TPL: Temporal Progressive Learning for Spiking Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recently, spiking neural networks (SNNs) have attracted widespread attention due to their brain-like information-process characteristics. In response to the non-differentiability of the activation function, the use of surrogate gradients makes it possible to train SNNs with backpropagation, achieving state-of-the-art performance with a small number of timesteps. Since the decision-making of SNNs relies on the accumulated outputs of multiple timesteps, which have temporal heterogeneity, the effective optimization of temporal heterogeneous outputs is crucial to the performance of SNNs. Existing research has found that the brain exhibits a progressive learning mechanism of temporal heterogeneous information. However, current training methods cannot satisfy the brain-like learning mechanism and cause temporal optimization deviation, temporal performance oscillation, and limited performance gains with increasing timestep lengths during direct training. To tackle the issues, we first construct an explicit gradient analysis framework based on the STBP gradient calculation paradigm to analyze the temporal optimization of SNNs. Then, based on the framework, we connect the brain's progressive learning mechanism to SNN, and propose the Temporal Progressive Learning (TPL) method to generate temporal heterogeneous optimization targets for temporal heterogeneous outputs, enabling the brain-like progressive learning mechanism in SNNs. We conduct experiments on static datasets including CIFAR10 and ImageNet, and a neuromorphic dataset DVS-CIFAR10. The results show that TPL exhibits effective temporal optimization, thereby improving the overall performance of SNNs. It is worth noting that our method achieves state-of-the-art accuracy on DVS-CIFAR10 and ImageNet datasets, with **88.60%** and **80.80%**, respectively. The code is available at https://anonymous.4open.science/r/TPL4SNN-ABEC.

1 Introduction

The brain is a powerful network that can self-organize and coordinate different cognitive functions. Brain-inspired computing models hold the promise of more advanced and general intelligence. As typical representatives, Spiking Neural Networks (SNNs) simulate the dynamic behavior of biological neurons by accumulating membrane potential and transmitting spikes [1, 2]. SNNs have demonstrated excellent dynamic property, spatio-temporal information processing abilities [3, 4] and low energy consumption and event-driven inference on neuromorphic hardware [5, 6], attracting widespread attention and application [7, 8, 9].

However, due to the non-differentiable of spike transmission, it remains challenging to train high-performance deep SNNs. Although ANN-SNN conversion approaches enable SNNs to achieve comparable performance as source ANNs, such approaches usually require a large number of timesteps, ignore the temporal dynamics of SNNs, and cannot be applied to neuromorphic datasets

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

[10, 11, 12]. On the other hand, to further exploit the performance of SNNs that is different from ANNs, recent research shows that the standard direct training (SDT) approach based on STBP with surrogate gradients can be used to train SNNs, which is more biologically interpretable than the conversion-based approaches, making SNNs achieve state-of-the-art performance with a small number of timesteps [13, 14, 15].

The decision-making of SNNs relies on the outputs over multiple timesteps [16, 17]. Neurons' dynamic behaviors in SNNs make the temporal outputs have different distributions, known as temporal heterogeneity [18, 19]. [20] has theoretically and experimentally demonstrated that the temporal heterogeneity characteristic is crucial to the performance of SNNs. In direct training, the optimization of the temporal heterogeneous outputs determines the performance of SNNs, prompting us to explore an effective optimization method. Inspired by neuroscience, existing research has found that the brain exhibits a progressive learning mechanism of temporal heterogeneous information [21, 22]. Specifically, this mechanism includes two processes: **1)** firstly, it accumulates the obtained temporal heterogeneous information; **2)** then it adjusts the accumulated information by enhancing the strength of task-relevant evidence and reducing that of task-irrelevant evidence.

However, for the temporal optimization in SNNs, most existing training approaches [23, 24, 25] completely ignore the temporal heterogeneity characteristic and don't possess the progressive learning mechanism. Although [26] has been aware of the temporal heterogeneity characteristic and incorporates contrastive supervision signal with temporal outputs, it still does not satisfy the progressive learning mechanism. Therefore, the above training approaches only make empirical improvements, limiting the representation ability of SNNs. In addition, there is a lack of an explicit gradient analysis framework to analyze the temporal optimization of SNNs.

In this work, we first construct an explicit gradient analysis framework based on the STBP gradient calculation paradigm [13] to analyze the temporal optimization of SNNs. Based on the framework, by connecting the brain's progressive learning to SNNs, we derive the properties that the temporal optimization of SNNs should satisfy and reveal the existing classic direct training approaches have the problems of temporal optimization deviation, temporal performance oscillation, and limited performance gains with increasing timestep lengths during direct training. Furthermore, based on the analysis, we propose the Temporal Progressive Learning (TPL) method to generate temporal heterogeneous optimization targets for temporal heterogeneous outputs, enabling the brain-like progressive learning mechanism in SNNs. Our experiments demonstrate that the proposed method exhibits more effective and stable performance during direct training, further bringing effective performance gains and improving the overall performance of SNNs. Figure A.1 depicts the workflow of TPL. The main contributions of this work are as follows:

- We first construct an explicit gradient analysis framework based on STBP gradient calculation paradigm [13]. Based on this framework, by connecting the brain's progressive learning mechanism to SNNs, we derive the properties that the temporal optimization of SNNs should satisfy and reveal the limitations of existing direct training approaches.
- We propose the Temporal Progressive Learning (TPL) method to generate temporal heterogeneous optimization targets for temporal heterogeneous outputs, enabling the brain-like progressive learning mechanism in SNNs.
- We conducte extensive experiments on the static datasets CIFAR10 and ImageNet and a neuromorphic dataset DVS-CIFAR10. The results demonstrate the biological plausibility and superior performance of our method. In particular, our method achieves state-of-the-art accuracy on DVS-CIFAR10 and ImageNet datasets, with **88.60%** and **80.80%**, respectively.

2 Related Work

2.1 Learning Methods of Spiking Neural Networks

In the field of SNN research, the learning algorithms can be divided into three categories: 1) converting ANN to SNN (ANN2SNN) [10, 12, 27]; 2) unsupervised learning [28]; and 3) supervised learning [13, 14, 23]. ANN2SNN converts a special trained ANN to an SNN that yields the same input-output mapping for a given task. Some recent conversion methods have achieved nearly lossless accuracy with VGG-16 and ResNet [11, 12, 29]. However, the converted SNN needs a longer time to rival the original ANN in precision as the conversion is based on rate-coding [30], which

increases the SNN's latency and restricts the practical application. Unsupervised learning is based on biologically plausible local learning rules and is usually considered more biologically plausible than other categories. However, unsupervised learning can only train shallow SNNs, and is hard to achieve desirable performance. Supervised learning uses the derivable approximation to overcome the non-differentiability of the spike activities[3, 7, 13, 23]. Hence, the SNN can be optimized with gradient descent algorithms as the ANN and achieves high performance. The SNN trained by the surrogate method is not limited to rate-coding, and can also handle dynamic datasets [31, 32]. Thus, supervised learning has increasingly aroused researchers' great interest in recent years. We focus on providing some insights to improve the performance of the supervised learning-based SNNs.

2.2 Temporal Information Optimization of Spiking Neural Networks

Due to the decision-making of SNN relies on the cumulative output of multiple timesteps in the direct training, effective temporal optimization is crucial to the performance of SNN. In recent years, some research has focused on this [23, 24, 25, 26]. One line of research focuses on improving the consistency of temporal output representation. For example, TET [23] assigns the same optimization target across all timesteps. TKS [24] facilitates model training by transferring the same knowledge for different timesteps. ETC [25] proposes a constraint that enhances temporal consistency, aiming to make the distribution at each timestep as similar as possible. However, the above methods sacrifice the temporal heterogeneity characteristic, ignore the correlation between different timesteps, and thus result in the overall performance of SNN is still limited. On the other hand, TCL proposes a temporal supervised contrastive learning framework to construct diverse temporal relationships from the perspective of whether the labels of each timestep are consistent. However, this is also an empirical improvement and does not fully consider the inherent neuron dynamic characteristics of SNN. Currently, neuroscience research has made significant breakthroughs in understanding how the brain organizes and processes temporal information [18, 21, 22, 19]. These discoveries inspire us to focus on optimizing the temporal heterogeneous outputs of SNNs.

3 Method

In this section, we first construct an explicit gradient analysis framework to analyze the temporal optimization of SNNs. Based on this framework, by connecting the brain's progressive learning mechanism to SNNs, we derive the properties that the temporal optimization of SNNs should satisfy and reveal the problems existing in the current classic direct training approaches. Ultimately, we introduce our Temporal Progressive Learning (TPL) method to generate temporal heterogeneous optimization targets for temporal heterogeneous outputs, enabling the brain-like progressive learning mechanism in SNNs.

