldsymbol{m}\|_1 = \sum_{i=1}^{T} m^i$ denotes the number of retained timesteps. This formulation ensures that:

$$\mathbb{E}[\boldsymbol{y}_{\text{net}}^{\text{drop}}] = \mathbb{E}[\boldsymbol{y}_{\text{net}}], \tag{6}$$

which will be proved in the next subsection.

The complete forward pass with temporal dropout can be formulated as follows:

$$\boldsymbol{y}_0^t = \boldsymbol{x}_{\text{in}}^t, \tag{7}$$

$$\boldsymbol{y}_l^t = g_l(\boldsymbol{y}_{l-1}^t), \tag{8}$$

$$\boldsymbol{y}^t = h(\boldsymbol{y}_L^t), \tag{9}$$

$$\boldsymbol{y}_{\text{net}}^{\text{drop}} = \frac{1}{\|\boldsymbol{m}\|_1} \sum_{t=1}^{T} m^t \boldsymbol{y}^t, \tag{10}$$

where $\boldsymbol{y}_l^t$ signifies the spike activity at layer l during timestep t.

During inference, we deactivate the dropout mechanism and compute the standard temporal average:

$$\boldsymbol{y}_{\text{net}} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{y}^t. \tag{11}$$

The cross-entropy loss during training is computed using the dropout-affected outputs:

$$\mathcal{L} = -\sum_{c=1}^{C} \boldsymbol{y}_c \log \left(\frac{\exp(\boldsymbol{y}_{\text{net},c}^{\text{drop}})}{\sum_{j=1}^{C} \exp(\boldsymbol{y}_{\text{net},j}^{\text{drop}})} \right), \tag{12}$$

where C denotes the number of classes and y_c represents the true label. The implementation of DropTime requires only straightforward modifications to standard SNN training procedures, ensuring compatibility with existing spiking neuron models and backpropagation-through-time techniques.

The difference with Dropout. Although DropTime is inspired by dropout, the two differ in three key aspects: (i) *action object*—dropout deactivates neurons or connections in the spatial dimension, whereas DropTime masks whole timesteps in the temporal dimension unique to SNNs; (ii) *objective*—dropout targets general overfitting, while DropTime specifically tackles the overfitting induced by increasing timesteps; (iii) *location*—dropout acts on intermediate layers, whereas DropTime operates on the output layer where temporal features are aggregated. Applying dropout directly to the

output layer would alter its expected value and disrupt the predicted distribution, an issue DropTime avoids, as formalized in the next subsection.

The difference with Temporal Dropout. A seemingly equivalent alternative is to adapt standard dropout into a Temporal Dropout variant, i.e., normalizing by the expected retained count pT rather than the actual count $\|\boldsymbol{m}\|_1$. Although both estimators are unbiased, DropTime's $1/\|\boldsymbol{m}\|_1$ normalization yields a lower-variance, per-iteration estimate of $\boldsymbol{y}_{\text{net}}$, which stabilizes optimization and brings a clear accuracy gain. A detailed empirical comparison is provided in the supplementary material.

3.3 Theoretical Analysis of DropTime

In this section, we first validate that the expectation of network outputs remains consistent between the training with DropTime and inference phases without DropTime, thereby ensuring the accuracy of our method. Furthermore, we demonstrate that DropTime introduces an implicit regularization term, which contributes to mitigating temporal overfitting. To prove the expectation of network outputs remains consistent of DropTime, we present the following theorem:

Theorem 3.1. *Under the DropTime mechanism with retention probability p, the expectation of network outputs is consistent between the training and inference phases:*

$$\mathbb{E}_{\boldsymbol{m}}[\boldsymbol{y}_{\text{net}}^{\text{drop}}] = \boldsymbol{y}_{\text{net}}. \tag{13}$$

Proof. Let $\boldsymbol{y}^t = h \circ g^L \circ \cdots \circ g^1(\boldsymbol{x}_{\text{in}}^t)$ denote the temporal output at timestep t. For the Bernoulli mask $\boldsymbol{m}$:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{m}}[\boldsymbol{y}_{\text{net}}^{\text{drop}}] &= \mathbb{E}\left[\frac{1}{\|\boldsymbol{m}\|_1}\sum_{t=1}^{T} m^t \boldsymbol{y}^t\right] \\
&= \sum_{k=1}^{T} \mathbb{E}\left[\frac{1}{k}\sum_{t=1}^{T} m^t \boldsymbol{y}^t \mid \|\boldsymbol{m}\|_1 = k\right] P(\|\boldsymbol{m}\|_1 = k) \\
&= \sum_{k=1}^{T} \frac{1}{k}\sum_{t=1}^{T} \mathbb{E}[m^t \mid \|\boldsymbol{m}\|_1 = k]\boldsymbol{y}^t P(\|\boldsymbol{m}\|_1 = k) \\
&= \sum_{k=1}^{T} \frac{1}{k}\sum_{t=1}^{T} \frac{k}{T}\boldsymbol{y}^t P(\|\boldsymbol{m}\|_1 = k) \\
&= \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{y}^t \sum_{k=1}^{T} P(\|\boldsymbol{m}\|_1 = k) \\
&= \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{y}^t = \boldsymbol{y}_{\text{net}}.
\end{aligned}$$

Next, we establish that DropTime effectively addresses the overfitting risks associated with SNNs through the following theorem:

