RETHINKING SPIKING NEURAL NETWORKS FROM AN ENSEMBLE LEARNING PERSPECTIVE

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs) have gained widespread attention for their low power consumption and spatio-temporal dynamics. In this paper, we consider an SNN as an ensemble of multiple temporal subnetworks that share architecture and weights, but produce different outputs due to differences in initial states (neuron membrane potentials). We identify a key factor influencing ensemble performance: excessive differences in the initial state lead to unstable subnetwork outputs that degrade performance, especially in the first two timesteps. To mitigate this, we propose to adaptively smooth the membrane potential cross adjacent timesteps to reduce the initial state discrepancy. Membrane potential smoothing allows for more stable ensemble effects and brings an additional bonus: additional pathways for forward propagation of information and backward propagation of gradients, mitigating temporal gradient vanishing and thus improving performance. Furthermore, we propose the temporally adjacent subnetwork guidance to improve the output consistency of subnetworks through distillation, further enhancing the ensemble stability and performance. Extensive experiments have shown that promoting stability yields consistent performance gains for VGG, ResNet, Transformer, and RNN architectures. Compared to existing methods, our method shows superior performance in neuromorphic/static object/gesture/speech recognition, object detection, and 3D point cloud classification tasks, achieving 80.60% accuracy on the CIFAR10-DVS dataset with only 5 timesteps.

1 INTRODUCTION

As the third generation of neural networks, spiking neural networks (SNNs) transmit discrete spikes between neurons and function over multiple timesteps Maass (1997). Benefiting from the low power consumption and spatio-temporal feature extraction capability, SNNs have achieved widespread applications in spatio-temporal tasks Wang & Yu (2024); Chakraborty et al. (2024). In particular, ultra-low latency and low-power inference can be achieved when SNNs is integrated with neuromorphic sensors and neuromorphic chips Yao et al. (2023); Ding et al. (2024).

To advance the performance of SNNs, previous work has improved the training method Wu et al. (2018); Bu et al. (2022); Zuo et al. (2024b), network architecture Yao et al. (2023); Shi et al. (2024), and neuron dynamics Taylor et al. (2023); Fang et al. (2021b); Ding et al. (2023) to significantly reduce the performance gap between SNNs and artificial neural networks (ANNs). Typically, these methods treat the spiking neurons in an SNN as an activation function that evolves over timesteps T, with the membrane potential expressing a continuous neuronal state. In this paper, we seek to rethink the spatio-temporal dynamics of SNNs from an alternative perspective: ensemble learning, and explore the key factor influencing the ensemble to optimize its performance.

For an SNN $f(\theta)$, its instance $f(\theta)_t$ produces an output O_t at each timestep t, we consider this instance to be a temporal subnetwork. In this way, we can obtain a collection of T temporal subnetworks with the same architecture $f(\cdot)$ and parameters θ: $\{f(\theta)_1, f(\theta)_2, \cdots, f(\theta)_T\}$. In general, the final output of the SNN is the average of the outputs over T timesteps: $O = \frac{1}{T}\sum_{t=1}^{T} O_t$. We view this averaging operation as an ensemble strategy: averaging the different outputs of multiple models improves overall performance Rokach (2010); Allen-Zhu & Li (2020). From this perspective, we can consider an SNN as an ensemble of T temporal subnetworks. These subnetworks share architecture and parameters, and their output variance arises from their different neuronal states,