3.1 Gradient Analysis for Temporal Optimization of SNN

Based on the concept of direct training, using BackPropagation Through Time (BPTT) and surrogate gradient, the gradients of SNN can be calculated through spatial-temporal backpropagation (STBP) [13]. To further analyze the temporal optimization of SNNs, we present the gradient calculation formula in the temporal dimension:

$$\frac{\partial L}{\partial \mathbf{W}} = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{O}_t - \boldsymbol{y}_t)\frac{\partial \boldsymbol{O}_t}{\partial \mathbf{W}}, \tag{1}$$

where T is the total number of timesteps. $\boldsymbol{O}_t$ denotes the predicted probability output at timestep t by performing a softmax calculation on the actual output of SNN. $\boldsymbol{y}_t$ represents the optimization target of $\boldsymbol{O}_t$. $\mathbf{W}$ is the network trainable parameters. The temporal optimization of direct training aims to minimize the error term $(\boldsymbol{O}_t - \boldsymbol{y}_t)$ by propagating the error back to $\mathbf{W}$ based on the chain rule and calculate the update gradients of $\mathbf{W}$. Therefore, the accurate calculation of $(\boldsymbol{O}_t - \boldsymbol{y}_t)$, that is, setting the appropriate optimization targets $\boldsymbol{y}_t$ for the temporal outputs $\boldsymbol{O}_t$ to ensure that $\boldsymbol{y}_t$ is compatible with $\boldsymbol{O}_t$, can guide $\mathbf{W}$ towards optimal optimization. To do this, we first analyze the properties of $\boldsymbol{O}_t$ as follows.

In SNNs, the property of $\boldsymbol{O}_t$ is determined by the behaviors of neurons. In our work, we utilize Leaky Integrate-and-Fire (LIF) neuron as the foundational computational unit. The dynamic behavior of LIF is described as follows:

$$\boldsymbol{U}_{t+1} = \alpha \boldsymbol{U}_t + \boldsymbol{I}_{t+1}, \tag{2}$$

$$\boldsymbol{S}_{t+1} = \boldsymbol{\Theta}(\boldsymbol{U}_{t+1} - V_{th}), \tag{3}$$

$$\boldsymbol{U}_{t+1} = \boldsymbol{U}_{t+1} \cdot (1 - \boldsymbol{S}_{t+1}), \tag{4}$$

where α is the constant leaky factor, $\boldsymbol{U}_t$ is the membrane potential at timestep t, and $\boldsymbol{I}_{t+1}$ denotes the pre-synaptic inputs, which is the product of synaptic weights $\mathbf{W}$ and the inputs from the previous layer. $\boldsymbol{\Theta}$ represents the Heaviside step function. Given a specific threshold V_{th}, when $\boldsymbol{U}_t$ exceeds V_{th}, the neuron fires a spike, i.e., $\boldsymbol{S}_t$, and $\boldsymbol{U}_t$ reset to 0. The output spike $\boldsymbol{S}_{t+1}$ will become the post synaptic spike and propagate to the next layer.

Equation 2 reveals that the new membrane potential $\boldsymbol{U}_{t+1}$ is determined by $\alpha \boldsymbol{U}_t$ and $\boldsymbol{I}_{t+1}$. This temporal evolution and dependence make $\boldsymbol{U}_t$ and $\boldsymbol{S}_t$ dynamically vary over timesteps. $\boldsymbol{O}_t$ in Equation 1 is derived from the forward propagation of $\boldsymbol{U}_t$ and $\boldsymbol{S}_t$ in the network. Therefore, the distribution of $\boldsymbol{O}_t$ is variable over timesteps, that is, $\boldsymbol{O}_t$ is temporal heterogeneous as defined in [18, 19]. This indicates that to set the appropriate optimization target for $\boldsymbol{O}_t$, $\boldsymbol{y}_t$ should have different distributions at different timesteps. Next, we explore how to generate the appropriate optimization targets $\boldsymbol{y}_t$ for the temporal outputs $\boldsymbol{O}_t$.

Inspired by neuroscience, existing research has found that the brain exhibits a progressive learning mechanism for temporal heterogeneous information [18, 19]. Specifically, this mechanism includes two processes: **1)** *Temporal Heterogeneous Information Accumulation.* It accumulates the temporal heterogeneous information obtained up to the current timestep. **2)** *Task Evidence Adjustment.* It adjusts the accumulated information by enhancing the strength of task-relevant evidence and reducing that of task-irrelevant evidence.

In the following, we connect the brain's progressive learning mechanism to SNNs for supervised learning tasks and analyze the properties that the temporal optimization of SNNs with the brain-like mechanism should satisfy.

Connect the brain-like mechanism to SNNs. 1) *Temporal Heterogeneous Information Accumulation in the SNNs.* we express the accumulated temporal heterogeneous outputs up to timestep t in SNNs as $\boldsymbol{O}_{AT,t} = \sum_{i=1}^{t} \boldsymbol{O}_i$. To ensure that $\boldsymbol{O}_{AT,t}$ is within the same range for different time lengths, which is beneficial to subsequent analysis and use, we calculate the mean of $\sum_{i=1}^{t} \boldsymbol{O}_i$, and the final expression is $\boldsymbol{O}_{AT,t} = \frac{1}{t} \sum_{i=1}^{t} \boldsymbol{O}_i$. **2)** *Task Evidence Adjustment in the SNNs.* In the distribution of $\boldsymbol{O}_{AT,t}$, we define the strength of task-relevant evidence as the probability of the target class in $\boldsymbol{O}_{AT,t}$, denoted as P_t^{target}, and it should increase as timestep t increases. The strength of task-irrelevant evidence is represented by the sum of the probabilities of the non-target classes in $\boldsymbol{O}_{AT,t}$, denoted as P_t^{others}, which should decrease as timestep t increases.

The properties in the SNNs with the brain-like mechanism. Based on the entropy calculation theory [33], the above dynamic adjustment of P_t^{target} and P_t^{others} in $\boldsymbol{O}_{AT,t}$ can be quantified by the entropy of $\boldsymbol{O}_{AT,t}$, which is denotes as $E(\boldsymbol{O}_{AT,t})$. Specifically, when P_t^{target} is small and P_t^{others} is large, $E(\boldsymbol{O}_{AT,t})$ is high, which is beneficial for exploration. Conversely, when P_t^{target} is large and P_t^{others} is small, $E(\boldsymbol{O}_{AT,t})$ is low, contributing to more convergence. Therefore, as timestep t increases, $E(\boldsymbol{O}_{AT,t})$ should decrease. Through mathematical derivations and analysis, we can further have the conclusion that $E(\boldsymbol{O}_t)$ decreases as timestep t increases (details see Appendix A.4). Moreover, for supervised learning tasks, the variation of $\boldsymbol{O}_t$ is dependent on the guidance of $\boldsymbol{y}_t$, and we can have the conclusion that the entropy of $\boldsymbol{y}_t$, i.e., $E(\boldsymbol{y}_t)$, should decrease as timestep t increases.

We introduce the classic direct training approaches SDT [14] and TET [23] into the proposed gradient analysis framework. Furthermore, we derive the issues in their temporal optimization and highlight the significance of the properties in the SNNs with the brain-like learning mechanism.

First, based on the gradient calculation formula of SDT (details see Appendix A.5), $\boldsymbol{y}_{\text{SDT},t}$ can be expressed as:

$$\boldsymbol{y}_{\text{SDT},t} = T \cdot \boldsymbol{y} - \sum_{t'=1, t' \neq t}^{T} \boldsymbol{O}_{t'}, \tag{5}$$

where $\boldsymbol{y}$ represents the true label, which is fixed across timesteps. $\sum_{t'=1,t'\neq t}^{T} \boldsymbol{O}_{t'}$ is the sum of the predicted probabilities at other timesteps, which is a constant at the current optimization timestep t. Equation 5 indicates that $\boldsymbol{y}_{\text{SDT},t}$ is heterogeneous between different timesteps. However, there are two problems: **1)** *Temporal Optimization Deviation.* It is important to note that $\boldsymbol{y}_{\text{SDT},t}$ is not directly related to $\boldsymbol{O}_t$, which means $\boldsymbol{y}_{\text{SDT},t}$ is not compatible with $\boldsymbol{O}_t$, causing temporal optimization deviation. **2)** *Temporal performance Oscillation.* The term $(\sum_{t'=1,t'\neq t}^{T} \boldsymbol{O}_{t'})$ in $\boldsymbol{y}_{\text{SDT},t}$ varies significantly at different timesteps, particularly during the initial training stage. According to the gradient coherence theory [34], if there is not effective control of $(\sum_{t'=1,t'\neq t}^{T} \boldsymbol{O}_{t'})$, $E(\boldsymbol{y}_{\text{SDT},t})$ can not decrease as timestep t increases, which means the optimization directions guided by $\boldsymbol{y}_{\text{SDT},t}$ at different timsteps are inconsistent, causing temporal performance oscillation. We further conduct experiments to validate these analytical conclusions (details see Section 4.2).