Theorem 3.2. *The DropTime mechanism introduces an implicit regularization term proportional to the variance of temporal features:*

$$\mathbb{E}_{\boldsymbol{m}}[\mathcal{L}(\boldsymbol{y}_{\text{net}}^{\text{drop}})] = \mathcal{L}_0 + \lambda \cdot \mathbb{E}_{\boldsymbol{m}}\left[\text{Var}(\boldsymbol{y}_{\text{net}}^{\text{drop}})\right], \tag{14}$$

where $\mathcal{L}_0$ is the original loss, $\lambda = \frac{1}{2}\text{Tr}(\nabla^2\mathcal{L}(\boldsymbol{y}_{\text{net}})) > 0$ represents the curvature-dependent regularization coefficient, and $\text{Var}(\boldsymbol{y}_{\text{net}}^{\text{drop}}) = \mathbb{E}_{\boldsymbol{m}}[(\boldsymbol{y}_{\text{net}}^{\text{drop}} - \boldsymbol{y}_{\text{net}})(\boldsymbol{y}_{\text{net}}^{\text{drop}} - \boldsymbol{y}_{\text{net}})^\top]$ denotes the output covariance matrix.

Proof. Using the Taylor expansion of the loss function around $\boldsymbol{y}_{\text{net}}$, we have:

$$\mathcal{L}(\boldsymbol{y}_{\text{net}}^{\text{drop}}) \approx \mathcal{L}(\boldsymbol{y}_{\text{net}}) + \underbrace{\nabla\mathcal{L}(\boldsymbol{y}_{\text{net}})^\top(\boldsymbol{y}_{\text{net}}^{\text{drop}} - \boldsymbol{y}_{\text{net}})}_{\text{First-order term}} + \frac{1}{2}\underbrace{(\boldsymbol{y}_{\text{net}}^{\text{drop}} - \boldsymbol{y}_{\text{net}})^\top\nabla^2\mathcal{L}(\boldsymbol{y}_{\text{net}})(\boldsymbol{y}_{\text{net}}^{\text{drop}} - \boldsymbol{y}_{\text{net}})}_{\text{Second-order term}}.$$

Next, we take the expectation with respect to the mask distribution $\boldsymbol{m} \sim \text{Bernoulli}(p)^T$:

$$\mathbb{E}_{\boldsymbol{m}}[\mathcal{L}(\boldsymbol{y}_{\text{net}}^{\text{drop}})] \approx \mathcal{L}_0 + \nabla\mathcal{L}(\boldsymbol{y}_{\text{net}})^\top\underbrace{\mathbb{E}_{\boldsymbol{m}}[\boldsymbol{y}_{\text{net}}^{\text{drop}} - \boldsymbol{y}_{\text{net}}]}_{=\mathbf{0}} + \frac{1}{2}\mathbb{E}_{\boldsymbol{m}}\left[(\boldsymbol{y}_{\text{net}}^{\text{drop}} - \boldsymbol{y}_{\text{net}})^\top\nabla^2\mathcal{L}(\boldsymbol{y}_{\text{net}})(\boldsymbol{y}_{\text{net}}^{\text{drop}} - \boldsymbol{y}_{\text{net}})\right].$$

The linear term vanishes due to the unbiased property $\mathbb{E}_{\boldsymbol{m}}[\boldsymbol{y}_{\text{net}}^{\text{drop}}] = \boldsymbol{y}_{\text{net}}$.

Utilizing the identity for quadratic forms, we have:

$$\begin{aligned}&\mathbb{E}\left[(\boldsymbol{y}_{\text{net}}^{\text{drop}} - \boldsymbol{y}_{\text{net}})^\top\nabla^2\mathcal{L}(\boldsymbol{y}_{\text{net}}^{\text{drop}} - \boldsymbol{y}_{\text{net}})\right]\\ &= \text{Tr}\left(\nabla^2\mathcal{L}\cdot\mathbb{E}\left[(\boldsymbol{y}_{\text{net}}^{\text{drop}} - \boldsymbol{y}_{\text{net}})(\boldsymbol{y}_{\text{net}}^{\text{drop}} - \boldsymbol{y}_{\text{net}})^\top\right]\right)\\ &= \text{Tr}(\nabla^2\mathcal{L}\cdot\text{Var}(\boldsymbol{y}_{\text{net}}^{\text{drop}})).\end{aligned}$$

By combining all components, we arrive at:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{m}}[\mathcal{L}(\boldsymbol{y}_{\text{net}}^{\text{drop}})] &\approx \mathcal{L}_0 + \frac{1}{2}\text{Tr}(\nabla^2\mathcal{L}\cdot\text{Var}(\boldsymbol{y}_{\text{net}}^{\text{drop}}))\\ &= \mathcal{L}_0 + \frac{1}{2}\mathbb{E}_{\boldsymbol{m}}\left[\text{Tr}(\nabla^2\mathcal{L}\cdot\text{Var}(\boldsymbol{y}_{\text{net}}^{\text{drop}}))\right]\\ &= \mathcal{L}_0 + \underbrace{\frac{1}{2}\text{Tr}(\nabla^2\mathcal{L})}_{\lambda}\cdot\mathbb{E}_{\boldsymbol{m}}[\text{Var}(\boldsymbol{y}_{\text{net}}^{\text{drop}})].\end{aligned}$$

This conclusion reveals an implicit regularization term proportional to the variance of the outputs induced by the DropTime masking. These combined effects lend mathematical support to the efficacy of DropTime in preventing temporal overfitting.

To further verify that DropTime alleviates overfitting, we also provide a train–test accuracy analysis in the supplementary material, which shows that DropTime narrows the train–test gap while simultaneously improving the test accuracy.