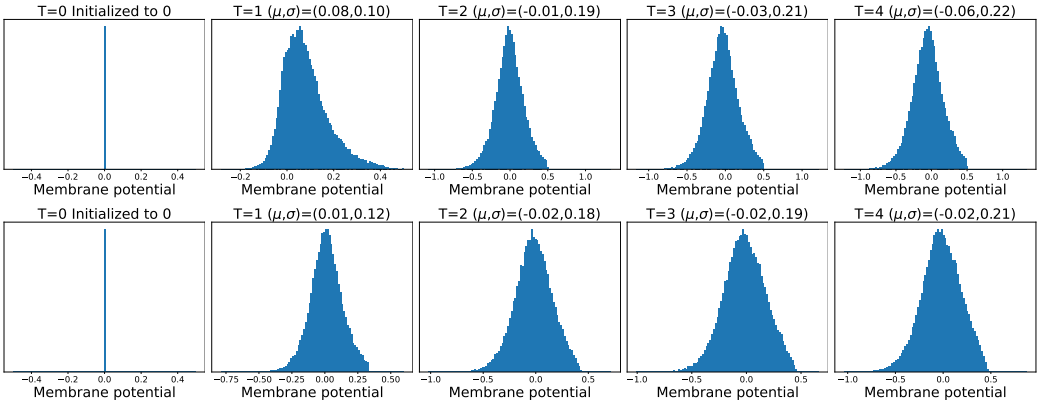


Figure 1: Membrane potential distribution in spiking VGG-9 on CIFAR10-DVS ((μ, σ) denotes the mean and standard deviation of the distribution). **Top**: The membrane potential distribution of the vanilla SNN varies greatly across timesteps, which affects performance. **Bottom**: Our method allows for a more stable distribution with smaller differences across timesteps. See Appendix A.4 for more visualization comparisons.

i.e., different membrane potentials $U(t)$ at each timestep trigger different output spikes $S(t)$. For example, for a leaky integrate-and-fire (LIF) neuron Wu et al. (2018) (see Section 3.1) with an initial membrane potential of 0 and a time constant and firing threshold of 2.0 and 1.0, respectively, the neuron will produce spikes $\{0, 1, 1, 0, 1\}$ even if inputs of intensity 0.8 are repeated over 5 timesteps. In light of this, we have identified a factor that is usually overlooked, but has a major impact on the performance of this temporal subnetwork ensemble learning: *if the difference in membrane potentials across timesteps is too large, it will lead to unstable outputs and thus affect the ensemble performance.* In particular, the initial membrane potential is usually set to 0 Wu et al. (2018); Ding et al. (2024), which leads to a drastic discrepancy in the membrane potential for the first two timesteps, thus degrading the performance of the SNN. To illustrate this phenomenon, we have visualized the membrane potential distribution in Fig 1(Top), where the distribution differences across timesteps can be clearly seen. In Table 1, we further explore the performance of the trained SNN for inference with 1 to 5 timesteps. The results show that the output of the vanilla SNN is poorly informative at the first timestep and only achieves decent performance after integrating subsequent temporal subnetworks with smaller differences in the membrane potential distribution. Additionally, the output is visualized in Fig. 4, again showing that the first two timestep outputs are confusing and thus affect the overall output. Therefore, to improve overall performance, the problem of excessive difference in membrane potential distribution and the resulting output should be mitigated rather than simply ignored.

Table 1: Comparison of SNNs with 1 to 5 inference timesteps.

	T=1	T=2	T=3	T=4	T=5
Vanilla SNN	10.00	60.10	69.50	73.30	74.10
Random MP	11.90	61.50	71.40	72.30	74.90
Ours	66.60	74.30	75.50	75.70	76.60

To this end, we propose membrane potential smoothing to reduce the difference in membrane potential distribution and thus improve the overall ensemble performance. At each timestep, we adaptively smooth the membrane potentials using that of the previous timestep, pushing the initial states of these temporal subnetworks more consistent and preventing them from producing outputs with excessive variances. We visualize the smoothed membrane potential in Fig. 1(Bottem), where the obtained membrane potential is shown to be smoother than the vanilla SNN. Meanwhile, membrane potential smoothing creates new pathways for forward propagation of information and backward propagation of gradients (see Fig. 2), alleviating temporal gradient vanishing Meng et al. (2023); Huang et al. (2024) and boosting the performance from another side. In addition, we propose to guide temporally adjacent subnetworks through distillation, encouraging them to produce more consistent outputs, further enhancing ensemble stability and performance. Our method is compatible with VGG, ResNet, Spiking Transformer, and RNN architectures with great versatility. In addition, extensive experiments have demonstrated the performance advantages of our method. Our contribution can be summarized as follows:

- We consider the SNN as an ensemble of multiple temporal subnetworks and point out that excessive differences in membrane potential distributions across timesteps, and hence output instability, is the key factor affecting performance.
- We propose membrane potential smoothing and temporally adjacent subnetwork guidance to adaptively reduce membrane potential differences and enhance output consistency across timesteps, respectively, while facilitating the propagation of forward information and backward gradients to improve ensemble stability and overall performance.
- Extensive experiments on neuromorphic/static object/gesture/speech recognition, object detection, and 3D point cloud classification tasks confirm the effectiveness, versatility, and performance advantages of our method. With only 5 timesteps, we achieved 80.60% accuracy on the challenging CIFAR10-DVS dataset.