Then, based on the gradient calculation formula of TET (details see Appendix A.5), $\boldsymbol{y}_{\text{TET},t}$ can be expressed as:

$$\boldsymbol{y}_{\text{TET},t} = \boldsymbol{y}, \tag{6}$$

which implies that the temporal heterogeneous outputs share the same fixed optimization target, i.e., the true label $\boldsymbol{y}$ and $E(\boldsymbol{y}_{\text{TET},t})$ keeps the same across all timesteps. Although this leads to homogeneous and stable temporal optimization, it does not show performance variation under different timesteps, causing the performance gains with increasing timestep lengths in SNNs to be limited. Experimental results also prove the above analysis (details see Section 4.2).

Most importantly, in the SNNs with the brain-like learning mechanism, the properties of $\boldsymbol{y}_t$ are beneficial to address the temporal optimization issues. Specifically, **1)** $\boldsymbol{y}_t$ is generated based on the accumulated temporal heterogeneous outputs up to the current timestep, ensuring $\boldsymbol{y}_t$ is compatible with $\boldsymbol{O}_t$. **2)** The decrease of $E(\boldsymbol{y}_t)$ over timesteps ensures the temporal optimization is convergent and stable. **3)** The temporal heterogeneity of $E(\boldsymbol{y}_t)$ enables the SNNs to have variable representations at different time scales, increasing the performance gains with increasing timestep lengths. We validate the superiority of TPL over SDT and TET in Section 4.

3.2 Temporal Progressive Learning

Based on the above analysis, under the guidance of the properties that $\boldsymbol{y}_t$ should satisfy in the SNNs with the brain-like learning mechanism, we propose the Temporal Progressive learning (TPL) method. TPL involves a new kind of loss function L_{TPL} to generate the temporal heterogeneous optimization target $\boldsymbol{y}_{\text{TPL},t}$ for the temporal heterogeneous output $\boldsymbol{O}_{\text{TPL},t}$, enabling the brain-like progressive learning mechanism in SNNs.

Firstly, since in neuroscience, the signal-to-noise ratio is a key factor in measuring the cognitive level of the brain [21, 22], we introduce the ratio between P_t^{target} and P_t^{others} of $\boldsymbol{y}_t$ to define the magnitude relationship between them as $\beta_t = \frac{P_t^{\text{target}}}{P_t^{\text{others}}}$. It should increase as timestep t increases.

However, the true label $\boldsymbol{y}$ is the only optimization target provided in supervised learning tasks. Since it is one-hot encoded and keeps fixed over timesteps, without any prior knowledge, it is not possible to directly adjust $\boldsymbol{y}$ to generate $\boldsymbol{y}_t$ that satisfies the variation of β_t and is compatible with $\boldsymbol{O}_t$. Inspired by the brain-like learning mechanism and fuzzy logic optimization theory [35], a more promising approach is to use the accumulated temporal heterogeneous outputs as the prior knowledge to regulate the distribution of $\boldsymbol{y}_t$. To do this, we design a new loss term $\theta_t^{\lambda_t}$ and obtain the loss function of TPL as follows:

$$L_{TPL,t} = L_{CE}(\boldsymbol{O}_t, \boldsymbol{y}) + \theta_t^{\lambda_t}, \tag{7}$$

$$\theta_t = \frac{P_t^{\text{target}}}{P_t^{\text{others}}}, \quad \lambda_t = \frac{T-t}{T}, \tag{8}$$

$$P_t^{\text{target}} = \frac{exp(Q_t^i/\tau)}{\sum_j^C exp(Q_t^j/\tau)}, \quad P_t^{\text{others}} = \frac{\sum_{k=1,k\neq i}^{C} exp(Q_t^k/\tau)}{\sum_j^C exp(Q_t^j/\tau)}, \tag{9}$$

$$\boldsymbol{Q}_t = \frac{1}{t}\sum_{i=1}^{t} \boldsymbol{O}_t, \tag{10}$$

where we use $\boldsymbol{Q}_t$ in Equation 10 to calculate the average accumulated temporal heterogeneous outputs up to the current timestep t ($t \in \{1, 2, ..., T\}$). Then, the probability distribution of ($\boldsymbol{Q}_t$ / τ) after softmax is decoupled into $\boldsymbol{P}_t^{\text{target}}$ and $\boldsymbol{P}_t^{\text{others}}$ in Equation 9, where C is the number of classes, i corresponds to the target class. We introduce the temperature parameter τ to control the smoothness of $\boldsymbol{Q}_t$, which is more conducive to learning the relationships between different classes. In particular, a larger value of τ leads to a more uniform distribution, indicating smaller differences in distribution among various classes. Conversely, a smaller value of τ will sharpen the distribution, resulting in the target class having notably higher values than the others. In Equation 8, θ_t measures the ratio between $\boldsymbol{P}_t^{\text{target}}$ and $\boldsymbol{P}_t^{\text{others}}$. In addition, a linearly decreasing function λ_t is utilized to control the preference level of $\theta_t^{\lambda_t}$ in the loss term of TPL. Specifically, $\theta_t^{\lambda_t}$ decreases as timestep t increases, and the optimization of TPL is increasingly determined by the CE term.

In summary, the loss function $L_{TPL,t}$ consists of two components: cross entropy loss $L_{CE}(\boldsymbol{O}_t, \boldsymbol{y})$ and the new term $\theta_t^{\lambda_t}$. $\boldsymbol{y}_{\text{TPL},t}$ is generated by the true label $\boldsymbol{y}$ and the new term $\theta_t^{\lambda_t}$, where $\theta_t^{\lambda_t}$ ensures that the probabilities of different classes in $\boldsymbol{y}_{\text{TPL},t}$ are all non-zero, enabling $\boldsymbol{y}_{\text{TPL},t}$ to vary over timesteps in a manner compatible with $\boldsymbol{O}_t$.

For the gradient calculation of TPL, we partition the overall gradient backpropagation into two components: gradient backpropagation originating from the target class and the non-target classes. Since the gradient updates in opposite directions between these two components, we focus on the gradient calculation for the target class i (details see Appendix A.5). Based on the gradient calculation formula of the target class i, we can derive $\boldsymbol{y}_{\text{TPL},t}^i$ can be expressed as:

$$y_{\text{TPL},t}^i = y^i - \lambda_t \theta_t^{\lambda_t - 1} \cdot \frac{1}{t\tau} \cdot P_t^i \tag{11}$$

Let $g(t) = \lambda_t \theta_t^{\lambda_t - 1} \frac{1}{t\tau} P_t^i$, we can indicate the variation of $y_{\text{TPL},t}^i$ by analyzing how $g(t)$ varies as timestep t increases. To this end, we further calculate the derivative of $g(t)$ with respect to timestep t as follows:

$$g'(t) = -(\frac{P_t^i}{1 - P_t^i})^{(\frac{T-t}{T})} \cdot (1 - P_t^i) \cdot \frac{1}{t\tau} \cdot [1 + \frac{T-t}{T} \cdot \ln(\frac{P_t^i}{1 - P_t^i}) + \frac{T-t}{T} \cdot \frac{1}{t}]. \tag{12}$$

After analyzing each element in $g'(t)$, we conclude that $g'(t) < 0$ (details see Appendix A.5), which means $g(t)$ is monotonically decreasing as timesteps t increases. Therefore, over timesteps, $y_{\text{TPL},t}^i$ is monotonically increasing, while the sum of probabilities of non-target classes is monotonically decreasing. According to [33], the entropy of the overall probability distribution of $\boldsymbol{y}_{\text{TPL},t}$, i.e., $E(\boldsymbol{y}_{\text{TPL},t})$, decreases as timestep t increases. We further conduct sufficient experimental verification on our TPL method, which can be seen in Section 4.

4 Experiments

We validate the effectiveness of our proposed method and compare it with existing works for classification tasks on static datasets, including CIFAR10, CIFAR100, ImageNet, and a neuromorphic dataset DVS-CIFAR10. The network architectures in this paper include ResNet-19, VGG-SNN, and Meta-SpikeFormer. We employ these architectures with surrogate gradient function from [23], and train them using TPL from scratch. More details of the configurations can be found in the Appendix.

4.1 Comparison with Exiting Works

CIFAR10. As shown in Table 1, we conduct experiments on CIFAR10 dataset based on ResNet-19 under simulation lengths of 2, 4, and 6, respectively. We report the mean and standard deviation of 3 runs under different random seeds. The temperature parameter τ is set to 2. On CIFAR10, our method achieves an accuracy of 95.34 % at the simulation length of 6, achieving competitive performance compared with other SOTA algorithms.