4 Experiment

We performed extensive experiments on several datasets, including CIFAR10(100) Krizhevsky et al. (2010), ImageNet Deng et al. (2009), and CIFAR10-DVS Li et al. (2017), to demonstrate the superior performance of our proposed method. The CIFAR10(100) dataset consists of 50k training images and 10k test images, divided into 10 (or 100) classes, with each image having a resolution of 32×32 pixels. The CIFAR10-DVS dataset is a converted version of CIFAR10, comprising 10k images, with 1k images per class, across 10 classes. ImageNet, a much larger dataset, contains over 1.25 million training images and 50k test images.

For all datasets, we set the firing threshold V_{th} to 1.0, the membrane potential leakage τ to 0.5, and the retention probability p for the timesteps to 0.9. All experiments were conducted on 8 NVIDIA A100 GPUs. Due to page constraints, the detailed experimental setup is provided in appendix.

4.1 Ablation Study

To assess the effectiveness of the proposed method, we performed several ablation studies on the CIFAR100 dataset using ResNet18 with varying timesteps. The results are shown in Figure 2.With

vanilla training, the accuracy achieved was 75.33%, 76.31%, 76.63%, 76.56%, 76.52%, and 76.21% for 2, 3, 4, 6, 8, and 10 timesteps, respectively—consistent with previous work initially, but accuracy began to decline after 4 timesteps, clearly reflecting overfitting as more timesteps failed to improve performance and even led to accuracy degradation.By applying our DropTime technique, the accuracy reached 76.06%, 76.72%, 77.19%, 77.65%, 77.83%, and 77.89% for the same timesteps. This result indicates that our method not only avoids overfitting but also maintains continuous accuracy improvement with more timesteps, effectively mitigating the diminishing returns in accuracy improvement and offering a promising solution for training SNNs with temporal dynamics.

We further conducted experiments on CIFAR10-DVS using ResNet18 with 10 timesteps. The baseline accuracy is only 79.00%, but by applying the DropTime technique, the accuracy increases to 83.50%, demonstrating a significant improvement. It is important to note that CIFAR-DVS is a relatively small dataset. This result highlights the effectiveness of our technique in data-scarce scenarios, as it encourages the network to learn redundant and robust representations.

Table 1: Comparison with SoTA methods on CIFAR10(100).

Dataset	Method	Type	Architecture	Timestep	Accuracy
CIFAR-10	Adaptive Calibration Wang et al. (2025b)	ANN2SNN	ResNet20	8	95.47%
	AGMM Liang et al. (2025)	SNN training	ResNet19	4	96.33%
	QP-SNN Wei et al. (2025)	SNN training	ResNet20	2	94.56%
	Dynamic structure learning Shen et al. (2025b)	SNN training	ResNet19	2	92.48%
	PTL Wu et al. (2021a)	Tandem Learning	VGG11	16	91.24%
	PLIF Fang et al. (2021b)	SNN training	PLIFNet	8	93.50%
	DSR Meng et al. (2022)	SNN training	ResNet18	20	95.40%
	KDSNN Xu et al. (2023)	SNN training	ResNet18	4	93.41%
	RecDis-SNN Guo et al. (2022c)	SNN training	ResNet19	2	93.64%
	Diet-SNN Rathi & Roy (2020)	SNN training	ResNet20	5	91.78%
				10	92.54%
	Dspike Li et al. (2021b)	SNN training	ResNet20	2	93.13%
				4	93.66%
	FSTA-SNN Yu et al. (2025)	SNN training	ResNet20	2	94.18%
				4	94.72%
	STAA-SNN Zhang et al. (2025b)	SNN training	ResNet20	2	94.35%
				4	95.03%
	TET Deng et al. (2022)	SNN training	ResNet19	2	94.16%
				4	94.44%
	Real Spike Guo et al. (2022d)	SNN training	ResNet19	2	95.31%
			ResNet20	4	91.89%
	Our method	SNN training	ResNet18	2	**95.24%**
				4	**95.66%**
			ResNet19	2	**96.40%**
CIFAR-100	RMP Han et al. (2020)	ANN2SNN	ResNet20	2048	67.82%
	QP-SNN Wei et al. (2025)	SNN training	ResNet20	2	74.78%
	Dynamic structure learning Shen et al. (2025b)	SNN training	ResNet19	2	70.03%
	T2FSNN Park et al. (2020)	ANN2SNN	VGG16	680	68.80%
	Real Spike Guo et al. (2022d)	SNN training	ResNet20	5	66.60%
	LTL Yang et al. (2022)	Tandem Learning	ResNet20	31	76.08%
	Diet-SNN Rathi & Roy (2020)	SNN training	ResNet20	5	64.07%
	RecDis-SNN Guo et al. (2022c)	SNN training	ResNet19	4	74.10%
	Dspike Li et al. (2021b)	SNN training	ResNet20	2	71.68%
				4	73.35%
	TET Deng et al. (2022)	SNN training	ResNet19	2	72.87%
				4	74.47%
	FSTA-SNN Yu et al. (2025)	SNN training	ResNet20	2	72.15%
				4	73.44%
	STAA-SNN Zhang et al. (2025b)	SNN training	ResNet20	2	73.20%
				4	75.10%
	Our method	SNN training	ResNet18	2	**76.06%**
				4	**77.19%**
			ResNet19	2	**79.10%**

4.2 Comparison with SoTA Methods

In this section, we present a comparative analysis of our proposed method against several state-of-the-art (SoTA) approaches, focusing on top-1 accuracy. We first evaluated our method on the

Table 2: Comparison with SoTA methods on ImageNet.