2 Related Work

2.1 Spiking Neural Network

Existing methods for training SNNs avoid the non-differentiability of the spiking neurons either by converting a pre-trained ANN Rueckauer et al. (2017) or by using the surrogate gradient for direct training Wu et al. (2018). Conversion-based methods require large latencies and struggle with the temporal properties of SNNs Deng & Gu (2021); Bu et al. (2022); kang you et al. (2024), the surrogate gradient-based methods are widely used as they can achieve decent performance with smaller latencies Taylor et al. (2023); Zuo et al. (2024a); Hu et al. (2024). In addition to training methods, previous work has focused on improving network architectures and spiking neuron dynamics, such as the Spiking Transformer architecture Yao et al. (2023); Shi et al. (2024), the ternary spike Guo et al. (2024) and the attention spiking neuron Ding et al. (2023). Compared to existing methods, we rethink the spatio-temporal dynamics of the SNN from the perspective of ensemble learning, identify the key factor affecting its performance: the excessive difference in membrane potential distribution, and propose solutions. Our solutions do not modify the core philosophies of these existing methods and are therefore compatible with a wide range of architectures and neuron types, and integration with existing methods can further unleash the potential of SNNs.

2.2 Ensemble Learning

Model ensemble aggregates the predicted outputs of multiple trained neural networks to produce a final output that can significantly improve the performance of a deep learning model Rokach (2010); Allen-Zhu & Li (2020). To reduce ensemble overhead, some methods use a backbone network and multiple heads to produce multiple outputs Tran et al. (2020); Ruan et al. (2023), or use checkpoints during training for the ensemble Furlanello et al. (2018); Lee et al. (2022). In the field of SNNs, previous studies have ensembled multiple SNN models to improve performance without optimizing the ensemble overhead Neculae et al. (2021); Elbrecht et al. (2020); Panda et al. (2017); Hussaini et al. (2023). In this paper, we consider each timestep SNN instance as a temporal subnetwork and treat the entire SNN as an ensemble, thus avoiding additional ensemble overhead. These temporal subnetworks share the same architecture and parameters, and their output differences arise from the membrane potential state of the spiking neurons within them prior to charging. A previous study Ren et al. (2023) attributed the effectiveness of SNNs in static point cloud classification to the ensemble effect, without further analysis. Instead, we point out the key factor influencing the ensemble performance: excessive differences in membrane potential distributions can lead to unstable outputs of these subnetworks, and propose solutions to mitigate this problem, thereby improving the performance.

2.3 Temporal Consistency in SNNs

Previous studies have shown that promoting temporal consistency can improve the performance of SNNs, such as distillation Zuo et al. (2024a); Dong et al. (2024) and contrastive learning Qiu et al. (2024) in the temporal dimension. However, existing methods directly promote output/feature consistency, similar to ANNs, without adequately considering the properties of SNNs. In contrast to existing methods, this paper highlights the negative impact of differences in membrane potential distributions across timesteps from an ensemble perspective and proposes to improve distribution consistency. Compared to output/feature consistency, membrane potential distribution consistency

offers significant performance gains and can be combined with them to synergistically maximize performance.

3 METHOD

In this section, we describe the temporal dynamics of the SNN from the basic LIF neuron model and interpret it as the ensemble of multiple temporal subnetworks. We then show that excessive differences in membrane potentials across timesteps affect ensemble performance, and we mitigate this problem by adaptively smoothing membrane potentials. In addition, we encourage temporally adjacent subnetworks to produce stable and consistent outputs through distillation, further facilitating ensemble stability and performance.