DVS-CIFAR10. Neuromorphic datasets are typically recorded using event-based cameras, effectively capturing object changes and usually exhibiting strong temporal dependencies. DVS-CIFAR10 is

Table 1: Compare with existing state-of-the-art methods.

Dataset	Model	Architecture	Simulation Length	Accuracy
CIFAR10	DSpike[38]	ResNet-18	6	94.25
	MLF[39]	ResNet-19	4	94.25
	GLIF[40]	ResNet-19	6	95.03
	RecDis-SNN[41]	ResNet-19	6	95.55
	TEBN[15]	ResNet-19	6	95.60
	IM-Loss[42]	ResNet-19	6	95.49
	RMP-Loss[43]	ResNet-19	6	96.10
	InfLoR-SNN[44]	ResNet-19	6	96.49
	LSG[45]	ResNet-19	6	95.52
	TKS[24]	ResNet-19	4	95.30
	TCL[26]	ResNet-19	4	95.03
	TET[23]	ResNet-19	6	94.50
	ETC[25]	SEW-ResNet-18	6	95.73
	TPL	ResNet-19	6	**95.34±0.23**
			4	**95.17±0.17**
			2	**94.98±0.20**
DVS-CIFAR10	DSpike[38]	ResNet-18	10	75.40
	MLF[39]	ResNet-19	10	70.36
	GLIF[40]	Wide 7B Net	16	78.10
	RecDis-SNN[41]	ResNet-19	10	72.42
	TEBN[15]	VGG-SNN	10	84.90
	IM-Loss[42]	ResNet-19	10	72.60
	RMP-Loss[43]	ResNet-19	10	76.20
	InfLoR-SNN[44]	ResNet-19	10	75.50
	LSG[45]	ResNet-19	10	77.90
	TET[23]	VGG-SNN	10	83.17
	TKS[24]	VGG-SNN	10	85.30
	TCL[26]	VGG-SNN	4	79.10
	ETC[25]	VGG-SNN	10	85.35
	TPL	VGG-SNN	10	**88.60±0.25**
ImageNet	SEW ResNet[4]	SEW-ResNet-34	4	67.04
	Diet-SNN[46]	VGG-16	5	69.00
	DSpike[38]	VGG-16	5	71.24
	GLIF[40]	ResNet-34	6	69.09
	RecDis-SNN[41]	ResNet-34	6	67.33
	TEBN[15]	SEW ResNet-34	4	68.28
	IM-Loss[42]	VGG-16	5	70.65
	RMP-Loss[43]	ResNet-34	4	65.17
	TKS[24]	SEW-ResNet-34	4	69.60
	TET[23]	Spiking-ResNet-34	6	64.79
	ETC[25]	Spiking-ResNet-34	6	69.64
	Spike-Driven Transformer V2[47]	Meta-SpikeFormer	4	80.00
	TPL	Meta-SpikeFormer	4	**80.80**

the most challenging mainstream neuromorphic dataset. Here, we adopt VGG-SNN on the DVS-CIFAR10 dataset, set temperature parameter $\tau = 2$, and report the mean and standard deviation of 3 runs under different random seeds. As shown in Table 1, along with data augmentation methods suitable for SNN [36], our method achieved the highest accuracy at 88.60% on DVS-CIFAR10, outperforming existing SOTA by 3.25%. Our TPL method achieves notably superior performance on DVS-CIFAR10 due to the diverse input-output pairs, which is beneficial to improve the performance [37]. The DVS dataset provides diverse input images at different timesteps, and the proposed TPL designs heterogeneous optimization targets $\boldsymbol{y}_{\mathrm{TPL},t}$ for $\boldsymbol{O}_t$, enhancing the diversity of input-output pairs and improving the performance of SNNs.

ImageNet. ImageNet dataset is one of computer vision's most widely used and challenging datasets. We choose Meta-SpikeFormer obtaining SOTA results in SNNs to verify our method on ImageNet with temperature parameter $\tau = 1$. As shown in Table 1, our method further improves the current SOTA performance in the SNN domain from 80.00% to 80.80%, with significant accuracy advantages compared to other existing approaches.

4.2 Temporal Optimization Issues of Existing Methods

SDT. Firure1 (a) shows the convergence curves of SDT at each timestep on CIFAR10 with a simulation length of 6. It is obvious that the loss curves oscillate and rise with the training process. The phenomenon proves the conclusion of our analysis in Section 3.1: SDT has the issues of temporal optimization deviation and temporal performance oscillation. Firure1 (d) shows the convergence curves of SDT for different accumulated timesteps (referred to as "AT" in the legend). It can be observed that there is little variation in the convergence loss values across different cumulative timesteps, indicating that increasing the timestep lengths of the SNN does not lead to significant

performance gains. In summary, the above problems will lead to the limitation of SNN's overall performance.

TET. Figure 1 (b) illustrates the loss curves at each timestep, and Figure 1 (e) visualizes the loss curves for different accumulated timesteps of TET on CIFAR10 with a simulation length of 6. It can be observed that compared to SDT, TET exhibits greater stability during training, which can be attributed to the use of the same optimization target for temporal outputs. However, TET does not show performance variation under different accumulated timestep lengths, which means the performance gains introduced by increasing the timestep lengths of SNN are limited. In Section 4.3, we further validate this phenomenon by analyzing the similarity between different temporal outputs and the time scalability in Figure 2 and Figure 3, respectively. These findings align with our analysis in Section 3.1, suggesting that TET neglects the heterogeneity and correlation of temporal information, thereby limiting the temporal performance of SNN.

4.3 Model Validation and Ablation Study

In this section, we demonstrate that the SNNs trained by our proposed TPL method can capture and process heterogeneous outputs at multiple time scales, and produce a brain-like learning mechanism. Moreover, we verify the robustness of our method within a reasonable variation range of parameters.

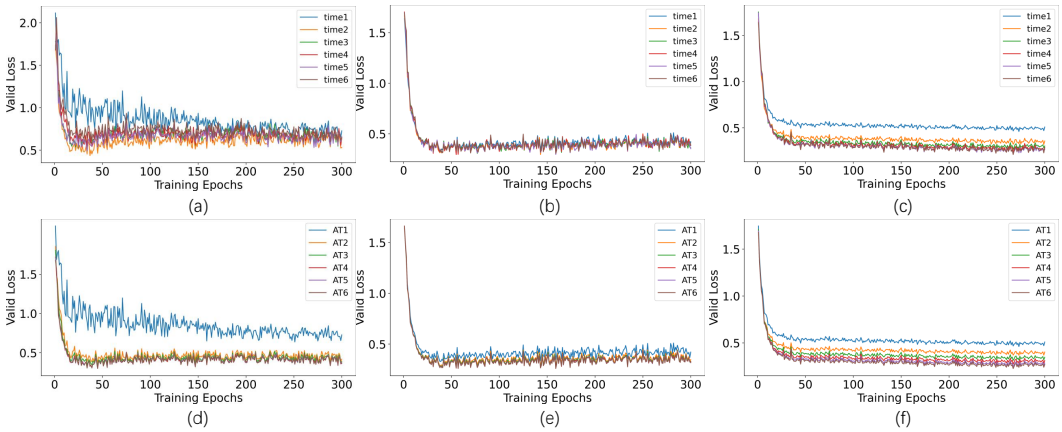

Figure 1: Loss convergence curves of SDT, TET and TPL.

Temporal Heterogeneity and Stability. Figure 2 presents the similarity matrices of the trained SNNs (T=4) using TET and TPL. Each element in the matrices represents the Kullback-Leibler (KL) divergence between the outputs at two timesteps. As shown in Figure 2, a value of the Kullback-Leibler (KL) divergence closer to 0 indicates greater homogeneity in TET. Conversely, a value further from 0 indicates higher heterogeneity in TPL. Compared to TET, TPL shows higher heterogeneity because TPL generates the temporal heterogeneous optimization target $\boldsymbol{y}_{\mathrm{TPL},t}$ for the temporal heterogeneous output $\boldsymbol{O}_{\mathrm{TPL},t}$. Next, we visualize the convergence curve at each timestep in Figure1 (c) on the CIFAR10 validation dataset with a total timestep of 6. The convergence curve of each timestep is more stable than that of SDT and TET. This indicates that our method can achieve both heterogeneity and stationarity in the temporal optimization.

Progressive Learning in Multiple Time Scales. We further inspect the temporal optimization effect of the proposed method on multiple time scales. As shown in Figure1 (f), we visualize the convergence curves with different accumulated timesteps on the CIFAR10 validation dataset with a total timestep length of 6. We can see that our method can converge stably under different accumulated timesteps, and the loss values decrease with the increase of the accumulated timesteps. This proves that TPL has the progressive learning capability for temporal heterogeneous outputs and further improves the performance of SNN.