Method	Type	Architecture	Timestep	Accuracy
TAB Jiang et al. (2024)	SNN training	ResNet34	4	67.78%
TET Deng et al. (2022)	SNN training	ResNet34	6	64.79%
RecDis-SNN Guo et al. (2022c)	SNN training	ResNet34	6	67.33%
OTTT Xiao et al. (2022)	SNN training	ResNet34	6	65.15%
GLIF Yao et al. (2022)	SNN training	ResNet34	4	67.52%
DSR Meng et al. (2022)	SNN training	ResNet18	50	67.74%
MS-ResNet Hu et al. (2023)	SNN training	ResNet18	6	63.10%
MPBN Guo et al. (2023b)	SNN training	ResNet18	4	63.14%
		ResNet34	4	64.71%
EnOF-SNN Guo et al. (2024)	SNN training	ResNet18	4	65.31%
		ResNet34	4	67.40%
SEW ResNet Fang et al. (2021a)	SNN training	ResNet18	4	63.18%
		ResNet34	4	67.04%
SSCL Zhang et al. (2024b)	SNN training	ResNet18	4	62.95%
		ResNet34	4	66.78%
Our method	SNN training	ResNet18	4	**65.40%**
		ResNet34	4	**68.78%**

CIFAR10 and CIFAR100 datasets. For the experiments, we employed the AdamW optimizer with a learning rate of 0.01, which decays to 0 via cosine annealing, and a weight decay of 0.02. All models were trained for 300 epochs with a batch size of 128. The results of these experiments are summarized in Table 1, which highlights the advantage of our method in comparison to others across both datasets. On the CIFAR10 dataset, previous works using ResNet18, ResNet19, and ResNet20 as backbones achieved peak accuracies of 95.40%, 96.33%, and 95.30% with 20, 2, and 4 timesteps, respectively. In contrast, our method, using ResNet18 and ResNet19, achieved accuracies of 95.66% and 96.40% with 4 and 2 timesteps, respectively. Notably, ResNet18 is smaller than ResNet20, yet our method outperforms or matches the performance of prior methods with fewer timesteps. On the CIFAR100 dataset, our method also outperforms existing approaches, achieving higher accuracy with fewer timesteps. For instance, our ResNet18 model with just 2 timesteps surpasses the performance of the current SoTA method, STAA-SNN, even when using ResNet20 with 4 timesteps, by approximately 1.0%. These results underscore the efficiency and effectiveness of our approach, demonstrating that fewer timesteps can yield superior performance in certain scenarios.

We then conducted experiments on the more complex ImageNet dataset, adjusting the learning rate to $4e^{-3}$. The results of these experiments are presented in Table 2. Recently, several SoTA methods have been proposed for this dataset, including TAB Jiang et al. (2024), TET Deng et al. (2022), DSR Meng et al. (2022), MPBN Guo et al. (2023b), EnOF-SNN Guo et al. (2024), SSCL Zhang et al. (2024b), and SEW ResNet Fang et al. (2021a). It is important to note that some methods using ResNet deviate from the standard ResNet backbone by generating integer outputs in intermediate layers, which results in higher energy consumption compared to methods that use the standard backbone. In contrast, our approach uses the standard ResNet18 and ResNet34 architectures, yet still performs effectively. Specifically, using ResNet34, our method achieves an accuracy of 68.78%, surpassing TAB and EnOF-SNN by 1.0% and 1.38%, respectively. Additional results on the neuromorphic dataset CIFAR10-DVS are provided in the appendix.

5 Conclusion

In this paper, we introduced a novel technique, DropTime, aimed at addressing the overfitting challenge often encountered with increasing timesteps in SNNs. Through a detailed mathematical formulation, we demonstrated how DropTime effectively mitigates the risk of overfitting while maintaining the network's expected output. Our method was evaluated on both static datasets (CIFAR-10, CIFAR-100, ImageNet) and spiking datasets (CIFAR10-DVS), using widely adopted network architectures. The experimental results indicate that SNNs trained with DropTime achieve superior accuracy compared to conventional models.

AI USE STATEMENT

In this work, we used generative AI tools solely for language polishing, including improving writing clarity and grammar. All AI-assisted edits were reviewed and revised by the authors. We take full responsibility for the final content of this work, including all text, claims, and results.

REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce our method and experiments throughout the paper and appendix.

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.

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.

Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11–20, 2022.

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.

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

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

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

Shikuang Deng, Hao Lin, Yuhang Li, and Shi Gu. Surrogate module learning: Reduce the gradient error accumulation in training spiking neural networks. In *International Conference on Machine Learning*, pp. 7645–7657. PMLR, 2023.

Shikuang Deng, Yuhang Wu, Kangrui Du, and Shi Gu. Spiking token mixer: An event-driven friendly former structure for spiking neural networks. *Advances in Neural Information Processing Systems*, 37:128825–128846, 2024.

Shikuang Deng, Jiayuan Zhang, Yuhang Wu, Ting Chen, and Shi Gu. Rethinking hebbian principle: Low-dimensional structural projection for unsupervised learning. *arXiv preprint arXiv:2510.14810*, 2025.

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

Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

Yongqi Ding, Lin Zuo, Mengmeng Jing, Pei He, and Hanpu Deng. Rethinking spiking neural networks from an ensemble learning perspective. *arXiv preprint arXiv:2502.14218*, 2025.