3.1 SNN AS AN ENSEMBLE OF TEMPORAL SUBNETWORKS

SNNs transmit information by generating binary spikes from spiking neurons. In this paper, we use the most commonly used LIF neuron model Wu et al. (2018). LIF neurons continuously receive inputs from presynaptic neurons accumulating membrane potential H, and a spike S is fired and resets the membrane potential when it reaches the firing threshold ϑ. The LIF neuron dynamics can be expressed as:

$$H_i^l(t) = U_i^l(t) + I_i^l(t) = \left(1 - \frac{1}{\tau}\right) H_i^l(t-1) + I_i^l(t), \text{charge} \tag{1}$$

$$S_i^l(t) = \begin{cases} 1, & H_i^l(t) \geq \vartheta \\ 0, & H_i^l(t) < \vartheta \end{cases}, \text{fire spike} \tag{2}$$

$$H_i^l(t) = H_i^l(t) - S_i^l(t)\vartheta, \text{reset} \tag{3}$$

where l, i, and t denote the layer, neuron, and timestep indexes, respectively, and $I_i^l(t) = \sum_j W_{i,j}^l S_j^{l-1}(t)$ is the cumulative current of the previous layer's neuron outputs and weights. τ is the time constant that controls the decay of the membrane potential with time. $U_i^l(t)$ is the initial membrane potential at timestep t.

From Eq. 1, we can see that the output $S_i^l(t)$ of the neuron at timestep t depends on its initial membrane potential $U_i^l(t)$. The membrane potential evolves continuously, so the SNN output varies from timestep to timestep. Established methods recklessly compute the average over T timesteps outputs $O = \frac{1}{T}\sum_{t=1}^{T} O_t$ without considering the rationale behind it, leading to suboptimal performance, whereas we explore this from an ensemble perspective to improve the performance of SNNs.

We consider each timestep of the SNN as a temporal subnetwork sharing the architecture $f(\cdot)$ and the parameter θ, and different subnetworks produce different outputs O_t arising from distinct neuron membrane potentials $U(t)$. When the membrane potential difference of these subnetworks is small, their outputs vary slightly, and the ensemble can promote the generalizability of the SNN. However, excessive differences in membrane potentials can lead to drastically different outputs from these subnetworks and degrade the SNN. Unfortunately, vanilla SNNs seem to suffer from this degradation, especially since the initial membrane potential is usually set to 0 , the membrane potentials of the first two timesteps show drastic differences, see Fig. 1. This membrane potential discrepancy leads to highly unstable outputs across timesteps, which increases the optimization difficulty and degrades the ensemble performance. In addition, we show the performance of the trained SNN with different inference timesteps in Table 1, and the results show that the output of the vanilla SNN at the first timestep is not discriminative at all.

3.2 MEMBRANE POTENTIAL SMOOTHING

To mitigate the above degradation problem, a natural solution is to randomly initialize the membrane potential to reduce the membrane potential difference for the first two timesteps, similar to Ren et al. (2023). However, the experiments we show in Table 1 indicate that this practice (Random MP) only slightly alleviates the problem and still struggles to achieve satisfactory performance. To this end, we propose membrane potential smoothing, which mitigates degradation by adaptively reducing the membrane potential difference between adjacent timesteps using a learnable smoothing coefficient.