Time Scalability. In addition to performance, latency is a crucial factor that constrains the development of SNNs. In this part, we demonstrate the time scalability of TPL and the advantages it brings on CIFAR10. During the training phase on CIFAR10, we set the simulation length to 6, while in the testing phase, we evaluated the performance across different timesteps. As shown in Figure 3, with

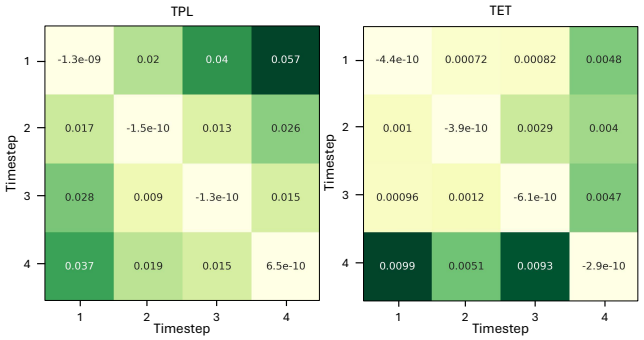

Figure 2: Temporal heterogeneity comparison.

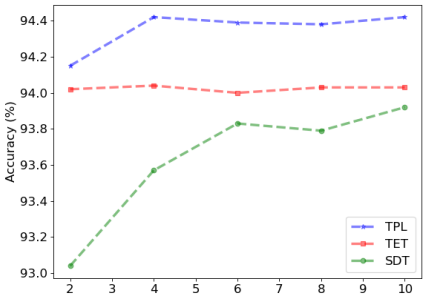

Figure 3: Time scalability verification.

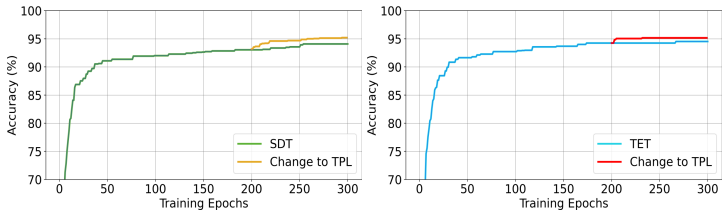

Figure 4: TPL helps further optimization of temporal information.

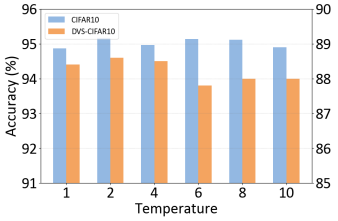

Figure 5: Effect influence of τ.

a simulation length of 2, our TPL has surpassed the accuracy of SDT and TET at a step size of 10. Besides, the accuracy of our method can maintain a stable effect at different timesteps, significantly reducing the network's latency.

Fine-Tuning Optimization Capabilities. We further validate the ability of the proposed method to alleviate the problem of insufficient temporal optimization in the existing methods SDT and TET. We adopt the ResNet-19 with a total simulation length of 6 on CIFAR10. First, we optimize L_{SDT} and L_{TET} for 200 epochs and then change the loss function to L_{TPL} after epoch 200. Figure 4 demonstrates the accuracy change on the validation set. After 200 epochs of training, L_{SDT} and L_{TET} converge to the local minimum, and the accuracy increases slowly. Nevertheless, after we change the loss function to L_{TPL}, the accuracy of SNN has been further improved, which is 0.6% higher than SDT and 0.4% higher than TET. This phenomenon illustrates that the L_{TPL} has the ability to further fully optimize the temporal outputs and bring out the better performance of SNN.

Influence of Hyperparameter τ. We analyze the sensitivity of TPL to the temperature parameter τ on CIFAR10 and DVS-CIFAR10 datasets. Figure 5 reports the accuracy of TPL across a range of τ values from 1 to 10, demonstrating the robustness of TPL to variations in τ. Notably, when $\tau > 4$ and $\tau < 2$, the performance of TPL has a little decrease. This is because higher τ smoothes the distribution of $\boldsymbol{y}_{\text{TPL},t}$, causing slower convergence, while lower τ sharpens the distribution of $\boldsymbol{y}_{\text{TPL},t}$ and increases the probability of the target class, making the temporal optimization effect of TPL is close to TET. However, these issues do not cause significant performance decreases. TPL maintains high accuracy with a minimum of 94.87% at $\tau = 1$ and a maximum of 95.17% at $\tau = 2$ on CIFAR10 and a minimum of 87.80% at $\tau = 6$ and a maximum of 88.60% at $\tau = 2$ on DVS-CIFAR10.

5 Conclusion and Limitations

In conclusion, this work first develops an explicit gradient analysis framework, connects the brain's progressive learning mechanism to SNNs, and proposes the Temporal Progressive Learning (TPL) method. TPL can generate temporal heterogeneous optimization targets for temporal heterogeneous outputs, enabling the brain-like progressive learning mechanism in SNNs. Extensive evaluation demonstrates the superiority of TPL, especially on DVS-CIFAR10. However, it is important to validate TPL on a broader range of time-related application datasets. In addition, we should include an in-depth analysis of the computational and training costs, as well as the convergence speed of TPL, to assess the applicability of our method in real-world scenarios.

6 Broader Impact

SNNs exhibit brain-like information processing capabilities, and effectively optimizing temporal information is vital for further developing the performance of SNNs. The proposed temporal progressive learning (TPL) method in our work replicates the brain's progressive learning of temporal heterogeneous information in SNNs by generating adaptive optimization targets. The potential impacts of this work are several-fold:

1) Precision: TPL offers superior precision on popular datasets, which demonstrates the effectiveness of the proposed method. Notably, it achieves state-of-the-art accuracy on the DVS-CIFAR10 and ImageNet datasets, with impressive results of 88.60% and 80.80%, respectively.

2) Low latency: TPL delivers high-precision training over a very short temporal window of a few timesteps. This is in contrast with many BP methods that require hundreds of timesteps for maintaining decent accuracy. Low latency computation immediately corresponds to fast decisions.

3) Community impact: Moreover, the explicit gradient analysis framework and TPL approach will be made publicly available. This will facilitate their adoption by brain-inspired computing researchers, promoting further advancements in the field.

References

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

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

[3] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J Thorpe, and Timothée Masquelier. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018.

[4] 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, 2021.

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

[6] 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.

[7] Jiaqi Yan, Qianhui Liu, Malu Zhang, Lang Feng, De Ma, Haizhou Li, and Gang Pan. Efficient spiking neural network design via neural architecture search. *Neural Networks*, page 106172, 2024.

[8] Qianhui Liu, Jiaqi Yan, Malu Zhang, Gang Pan, and Haizhou Li. Lite-snn: Designing lightweight and efficient spiking neural network through spatial-temporal compressive network search and joint optimization. *arXiv preprint arXiv:2401.14652*, 2024.

[9] Zhanfeng Liao, Yan Liu, Qian Zheng, and Gang Pan. Spiking nerf: Representing the real-world geometry by a discontinuous representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13790–13798, 2024.

[10] Yangfan Hu, Qian Zheng, Xudong Jiang, and Gang Pan. Fast-snn: Fast spiking neural network by converting quantized ann. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[11] 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*, pages 6316–6325. PMLR, 2021.

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

[13] Yujie Wu, Lei Deng, Guoqi Li, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:323875, 2018.

[14] 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, pages 11062–11070, 2021.

[15] Chaoteng Duan, Jianhao Ding, Shiyan Chen, Zhaofei Yu, and Tiejun Huang. Temporal effective batch normalization in spiking neural networks. *Advances in Neural Information Processing Systems*, 35:34377–34390, 2022.

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

[17] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11270–11277, 2020.

[18] Nicolas Perez-Nieves, Vincent CH Leung, Pier Luigi Dragotti, and Dan FM Goodman. Neural heterogeneity promotes robust learning. *Nature communications*, 12(1):5791, 2021.

[19] Hanle Zheng, Zhong Zheng, Rui Hu, Bo Xiao, Yujie Wu, Fangwen Yu, Xue Liu, Guoqi Li, and Lei Deng. Temporal dendritic heterogeneity incorporated with spiking neural networks for learning multi-timescale dynamics. *Nature Communications*, 15(1):277, 2024.

[20] Richard Gast, Sara A Solla, and Ann Kennedy. Neural heterogeneity controls computations in spiking neural networks. *Proceedings of the National Academy of Sciences*, 121(3):e2311885121, 2024.

[21] Dante Francisco Wasmuht, Eelke Spaak, Timothy J Buschman, Earl K Miller, and Mark G Stokes. Intrinsic neuronal dynamics predict distinct functional roles during working memory. *Nature communications*, 9(1):3499, 2018.

[22] Benjamin B Scott, Christine M Constantinople, Athena Akrami, Timothy D Hanks, Carlos D Brody, and David W Tank. Fronto-parietal cortical circuits encode accumulated evidence with a diversity of timescales. *Neuron*, 95(2):385–398, 2017.