Claudio Filipi Goncalves do Santos, Danilo Colombo, Mateus Roder, and João Paulo Papa. Max-dropout: deep neural network regularization based on maximum output values. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2671–2676. IEEE, 2021.

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

Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2661–2671, 2021b.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *Advances in neural information processing systems*, 31, 2018.

Yufei Guo, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Yinglei Wang, Xuhui Huang, and Zhe Ma. IM-loss: Information maximization loss for spiking neural networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022a. URL https://openreview.net/forum?id=Jw34v_84m2b.

Yufei Guo, Yuanpei Chen, Liwen Zhang, YingLei Wang, Xiaode Liu, Xinyi Tong, Yuanyuan Ou, Xuhui Huang, and Zhe Ma. Reducing information loss for spiking neural networks. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (eds.), *Computer Vision – ECCV 2022*, pp. 36–52, Cham, 2022b. Springer Nature Switzerland. ISBN 978-3-031-20083-0.

Yufei Guo, Xinyi Tong, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Zhe Ma, and Xuhui Huang. Recdis-snn: Rectifying membrane potential distribution for directly training spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 326–335, June 2022c.

Yufei Guo, Liwen Zhang, Yuanpei Chen, Xinyi Tong, Xiaode Liu, YingLei Wang, Xuhui Huang, and Zhe Ma. Real spike: Learning real-valued spikes for spiking neural networks. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XII*, pp. 52–68. Springer, 2022d.

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

Yufei Guo, Yuhan Zhang, Yuanpei Chen, Weihang Peng, Xiaode Liu, Liwen Zhang, Xuhui Huang, and Zhe Ma. Membrane potential batch normalization for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19420–19430, October 2023b.

Yufei Guo, Weihang Peng, Xiaode Liu, Yuanpei Chen, Yuhan Zhang, Xin Tong, Zhou Jie, and Zhe Ma. Enof-snn: Training accurate spiking neural networks via enhancing the output feature. *Advances in Neural Information Processing Systems*, 37:51708–51726, 2024.

Bing Han and Kaushik Roy. Deep spiking neural network: Energy efficiency through time based coding. In *European Conference on Computer Vision*, pp. 388–404. Springer, 2020.

Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13558–13567, 2020.

Chan Sik Han and Keon Myung Lee. Correlation-based regularization for fast and energy-efficient spiking neural networks. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, SAC '22, pp. 1048–1055, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387132.

Yunzhe Hao, Xuhui Huang, Meng Dong, and Bo Xu. A biologically plausible supervised learning method for spiking neural networks using the symmetric stdp rule. *Neural Networks*, 121:387–395, 2020.

Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. *arXiv preprint arXiv:2302.02091*, 2023a.

Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes. *arXiv preprint arXiv:2302.10685*, 2023b.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Nguyen-Dong Ho and Ik-Joon Chang. Tcl: an ann-to-snn conversion with trainable clipping layers. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 793–798. IEEE, 2021.

Yangfan Hu, Qian Zheng, and Gang Pan. Bitsnns: revisiting energy-efficient spiking neural networks. *IEEE Transactions on Cognitive and Developmental Systems*, 16(5):1736–1747, 2024.

Yifan Hu, Lei Deng, Yujie Wu, Man Yao, and Guoqi Li. Advancing spiking neural networks towards deep residual learning, 2023.

Haiyan Jiang, Vincent Zoonekynd, Giulia De Masi, Bin Gu, and Huan Xiong. Tab: Temporal accumulated batch normalization in spiking neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

Salman H Khan, Munawar Hayat, and Fatih Porikli. Regularization of deep neural networks with spectral dropout. *Neural Networks*, 110:82–90, 2019.

Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-yolo: Spiking neural network for energy-efficient object detection, 2019.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 5(4):1, 2010.

Yuxiang Lan, Yachao Zhang, Xu Ma, Yanyun Qu, and Yun Fu. Efficient converted spiking neural network for 3d and 2d classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9211–9220, 2023.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016.

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 *International Conference on Machine Learning*, pp. 6316–6325. PMLR, 2021a.

Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34:23426–23439, 2021b.

Yuhang Li, Tamar Geller, Youngeun Kim, and Priyadarshini Panda. Seenn: towards temporal spiking early exit neural networks. *Advances in Neural Information Processing Systems*, 36:63327–63342, 2023.

Yu Liang, Wenjie Wei, Ammar Belatreche, Honglin Cao, Zijian Zhou, Shuai Wang, Malu Zhang, and Yang Yang. Towards accurate binary spiking neural networks: Learning with adaptive gradient modulation mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 1402–1410, 2025.

S. A. Lobov, A. N. Mikhaylov, and V. B. Kazantsev. Spatial properties of stdp in a self-learning spiking neural network enable controlling a mobile robot. *Frontiers in Neuroscience*, 14:–, 2020.

Xinhao Luo, Man Yao, Yuhong Chou, Bo Xu, and Guoqi Li. Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection. In *European Conference on Computer Vision*, pp. 253–272. Springer, 2024.

De Ma, Juncheng Shen, Zonghua Gu, Ming Zhang, Xiaolei Zhu, Xiaoqiang Xu, Qi Xu, Yangjing Shen, and Gang Pan. Darwin: A neuromorphic hardware co-processor based on spiking neural networks. *Journal of Systems Architecture*, 77:43–51, 2017.

Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Training high-performance low-latency spiking neural networks by differentiation on spike representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12444–12453, 2022.