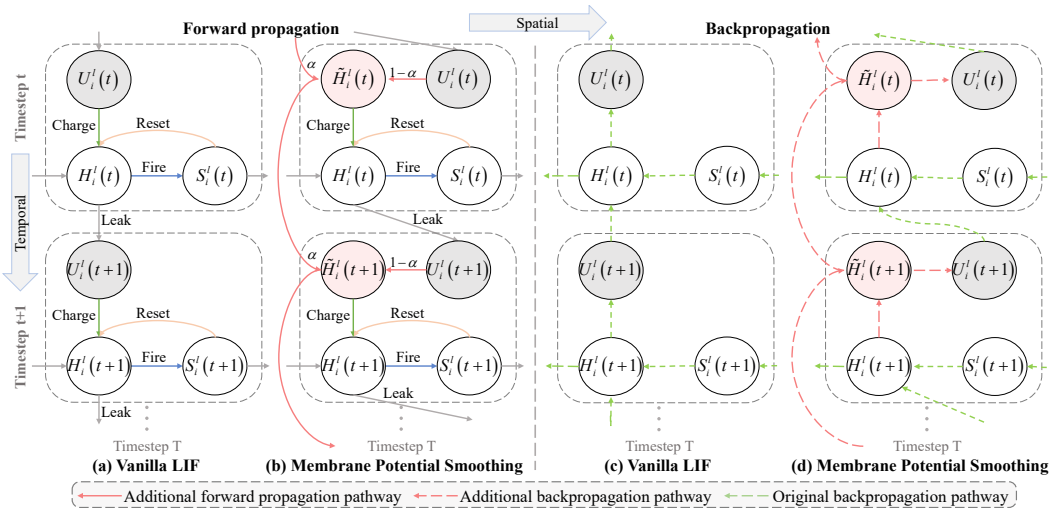


Figure 2: Illustration of (a) the vanilla LIF neuron and (b) the membrane potential smoothing. We smooth the membrane potential at timestep t using the layer-shared coefficient α^l and the smoothed membrane potential $\tilde{H}_i^l(t-1)$ at timestep $t-1$ to reduce membrane potential differences and create additional information/gradient propagation pathways.

At each timestep, we consider the initial membrane potential state $U_i^l(t)$ of the spiking neuron as the initial state of the corresponding subnetwork. We argue that by allowing these subnetworks to have similar initial states, the output spikes generated after receiving the input current will also be similar, resulting in stable SNN outputs (The input currents typically follow the same distribution due to the normalization layer). Therefore, we weight the current initial state by the smoothed state of the previous timestep with a layer-shared smoothing coefficient α^l to reduce the difference between the two. The smoothed membrane potential $\tilde{H}_i^l(t)$ receives the input current from the previous layer, which then generates spikes and resets the membrane potential, iterating to the next timestep. The membrane potential smoothing and the charge dynamics of a spiking neuron can be expressed as:

$$\tilde{H}_i^l(t) = \alpha^l \tilde{H}_i^l(t-1) + (1-\alpha^l) U_i^l(t), \text{smoothing} \tag{4}$$

$$H_i^l(t) = \tilde{H}_i^l(t) + I_i^l(t). \text{charge} \tag{5}$$

The smoothing coefficient α^l and the parameter θ of the SNN are co-optimized during training to achieve the optimal smoothing effect. To ensure that $\alpha^l \in (0,1)$, in the practical implementation we train the parameter β^l and let $\alpha^l = sigmoid(\beta^l)$. By default, β^l is initialized to 0, i.e. the initial value of α^l is 0.5. In Section 4.3, we will analyze the influence of the initial value on the performance and convergence. Since the spike activity (Eq. 2) is not differentiable, we use the rectangular function Wu et al. (2018) to calculate the spike derivative:

$$\frac{\partial S_i^l(t)}{\partial H_i^l(t)} \approx \frac{\partial h(H_i^l(t), \vartheta)}{\partial H_i^l(t)} = \frac{1}{a} \text{sign}(|H_i^l(t) - \vartheta| < \frac{a}{2}), \tag{6}$$

where a is the hyperparameter that controls the shape of the rectangular function and is set to 1.0. Accordingly, the derivative of a spike with respect to α^l can be calculated as:

$$\frac{\partial S_i^l(t)}{\partial \alpha^l} = \frac{\partial S_i^l(t)}{\partial H_i^l(t)} \frac{\partial H_i^l(t)}{\partial \tilde{H}_i^l(t)} \frac{\partial \tilde{H}_i^l(t)}{\partial \alpha^l} \approx \frac{1}{a} \text{sign}(|H_i^l(t) - \vartheta| < \frac{a}{2}) (\tilde{H}_i^l(t-1) - U_i^l(t)). \tag{7}$$

The derivative of the loss function $\mathcal{L}$ with respect to α^l can be calculated as:

$$\frac{\partial \mathcal{L}}{\partial \alpha^l} = \sum_t^T \sum_i^{n_l} \frac{\partial \mathcal{L}}{\partial S_i^l(t)} \frac{\partial S_i^l(t)}{\partial \alpha^l}, \tag{8}$$

where n_l is the number of neurons in layer l.