[23] 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.

[24] Yiting Dong, Dongcheng Zhao, and Yi Zeng. Temporal knowledge sharing enable spiking neural network learning from past and future. *IEEE Transactions on Artificial Intelligence*, 2024.

[25] Dongcheng Zhao, Guobin Shen, Yiting Dong, Yang Li, and Yi Zeng. Improving stability and performance of spiking neural networks through enhancing temporal consistency. *arXiv preprint arXiv:2305.14174*, 2023.

[26] Haonan Qiu, Zeyin Song, Yanqi Chen, Munan Ning, Wei Fang, Tao Sun, Zhengyu Ma, Li Yuan, and Yonghong Tian. Temporal contrastive learning for spiking neural networks. *arXiv preprint arXiv:2305.13909*, 2023.

[27] 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.

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

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

[30] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:294078, 2017.

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

[32] 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*, pages 2661–2671, 2021.

[33] Michael C Mackey. The dynamic origin of increasing entropy. *Reviews of Modern Physics*, 61(4):981, 1989.

[34] Satrajit Chatterjee. Coherent gradients: An approach to understanding generalization in gradient descent-based optimization. *arXiv preprint arXiv:2002.10657*, 2020.

[35] Alejandra Mancilla, Oscar Castillo, and Mario García Valdez. Optimization of fuzzy logic controllers with distributed bio-inspired algorithms. *Recent Advances of Hybrid Intelligent Systems Based on Soft Computing*, pages 1–11, 2021.

[36] Yuhang Li, Youngeun Kim, Hyoungseob Park, Tamar Geller, and Priyadarshini Panda. Neuromorphic data augmentation for training spiking neural networks. In *European Conference on Computer Vision*, pages 631–649. Springer, 2022.

[37] Nicolangelo Iannella and Andrew D Back. A spiking neural network architecture for nonlinear function approximation. *Neural networks*, 14(6-7):933–939, 2001.

[38] 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, 2021.

[39] Lang Feng, Qianhui Liu, Huajin Tang, De Ma, and Gang Pan. Multi-level firing with spiking ds-resnet: Enabling better and deeper directly-trained spiking neural networks. *arXiv preprint arXiv:2210.06386*, 2022.

[40] Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:32160–32171, 2022.

[41] 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*, pages 326–335, 2022.

[42] Yufei Guo, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Yinglei Wang, Xuhui Huang, and Zhe Ma. Im-loss: information maximization loss for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:156–166, 2022.

[43] Yufei Guo, Xiaode Liu, Yuanpei Chen, Liwen Zhang, Weihang Peng, Yuhan Zhang, Xuhui Huang, and Zhe Ma. Rmp-loss: Regularizing membrane potential distribution for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17391–17401, 2023.

[44] 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 *European Conference on Computer Vision*, pages 36–52. Springer, 2022.

[45] Shuang Lian, Jiangrong Shen, Qianhui Liu, Ziming Wang, Rui Yan, and Huajin Tang. Learnable surrogate gradient for direct training spiking neural networks. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 3002–3010, 2023.

[46] Nitin Rathi and Kaushik Roy. Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 34(6):3174–3182, 2021.

[47] 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.

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

[49] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:244131, 2017.

[50] 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*, pages 248–255. Ieee, 2009.

A Appendix

A.1 Dataset and Training Detail

CIAFR10. CIFAR10 dataset [48] contains 10 classes and consists of 50k training images and 10k testing images with the size of 32 × 32. The original random horizontal flip and crop are applied to the training image augmentation. We train ResNet-19 on the CIFAR10 dataset under simulation lengths of 2, 4, and 6 for 300 epochs, respectively. We use an Adam optimizer with a learning rate of 0.01 and apply cosine decay, gradually reducing the learning rate to 0.

DVS-CIAFR10. DVS-CIFAR10 [49], the most challenging mainstream neuromorphic dataset, is converted from CIFAR10. It has 10k images with a size of 128×128. We apply a 9:1 train-test split (i.e., 9k training images and 1k test images). In our training, we integrate the event data into 10 frames and resize the resolution to 48 × 48. Random horizontal flip and random roll within 5 pixels are taken as augmentation [36]. We adopt VGG-SNN architecture with 300 epochs of training on this classification task. And we use an Adam optimizer with a learning rate of 0.01 and cosine decay to 0.

ImageNet. ImageNet [50] contains more than 1250k training images and 50k validation images. We crop the images to 224×224 and using the standard augmentation for the training data. We fitune the pretrained model with timestep $T = 1$ for 200 epochs to $T = 4$ using our proposed method for 20 epochs with a learning rate of $3e - 5$. The rest of the experimental settings are the same as [47].

It is worth noting that the result of our TPL method on Meta-SpikeFormer [47] is obtained through training from scratch. Meta-SpikeFormer employs T=1 during the pre-training phase and uses T=4 during the fine-tuning phase. When T=1, the loss function of our method TPL is equivalent to the loss function used by Meta-SpikeFormer. However, when T=4, the loss function of our method differs from that of Meta-SpikeFormer. Therefore, we directly fine-tuned Meta-SpikeFormer based on the pre-trained model.

A.2 Experimental Results on CIFAR100

CIAFR100. CIFAR100 has the same configurations as CIFAR10, except it contains 100 classes.

Table A.1: Compare with existing state-of-the-art methods on CIFAR100.

Dataset	Methods	Architecture	Simulation Length	Accuracy
CIFAR100	Diet-SNN	VGG-16	5	64.07
	DSpike	ResNet-18	6	74.24
	GLIF	ResNet-19	6	77.35
	RecDis-SNN	ResNet-19	4	74.10
	TEBN	ResNet-19	6	78.76
	IM-Loss	VGG-16	5	70.18
	RMP-Loss	ResNet-19	6	78.98
	InfLoR-SNN	ResNet-19	6	78.98
	LSG	ResNet-19	6	77.13
	TKS	ResNet-19	4	76.20
	TCL	ResNet-19	4	79.73
	STBP-tdBN	ResNet-19	6	71.12
			4	70.86
			2	69.41
	TET	ResNet-19	6	74.72
			4	74.47
			2	72.87
	ETC	SEW-ResNet-18	6	78.25
			4	77.65
			2	75.96
	TPL	ResNet-19	6	**78.16**
			4	**77.50**
			2	**76.50**

As shown in Table A.1, we conduct experiments on CIFAR100 dataset based on ResNet-19 under simulation lengths of 2, 4, and 6, respectively. The temperature parameter τ is set to 2. On CIFAR100,

our method achieves an accuracy of 78.16% at the simulation length of 6, achieving competitive performance compared with other SOTA approaches.

A.3 Workflow of TPL

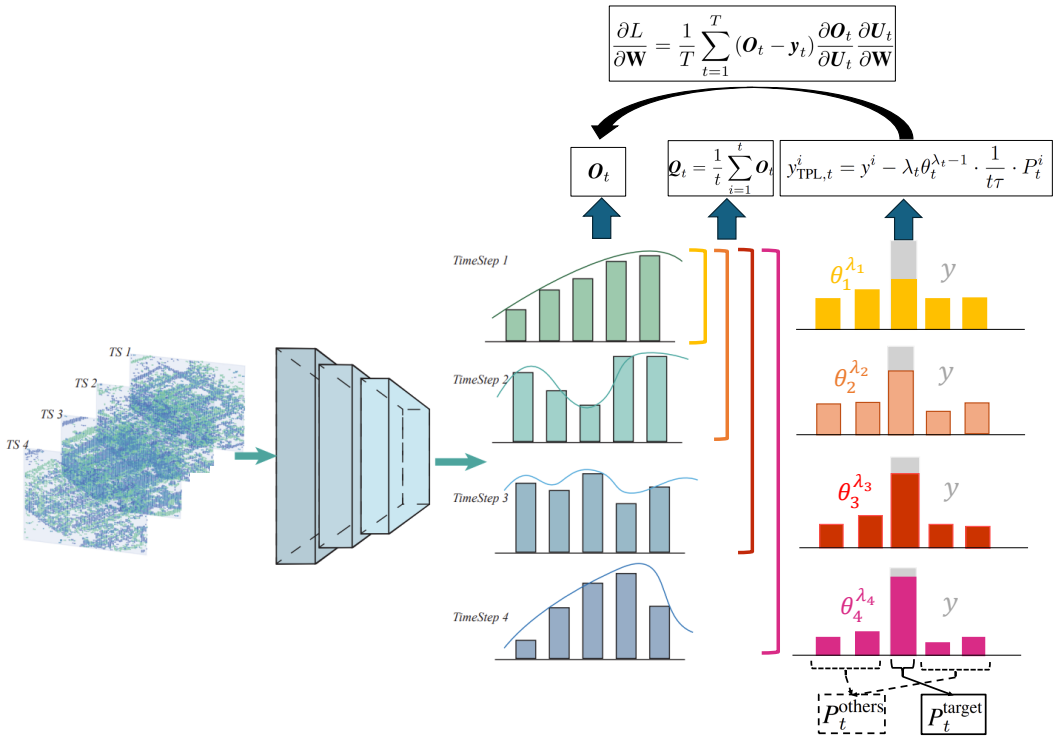

Figure A.1: Workflow of the Temporal Progressive Learning (TPL) method. We generate a temporal heterogeneous optimization target for temporal heterogeneous output.