Wei Miao, Jiangrong Shen, Qi Xu, Timo Hamalainen, Yi Xu, and Fengyu Cong. Spikingyolox: Improved yolox object detection with fast fourier convolution and spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 1465–1473, 2025.

Hojjat Moayed and Eghbal G Mansoori. Skipout: an adaptive layer-level regularization framework for deep neural networks. *IEEE Access*, 10:62391–62401, 2022.

Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34:21997–22009, 2021.

Seongsik Park, Seijoon Kim, Byunggook Na, and Sungroh Yoon. T2fsnn: Deep spiking neural networks with time-to-first-spike coding. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2020.

Sungrae Park, JunKeon Park, Su-Jin Shin, and Il-Chul Moon. Adversarial dropout for supervised and semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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.

Thomas Pellegrini, Romain Zimmer, and Timothée Masquelier. Low-activity supervised convolutional spiking neural networks applied to speech commands recognition. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pp. 97–103, 2021. doi: 10.1109/SLT48900.2021.9383587.

Hieu Pham and Quoc Le. Autodropout: Learning dropout patterns to regularize deep networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 9351–9359, 2021.

Jinye Qu, Zeyu Gao, Tielin Zhang, Yanfeng Lu, Huajin Tang, and Hong Qiao. Spiking neural network for ultra-low-latency and high-accurate object detection, 2023.

Nitin Rathi and Kaushik Roy. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*, 2020.

Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

Hangchi Shen, Qian Zheng, Huamin Wang, and Gang Pan. Rethinking the membrane dynamics and optimization objectives of spiking neural networks. *Advances in Neural Information Processing Systems*, 37:92697–92720, 2024.

Jiangrong Shen, Wenyao Ni, Qi Xu, Gang Pan, and Huajin Tang. Context gating in spiking neural networks: Achieving lifelong learning through integration of local and global plasticity. *Knowledge-Based Systems*, 311:112999, 2025a.

Jiangrong Shen, Qi Xu, Gang Pan, and Badong Chen. Improving the sparse structure learning of spiking neural networks from the view of compression efficiency. *arXiv preprint arXiv:2502.13572*, 2025b.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066. PMLR, 2013.

Shuai Wang, Malu Zhang, Dehao Zhang, Ammar Belatreche, Yichen Xiao, Yu Liang, Yimeng Shan, Qian Sun, Enqi Zhang, and Yang Yang. Spiking vision transformer with saccadic attention. *arXiv preprint arXiv:2502.12677*, 2025a.

Siqi Wang, Tee Hiang Cheng, and Meng-Hiot Lim. Ltmd: learning improvement of spiking neural networks with learnable thresholding neurons and moderate dropout. *Advances in Neural Information Processing Systems*, 35:28350–28362, 2022.

Ziqing Wang, Yuetong Fang, Jiahang Cao, Hongwei Ren, and Renjing Xu. Adaptive calibration: A unified conversion framework of spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 1583–1591, 2025b.

Wenjie Wei, Malu Zhang, Zijian Zhou, Ammar Belatreche, Yimeng Shan, Yu Liang, Honglin Cao, Jieyuan Zhang, and Yang Yang. Qp-snn: Quantized and pruned spiking neural networks. *arXiv preprint arXiv:2502.05905*, 2025.

Jibin Wu, Chenglin Xu, Xiao Han, Daquan Zhou, Malu Zhang, Haizhou Li, and Kay Chen Tan. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7824–7840, 2021a.

Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al. R-drop: Regularized dropout for neural networks. *Advances in neural information processing systems*, 34:10890–10905, 2021b.

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

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.

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

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=Siv3nHYHheI.

Mengting Xu, De Ma, Huajin Tang, Qian Zheng, and Gang Pan. Feel-snn: Robust spiking neural networks with frequency encoding and evolutionary leak factor. *Advances in Neural Information Processing Systems*, 37:91930–91950, 2024.

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

Qi Xu, Jie Deng, Jiangrong Shen, Biwu Chen, Huajin Tang, and Gang Pan. Hybrid spiking vision transformer for object detection with event cameras. *arXiv preprint arXiv:2505.07715*, 2025a.

Qi Xu, Junyang Zhu, Dongdong Zhou, Hao Chen, Yang Liu, Jiangrong Shen, and Qiang Zhang. Self-cross feature based spiking neural networks for efficient few-shot learning. *arXiv preprint arXiv:2505.07921*, 2025b.

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

Man Yao, JiaKui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. *arXiv preprint arXiv:2404.03663*, 2024.

Man Yao, Xuerui Qiu, Tianxiang Hu, Jiakui Hu, Yuhong Chou, Keyu Tian, Jianxing Liao, Luziwei Leng, Bo Xu, and Guoqi Li. Scaling spike-driven transformer with efficient spike firing approximation training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

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

Kairong Yu, Tianqing Zhang, Hongwei Wang, and Qi Xu. Fsta-snn: Frequency-based spatial-temporal attention module for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 22227–22235, 2025.

Q. Yu, C. Ma, S. Song, G. Zhang, and K. C. Tan. Constructing accurate and efficient deep spiking neural networks with double-threshold and augmented schemes. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–13, 2021.

DingYi Zeng, Yuchen Wang, Honglin Cao, Wanlong Liu, Yichen Xiao, Wenyu Chen, Malu Zhang, Guoqing Wang, Yang Yang, et al. Leveraging asynchronous spiking neural networks for ultra efficient event-based visual processing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 1620–1628, 2025.

Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

Malu Zhang, Xiaoling Luo, Jibin Wu, Ammar Belatreche, Siqi Cai, Yang Yang, and Haizhou Li. Toward building human-like sequential memory using brain-inspired spiking neural models. *IEEE transactions on neural networks and learning systems*, 2025a.

Tianqing Zhang, Kairong Yu, Xian Zhong, Hongwei Wang, Qi Xu, and Qiang Zhang. Staa-snn: Spatial-temporal attention aggregator for spiking neural networks. *arXiv preprint arXiv:2503.02689*, 2025b.

Weixing Zhang, Zongrui Li, De Ma, Huajin Tang, Xudong Jiang, Qian Zheng, and Gang Pan. Spiking gs: Towards high-accuracy and low-cost surface reconstruction via spiking neuron-based gaussian splatting. *arXiv preprint arXiv:2410.07266*, 2024a.

Yuhan Zhang, Xiaode Liu, Yuanpei Chen, Weihang Peng, Yufei Guo, Xuhui Huang, and Zhe Ma. Enhancing representation of spiking neural networks via similarity-sensitive contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16926–16934, 2024b.

Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11062–11070, 2021.

Konrad Zolna, Devansh Arpit, Dendi Suhubdy, and Yoshua Bengio. Fraternal dropout. *arXiv preprint arXiv:1711.00066*, 2017.

Shihao Zou, Yuxuan Mu, Xinxin Zuo, Sen Wang, and Li Cheng. Event-based human pose tracking by spiking spatiotemporal transformer, 2023.

A Implementation Details

For the static datasets (CIFAR10, CIFAR100, and ImageNet), we applied data normalization to ensure that the images have a mean of 0 and a variance of 1. We also employed random horizontal flipping and random cropping as data augmentation techniques. For a fair comparison, the images were encoded into binary spikes using the first layer of the SNN, following recent works Rathi & Roy (2020); Fang et al. (2021b;a). Additionally, AutoAugment Cubuk et al. (2018) was utilized for further data augmentation on CIFAR-10(100), as done in previous studies Guo et al. (2022b); Li et al. (2021b). In the case of the CIFAR10-DVS dataset, we split it into 9k training images and 1k test images, as outlined in Wu et al. (2019). The training image frames were resized to 48×48, as in Zheng et al. (2021), with random horizontal flipping and random pixel shifting (within a 5-pixel range) applied during training. The test images were resized to 48×48 without any additional processing, following the methodology in Li et al. (2021b).

B Results on CIFAR10-DVS

Table 3: Comparison with SoTA methods on CIFAR10-DVS.

Method	Type	Architecture	Timestep	Accuracy
Ensemble learning Ding et al. (2025)	SNN training	ResNet18	10	70.03%
STBP-tdBN Zheng et al. (2021)	SNN training	ResNet19	10	67.80%
STAA-SNN Zhang et al. (2025b)	SNN training	ResNet20	16	81.90%
FSTA-SNN Yu et al. (2025)	SNN training	ResNet20	10	81.50%
Dspike Li et al. (2021b)	SNN training	ResNet18	10	75.40%
SSCL Zhang et al. (2024b)	SNN training	ResNet18	10	78.50%
SML Deng et al. (2023)	SNN training	ResNet18	10	83.19%
Our method	SNN training	ResNet18	10	**83.50%**

Additionally, we tested our method on the widely used neuromorphic dataset CIFAR10-DVS, using the same hyperparameter settings as for CIFAR. To ensure a fair comparison, we incorporated the Surrogate Module, as used in SML Deng et al. (2023). Using ResNet18, which is smaller than ResNet19 and ResNet20, our method achieved a remarkable accuracy of 83.50%. These results not only demonstrate a significant improvement over previous approaches but also validate the efficacy of our proposed method.

C Energy Consumption

We further analyze the energy consumption of DropTime to confirm that it preserves the inherent energy efficiency of SNNs. During training, DropTime and the baseline incur a comparable computational load, and thus comparable energy usage. During inference, DropTime is deactivated and introduces no dropout-like operations, so the inference procedure retains the same energy efficiency as the baseline.

To quantify this, we measure the mean activation sparsity of spiking ResNet18 over the entire CIFAR-100 test set using 2 timesteps, with and without DropTime, and estimate the corresponding energy consumption. Following Diet-SNN Rathi & Roy (2020), the energy is evaluated under 45nm CMOS technology, where a multiply-accumulate (MAC) operation costs 4.6 pJ and an accumulation costs 0.9 pJ. Based on these unit costs, we compute the energy of a single forward pass. As reported in Table 4, our network consumes 102.66 mJ per forward pass, whereas the baseline consumes 102.80 mJ, and the two activation sparsities are nearly identical (25.69% vs. 25.73%). These results indicate that DropTime hardly affects the energy consumption of SNNs.

D Ablation Study on the Retention Probability p

DropTime relies on a single hyperparameter, the retention probability p, which is fixed to $p = 0.9$ in the main paper. Here we examine how sensitive DropTime is to this choice. We evaluate spiking

Table 4: Mean activation sparsity and per-forward-pass energy consumption of spiking ResNet18 on the CIFAR-100 test set (2 timesteps), estimated under 45nm CMOS technology Rathi & Roy (2020).

Method	Sparsity	Energy
Without DropTime	25.73%	102.80 mJ
With DropTime	25.69%	102.66 mJ

ResNet18 on CIFAR-100 over $p \in \{0.75, 0.8, 0.85, 0.9, 0.95\}$ under both 2 and 4 timesteps, and report the results in Table 5.