It is worth noting that in addition to mitigating the difference in membrane potential distribution, membrane potential smoothing can also facilitate the propagation of the gradient in the temporal

dimension. Previous studies have shown that in SNNs, the temporal gradient is a small percentage Meng et al. (2023) and is highly susceptible to gradient vanishing, leading to performance degradation Huang et al. (2024). As shown in Fig. 2, our method establishes a forward information transfer pathway from $\tilde{H}_i^l(t-1)$ to $\tilde{H}_i^l(t)$, and also propagates the error gradient in the backward direction, thus mitigating the influence of the temporal gradient vanishing. From another perspective, the additional pathways can be viewed as residual connections in the temporal dimension, facilitating the propagation of information and gradients.

Notably, membrane potential smoothing is integrated with LIF neurons in this paper (see Appendix A.1 for complete neuron dynamics), but this smoothing method is not limited to specific neuron types and can be integrated with other pre-existing neurons to further improve performance (See Appendix A.8).

3.3 Temporally Adjacent Subnetwork Guidance

Membrane potential smoothing aims to pull together the initial states of the subnetworks. In addition, we propose the temporally adjacent subnetwork guidance to further promote the output stability of these subnetworks and improve the ensemble performance.

Inspired by knowledge distillation Hinton (2015), we guide the output by identifying the "teacher" and "student" from two temporally adjacent subnetworks. Since spiking neurons need to accumulate membrane potentials before they can produce stable spiking outputs, we treat the early timestep subnetwork as a weak "student" that is guided by the stable "teacher" with a later timestep. Taking the t-th and $t+1$-th subnetworks as an example, the logits are first calculated based on the outputs O_t and O_{t+1} of the two subnetworks, respectively:

$$p(t) = \frac{e^{O_{t,j}/T_{KL}}}{\sum_{c=1}^{C} e^{O_{t,c}/T_{KL}}}, p(t+1) = \frac{e^{O_{t+1,j}/T_{KL}}}{\sum_{c=1}^{C} e^{O_{t+1,c}/T_{KL}}}, \tag{9}$$

where C denotes the C-way classification task, the subscript j indicates the j-th class, and T_{KL} is the temperature hyperparameter set to 2. We then use KL divergence to encourage the logits of subnetwork t to be as similar as possible to the logits of subnetwork $t+1$ (For regression tasks that output continuous predicted values, we directly compute the output MSE as in Appendix A.6):

$$\mathcal{L}_t = {T_{KL}}^2 KL(p(t+1)||p(t)) = {T_{KL}}^2 \sum_{j=1}^{C} p(t+1)_j log(\frac{p(t+1)_j}{p(t)_j}). \tag{10}$$

By performing guidance between each pair of temporally adjacent subnetworks, we obtain a total of $T-1$ guidance losses: $\{\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_{T-1}\}$. Instead of accumulating all these losses directly, we keep the largest one and drop the others with a probability P, as in Sariyildiz et al. (2024). The rationale behind this is that as the SNN is trained, the outputs of adjacent timesteps may already be quite similar and it is unnecessary to align them completely. In this paper, P is set to 0.5.

We define the function that selects the largest loss and randomly discards the others as $drop(\cdot)$ (see Appendix A.2 for pseudocode). During training, $drop(\{\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_{T-1}\})$ and cross-entropy loss $\mathcal{L}_{CE}$ synergistically train the SNN:

$$\mathcal{L}_{total} = \gamma \mathcal{L}_{guidance} + \mathcal{L}_{CE} = \gamma drop(\{\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_{T-1}\}) + \mathcal{L}_{CE}, \tag{11}$$

where γ is the coefficient for controlling the guidance loss, which is set to 1.0 by default.

In this way, we synergistically increase the stability of the ensemble at both the level of the initial state (membrane potential) and the output of the subnetwork, greatly improving the overall performance of the SNN. We show the pseudocode for the training process of the temporally adjacent subnetwork guidance in Algorithm 1.

4 Experiments

We perform experiments on three benchmark neuromorphic datasets: CIFAR10-DVS Li et al. (2017), DVS-Gesture Amir et al. (2017), and N-Caltech101 Orchard et al. (2015). To demonstrate the versatility of our method, the experiments use the VGG, ResNet, and SpikingResformer Shi et al.