A.4 Derivation and Analysis of the SNNs with the Brain-Like Mechanism

We provide more detailed mathematical derivations and analysis to support the conclusion in Section 3.1 that $E(\boldsymbol{y}_t)$ should gradually decrease as timestep t increases as follows.

1) Firstly, in the SNNs with the brain's progressive learning mechanism for the supervised learning tasks, we construct the expression of the accumulated temporal heterogeneous outputs $\mathbf{O}_{AT,t}$ in Equation A.1 and analyze how it varies at different timesteps s as follows:

$$\mathbf{O}_{AT,t} = \frac{1}{t}\sum_{k=1}^{t}\mathbf{O}_k. \tag{A.1}$$

According to the definition of the brain's progressive learning mechanism [18, 19], the probability of the target class i in $\mathbf{O}_{AT,t}$, denoted as $O^i_{AT,t}$ should increase as timestep t increases. This implies that:

$$O^i_{AT,t} > O^i_{AT,t-1}. \tag{A.2}$$

Simultaneously, the sum of the probabilities of the non-target classes in $\mathbf{O}_{AT,t}$, denoted as $\sum_{j=1,j\neq i}^{C} O^j_{AT,t}$, where C is the total number of classes, should decrease as timestep t increases, that is:

$$\sum_{j=1,j\neq i}^{C} O^j_{AT,t} < \sum_{j=1,j\neq i}^{C} O^j_{AT,t-1}. \tag{A.3}$$

Since $O^i_{AT,t}$ and $\sum_{j=1,j\neq i}^{C} O^j_{AT,t}$ exhibit contrasting variation trends, we will focus on the mathematical derivations and analysis related to $O^i_{AT,t}$ in the following context.

2))Secondly, based on Equation A.2, we can derive the magnitude relationship between $\mathbf{O}_{AT,t}$ and $\mathbf{O}_t$ as follows:

$$O^i_{AT,t} = \frac{1}{t}\sum_{k=1}^{t} O^i_k > O^i_{AT,t-1} = \frac{1}{t-1}\sum_{k=1}^{t-1} O^i_k. \tag{A.4}$$

$$\frac{1}{t}O^i_t + \frac{1}{t}\sum_{k=1}^{t-1} O^i_k > \frac{1}{t-1}\sum_{k=1}^{t-1} O^i_k. \tag{A.5}$$

$$O^i_t > \frac{1}{t-1}\sum_{k=1}^{t-1} O^i_k. \tag{A.6}$$

$$O^i_t > O^i_{AT,t-1}. \tag{A.7}$$

3) Furthermore, we can derive the magnitude relationship between O^i_t and O^i_{t-1} by subtracting O^i_{t-1} from each element in Equation A.7, and we have $(O^i_t - O^i_{t-1}) > (O^i_{AT,t-1} - O^i_{t-1})$. Since the result of $(O^i_{AT,t-1} - O^i_{t-1})$ indicates the range of $(O^i_t - O^i_{t-1})$, we further analyze it as follows:

$$\begin{aligned}
O^i_{AT,t-1} - O^i_{t-1} &= \frac{1}{t-1}\sum_{k=1}^{t-1} O^i_k - O^i_{t-1} \\
&= \frac{1}{t-1}\sum_{k=1}^{t-2} O^i_k + \frac{1}{t-1}O^i_{t-1} - O^i_{t-1} \\
&= \frac{1}{t-1}\sum_{k=1}^{t-2} O^i_k + \frac{2-t}{t-1}O^i_{t-1} \\
&= \frac{t-2}{t-1}O^i_{AT,t-2} + \frac{2-t}{t-1}O^i_{t-1} \\
&= \frac{2-t}{t-1}(O^i_{t-1} - O^i_{AT,t-2}).
\end{aligned} \tag{A.8}$$

According to Equation A.7, $(O^i_{t-1} - O^i_{AT,t-2}) > 0$. In addition, when $t > 1$ and $t \leq 2$, $\frac{2-t}{t-1}(O^i_{t-1} - O^i_{AT,t-2}) \geq 0$. When $t > 2$, $\frac{2-t}{t-1}(O^i_{t-1} - O^i_{AT,t-2})$ is close to 0, because $\frac{2-t}{t-1}$ and $(O^i_{t-1} - O^i_{AT,t-2})$ are very small numbers of the order of 0.1. Overall, $\frac{2-t}{t-1}(O^i_{t-1} - O^i_{AT,t-2}) \geq 0$, that is, $O^i_{AT,t-1} - O^i_{t-1} \geq 0$, and we can conclude that:

$$O^i_t - O^i_{t-1} > 0. \tag{A.9}$$

Since $\sum_{j=1,j\neq i}^{C} O^j_t = 1 - O^i_t$, we can derive that:

$$\sum_{j=1,j\neq i}^{C} O^j_t - \sum_{j=1,j\neq i}^{C} O^j_{t-1} < 0. \tag{A.10}$$

4) Therefore, Equation A.9 and Equation A.10 indicate that in the overall probability distribution of $\mathbf{O}_t$, the probability of the target class O^i_t gradually increases while the sum of probabilities of the non-target classes $\sum_{j=1,j\neq i}^{C} O^j_t$ decreases as timestep t increases. According to entropy calculation theory [33], the entropy of $\mathbf{O}_t$, denoted as $E(\mathbf{O}_t)$ should decrease as timestep t increases. Moreover, for the supervised learning tasks, the variation of $\mathbf{O}_t$ is dependent on the guidance of $\boldsymbol{y}_t$ as shown in Equation 1, which means the entropy of $\boldsymbol{y}_t$, i.e. $E(\boldsymbol{y}_t)$, should gradually decrease as timestep t increases.

A.5 Derivation of Gradient Calculation Formula

SDT. SNN calculates the standard cross-entropy (CE) loss between the average prediction probability across timesteps and the true label $\boldsymbol{y}$. The loss function of SDT L_{SDT} is:

$$L_{SDT} = L_{CE}(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{O}_t, \boldsymbol{y}). \tag{A.11}$$

Following the chain rule, the gradient calculation formula of SDT is:

$$\begin{aligned}\frac{\partial L_{SDT}}{\partial \mathbf{W}} &= \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{O}_{\text{mean}} - \boldsymbol{y})\frac{\partial \boldsymbol{O}_t}{\partial \mathbf{W}} \\ &= \frac{1}{T}\sum_{t=1}^{T}(\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{O}_t - \boldsymbol{y})\frac{\partial \boldsymbol{O}_t}{\partial \mathbf{W}} \\ &= \frac{1}{T}\sum_{t=1}^{T}\left[\frac{1}{T}\boldsymbol{O}_t - (\boldsymbol{y} - \frac{1}{T}\sum_{t'=1,t'\neq t}^{T}\boldsymbol{O}_{t'})\right]\frac{\partial \boldsymbol{O}_t}{\partial \mathbf{W}}.\end{aligned} \tag{A.12}$$

According to Equation1, we can have:

$$\boldsymbol{y}_{SDT,t} = T\cdot\boldsymbol{y} - \sum_{t'=1,t'\neq t}^{T}\boldsymbol{O}_{t'}, \tag{A.13}$$

TET. It assigns the same optimization target, i.e., the true label $\boldsymbol{y}$, across all timesteps. The loss function of TET L_{TET} is:

$$L_{TET} = \frac{1}{T}\sum_{t=1}^{T}L_{CE}(\boldsymbol{O}_t, \boldsymbol{y}). \tag{A.14}$$

Following the chain rule, the gradient calculation formula of TET is:

$$\frac{\partial L}{\partial \mathbf{W}} = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{O}_t - \boldsymbol{y}_t)\frac{\partial \boldsymbol{O}_t}{\partial \mathbf{W}}. \tag{A.15}$$

According to Equation 1, we can have:

$$\boldsymbol{y}_{\text{TET},t} = \boldsymbol{y}. \tag{A.16}$$

TPL. We partition the overall gradient backpropagation of TPL into two components: gradient backpropagation originating from the target class and the non-target classes. Based on Equation 7-10, following the chain rule, we provide the mathematical derivations and analysis of these two components, respectively, as follows.