Compared with the vanilla baseline (75.33% at 2 timesteps and 76.63% at 4 timesteps), DropTime consistently improves the accuracy across the entire range of $p \in [0.75, 0.95]$, confirming that the regularization effect of DropTime is robust to the choice of p. In particular, $p = 0.85$ and $p = 0.9$ yield relatively favorable results; we therefore adopt $p = 0.9$ throughout the main paper. These observations suggest that DropTime is insensitive to p within a broad range, and a value around 0.85–0.9 is recommended in practice.

Table 5: Ablation study on the retention probability p of DropTime with spiking ResNet18 on CIFAR-100. Reported values are top-1 accuracy (%). The best result under each timestep is highlighted in bold.

Timestep	0.75	0.80	0.85	0.90	0.95
2	75.67	75.76	75.99	**76.06**	75.34
4	77.16	77.01	76.95	**77.19**	76.73

E Train–Test Accuracy Analysis

To directly verify that DropTime alleviates the overfitting induced by increasing timesteps, we report both the training and test accuracy of spiking ResNet18 on CIFAR-100 across timesteps from 2 to 10, with and without DropTime. The results are summarized in Table 6.

For the vanilla model, the training accuracy keeps rising as the number of timesteps increases (from 85.96% to 87.81%), whereas the test accuracy peaks at 4 timesteps (76.63%) and then declines (down to 76.21% at 10 timesteps). The continually enlarging train–test gap is a clear symptom of aggravated overfitting. In contrast, DropTime suppresses the training accuracy (83.44%–85.14%) while steadily improving the test accuracy up to 77.89% at 10 timesteps, yielding a markedly smaller train–test gap. These results demonstrate that DropTime narrows the train–test accuracy gap while simultaneously increasing the test accuracy, directly evidencing the alleviation of the overfitting caused by increasing timesteps.

Table 6: Train and test accuracy (%) of spiking ResNet18 on CIFAR-100 with varying timesteps. DropTime reduces the train–test gap while improving the test accuracy.

Method		Timestep				
		2	4	6	8	10
Vanilla	Train	85.96	86.91	87.42	87.69	87.81
	Test	75.33	76.63	76.56	76.52	76.21
DropTime	Train	83.44	84.25	84.87	85.03	85.14
	Test	**76.06**	**77.19**	**77.65**	**77.83**	**77.89**

F DropTime vs. Naive Temporal Dropout

A natural question is whether the gain of DropTime comes from the idea of masking timesteps itself, or from its specific normalization. To disentangle the two, we compare DropTime against a naive temporal dropout baseline that applies standard inverted dropout to the temporal aggregation but normalizes by the *expected* number of retained timesteps pT, instead of the *actual* count $\|\boldsymbol{m}\|_1$:

$$\tilde{\boldsymbol{y}}_{\text{net}} = \frac{1}{pT}\sum_{t=1}^{T} m^t\,\boldsymbol{y}^t, \qquad \boldsymbol{y}_{\text{net}}^{\text{drop}} = \frac{1}{\|\boldsymbol{m}\|_1}\sum_{t=1}^{T} m^t\,\boldsymbol{y}^t. \tag{15}$$

Both estimators are unbiased, i.e., $\mathbb{E}[\tilde{\boldsymbol{y}}_{\text{net}}] = \mathbb{E}[\boldsymbol{y}_{\text{net}}^{\text{drop}}] = \boldsymbol{y}_{\text{net}}$, yet they differ in variance: normalizing by the random $\|\boldsymbol{m}\|_1$ keeps every mini-batch output close to a low-variance estimate of $\boldsymbol{y}_{\text{net}}$, whereas the constant pT normalization lets the output magnitude fluctuate with how many timesteps happen to be retained in that iteration.

We evaluate spiking ResNet18 on CIFAR-100 with 2 and 4 timesteps ($p = 0.9$) for the vanilla baseline, the naive temporal dropout, and DropTime. The results are reported in Table 7.

Table 7: DropTime vs. naive temporal dropout (inverted dropout with a $1/(pT)$ normalization) on spiking ResNet18 / CIFAR-100. Reported values are top-1 accuracy (%).

Method	Timestep = 2	Timestep = 4
Vanilla baseline	75.33	76.63
Naive temporal dropout	75.64	76.86
DropTime (ours)	**76.06**	**77.19**

Table 7 reveals two findings. First, both temporal-masking variants outperform the vanilla baseline (e.g., naive temporal dropout gains $+0.31\%$ at 2 timesteps and $+0.23\%$ at 4 timesteps), confirming that randomly masking timesteps is itself beneficial for mitigating the timestep-induced overfitting. Second, DropTime consistently surpasses the naive temporal dropout by a clear margin ($+0.42\%$ at 2 timesteps and $+0.33\%$ at 4 timesteps). This gap isolates the contribution of the $1/\|\boldsymbol{m}\|_1$ normalization: by rescaling with the *actual* number of retained timesteps, DropTime produces a low-variance, per-iteration unbiased estimate of $\boldsymbol{y}_{\text{net}}$ (cf. the variance argument above), which stabilizes optimization and translates into higher accuracy. In contrast, the constant $1/(pT)$ normalization of the naive variant leaves the output magnitude sensitive to how many timesteps happen to be retained in each iteration, injecting extra variance that partially offsets its regularization effect. Taken together, these results indicate that the gain of DropTime stems not only from the idea of masking timesteps, but to a comparable extent from its self-normalizing design.