For the gradient backpropagation originating from the target class i, the gradient calculation formula is:

$$\begin{aligned}\frac{\partial L_{TPL}}{\partial \mathbf{W}} &= \frac{\partial L_{CE}}{\partial \mathbf{W}} + \frac{\partial \theta_t^{\lambda_t}}{\partial \mathbf{W}} \\ &= \frac{\partial L_{CE}}{\partial O_t^i}\frac{\partial O_t^i}{\partial \mathbf{W}} + \frac{\partial \theta_t^{\lambda_t}}{\partial \theta_t}\frac{\partial \theta_t}{\partial P_t^i}\frac{\partial P_t^i}{\partial Q_t^i}\frac{\partial Q_t^i}{\partial O_t^i}\frac{\partial O_t^i}{\partial \mathbf{W}}.\end{aligned} \tag{A.17}$$

$$\frac{\partial \theta_t^{\lambda_t}}{\partial \mathbf{W}} = \frac{\partial L_{CE}}{\partial O_t^i}\frac{\partial O_t^i}{\partial \mathbf{W}} + \frac{\partial \theta_t^{\lambda_t}}{\partial \theta_t}\frac{\partial \theta_t}{\partial P_t^i}\frac{\partial P_t^i}{\partial Q_t^i}\frac{\partial Q_t^i}{\partial O_t^i}\frac{\partial O_t^i}{\partial \mathbf{W}}. \tag{A.18}$$

$$\frac{\partial L_{CE}}{\partial O_t^i} = O_t^i - y^i, \tag{A.19}$$

where y^i is the probability of the target class i in the true lable $\boldsymbol{y}$.

$$\frac{\partial \theta_t^{\lambda_t}}{\partial \theta_t} = \lambda_t \theta_t^{\lambda_t - 1}. \tag{A.20}$$

$$\frac{\partial \theta_t}{\partial P_t^i} = \frac{1}{P_t^{\text{others}}}. \tag{A.21}$$

$$\frac{\partial P_t^i}{\partial Q_t^i} = \frac{exp(Q_t^i/\tau)(1/\tau)}{\sum_j^C exp(Q_t^j/\tau)} - \frac{exp(Q_t^i/\tau)^2(1/\tau)}{[\sum_j^C exp(Q_t^j/\tau)]^2}, \tag{A.22}$$

where C is the total number of classes.

$$\frac{\partial Q_t^k}{\partial O_t^k} = \frac{1}{t}. \tag{A.23}$$

Then, we have:

$$\begin{aligned}
\frac{\partial L_{TPL}}{\partial \mathbf{W}} &= (O_t^i - y^i)\frac{\partial O_t^i}{\partial \mathbf{W}} + \lambda_t \theta_t^{\lambda_t - 1} \cdot \frac{1}{P_t^{\text{others}}} \cdot \big[\frac{exp(Q_t^i/\tau)(1/\tau)}{\sum_j^C exp(Q_t^j/\tau)} - \frac{exp(Q_t^i/\tau)^2(1/\tau)}{(\sum_j^C exp(Q_t^j/\tau))^2}\big] \cdot \frac{1}{t} \cdot \frac{\partial O_t^i}{\partial \mathbf{W}} \\
&= \left[(O_t^i - y^i) + \lambda_t \theta_t^{\lambda_t - 1} \cdot \frac{1}{P_t^{\text{others}}} \cdot \frac{1}{t\tau} \cdot \big[P_t^i - (P_t^i)^2\big] \cdot \right] \frac{\partial O_t^i}{\partial \mathbf{W}} \\
&= \left[O_t^i - \big[y^i - \lambda_t \theta_t^{\lambda_t - 1} \cdot \frac{1}{t\tau} \cdot \frac{P_t^i}{P_t^{\text{others}}} \cdot \big(1 - P_t^i\big)\big]\right] \frac{\partial O_t^i}{\partial \mathbf{W}} \\
&= \left[O_t^i - \big(y^i - \lambda_t \theta_t^{\lambda_t - 1} \cdot \frac{1}{t\tau} \cdot P_t^i\big)\right] \frac{\partial O_t^i}{\partial \mathbf{W}}
\end{aligned} \tag{A.24}$$

Finally, we have:

$$y_{\text{TPL},t}^i = y^i - \lambda_t \theta_t^{\lambda_t - 1} \frac{1}{t\tau} P_t^i. \tag{A.25}$$

Let $g(t) = \lambda_t \theta_t^{\lambda_t - 1} \frac{1}{t\tau} P_t^i$, we can indicate the variation of $y_{\text{TPL},t}^i$ by analyzing how $g(t)$ varies as timestep increases. To this end, we further calculate the derivative of $g(t)$ with respect to timestep t as follows:

$$\begin{aligned}
g'(t) &= -(\frac{P_t^k}{1 - P_t^k})^{(\frac{T-t}{T})} \cdot (1 - P_t^k) \cdot \frac{1}{t\tau} - \frac{T-t}{T} \cdot (\frac{P_t^k}{1 - P_t^k})^{(\frac{T-t}{T})} \cdot \ln(\frac{P_t^k}{1 - P_t^k}) \cdot (1 - P_t^k) \cdot \frac{1}{t\tau} \\
&\quad - \frac{T-t}{T}(\frac{P_t^k}{1 - P_t^k})^{(\frac{T-t}{T})} \cdot (1 - P_t^k) \cdot \frac{1}{t^2\tau} \\
&= -(\frac{P_t^i}{1 - P_t^i})^{(\frac{T-t}{T})} \cdot (1 - P_t^i) \cdot \frac{1}{t\tau}[1 + \frac{T-t}{T} \cdot \ln(\frac{P_t^i}{1 - P_t^i}) + \frac{T-t}{T} \cdot \frac{1}{t}].
\end{aligned} \tag{A.26}$$

Due to that $[(\frac{P_t^i}{1-P_t^i})^{(\frac{T-t}{T})}(1 - P_t^i)\frac{1}{t\tau}] > 0$, we let $h(t) = [1 + \frac{T-t}{T} \cdot \ln(\frac{P_t^i}{1-P_t^i}) + \frac{T-t}{T} \cdot \frac{1}{t}]$. Since $\ln(\frac{P_t^i}{1-P_t^i})$ exhibits different trends in the two cases of $P_t^i \geq 0.5$ and $P_t^i < 0.5$. We next analyze the value of $h(t)$ in these two cases.

1) When $P_t^i \geq 0.5$, each term in $h(t)$ is greater than 0, and $h(t) > 0$.

2) When $P_t^i < 0.5$, we employ the enumeration approach and substitute $T \in \{2, 4, 6\}$ and $t \in \{1, 2, ..., T\}$ into $h(t)$. Then We derive that for any T and t, when $P_t^i \geq 0.15$, $h(t) > 0$. When $P_t^k < 0.15$, according to Equation A.25, $y_{\text{TPL},t}^i$ would be close to y^i, which will optimize P_t^i to be greater than 0.15 and $h(t) > 0$.

In summary, based on the above analysis, we can conclude that as the optimization continues, $h(t)$ will be greater than 0, resulting in $g'(t) < 0$. Therefore, in Equation A.26, $g(t)$ is monotonically decreasing as timesteps t increases, so that $y^i_{\text{TPL},t}$ is monotonically increasing. In addition, since $y^i_{\text{TPL},t} = 1 - \sum_{j=1,j\neq i}^{C}$, $\sum_{j=1,j\neq i}^{C}$ is monotonically decreasing as timesteps t increases. According to entropy calculation theory [32], the entropy of the overall probability distribution of $y_{\text{TPL},t}$, i.e., $E(y_{\text{TPL},t})$, decreases as timestep t increases.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction of the paper present the main claims and contributions of our paper.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We have discussed the limitations of our work in the conclusion.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

Justification: We have conducted a detailed analysis of the theoretical derivations in the paper.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

Answer: [Yes]

Justification: The paper fully discloses all the information needed to reproduce the main experimental results.

Guidelines:

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

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

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

Answer: [Yes]

Justification: We provide open access to the data and code, and introduce how to faithfully reproduce the main experimental results, as described in supplemental material.

Guidelines:

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

6. **Experimental Setting/Details**

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

Answer: [Yes]

Justification: The paper specifies all the training and test details necessary to understand the results.

Guidelines:

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

7. **Experiment Statistical Significance**

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

Answer: [Yes]

Justification: We have reported the mean and standard deviation of multiple runs.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: Relevant information is written in the Appendix.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: The research conducted in the paper conform with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: We have discussed both potential positive societal impacts and negative societal impacts of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper poses no such risks

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The creators or original owners of assets (such as code, data, models) used in a paper are properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

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

    Answer: [Yes]

    Justification: The paper does not release new assets.

    Guidelines:

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

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

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

    Answer: [NA]

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

    Guidelines:

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

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

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

    Answer: [NA]

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

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.