Algorithm 1 Temporally adjacent subnetwork guidance for SNNs

Input: input data x, label Y.
Parameter: timestep T, Guidance loss coefficient γ.
Output: Trained SNN.

Initialize SNN $f(\cdot)$ with parameters θ
// Optimization over multiple iterations
for $i = 1, 2, \cdots, i_{train}$ iterations **do**
  for $t = 1, 2, \cdots, T$ **do**
    $O_t = f(\theta; x(t))$; // Calculate the output at timestep t
    if $t > 1$ **then**
      $\mathcal{L}_{t-1} = \mathrm{KL}(O_{t-1}; O_t)$; // Calculate the guidance loss by Eq. 10
    end if
  end for
  $O = \frac{1}{T}\sum_{t=1}^{T} O_t$; // Calculate the ensemble average output
  $\mathcal{L}_{CE} = \text{Cross-entropy}(O, Y)$; // Calculate the cross-entropy loss
  $\mathcal{L}_{guidance} = drop(\{\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_{T-1}\})$; // Random drop guidance loss
  $\mathcal{L}_{total} = \gamma\mathcal{L}_{guidance} + \mathcal{L}_{CE}$; // Calculate total loss by Eq. 11
  Backpropagation and optimize model parameters θ;
end for
return Trained SNN.

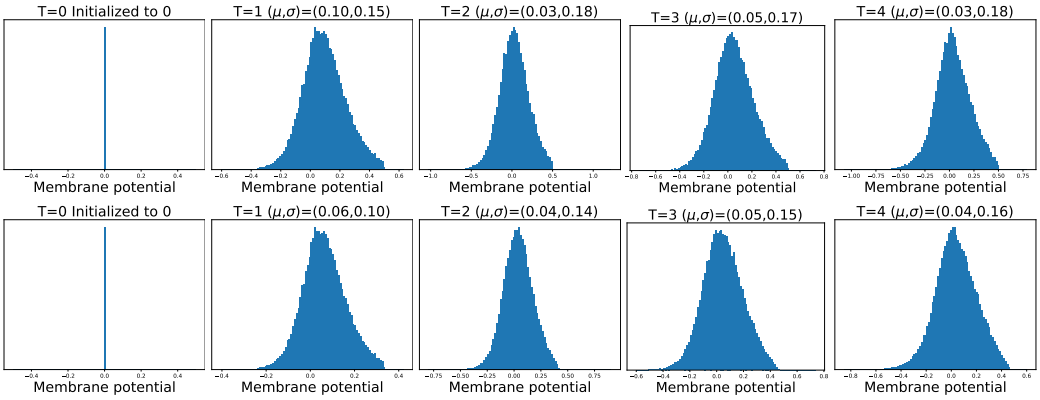


Figure 3: Visualization of the membrane potential distribution before (top) and after (bottom) smoothing. Smoothing reduces distribution differences, especially for $T1 \rightarrow T2 \rightarrow T3$.

(2024) architectures. In addition, we conduct experiments on neuromorphic speech recognition, static object recognition/detection, and 3D point cloud classification. Please see the Appendix A.3 for detailed experimental setup.

4.1 Ablation Study

The ablation study results of our method on the three architectures are shown in Table 2. The results show that our method is effective in improving model performance for both VGG, ResNet, and Transformer architectures. This indicates that excessive differences in membrane potential across timesteps are prevalent across various SNN architectures and datasets, and that our solution is able to consistently mitigate this problem. The visualization of the membrane potential distribution before and after smoothing is shown in Fig. 3, and it can be seen that smoothing does indeed reduce the distribution difference, especially from $T1$ to $T2$ to $T3$. In addition, smoothing and guidance alone can improve SNN performance, and the combination of both can maximize performance gains. This shows that our two solutions are not mutually exclusive and can work in synergy to improve the stability of the SNN ensemble. Appendices A.6 to A.7 show the ablation studies of our method for object detection and neuromorphic speech recognition tasks.

Table 2: Ablation study results of the proposed method (%).

Dataset	Method	VGG-9	ResNet-18	SpikingResformer
CIFAR10-DVS	Baseline	73.97	66.73	77.60
	+Smooth	$74.80_{+0.83}$	$68.07_{+1.34}$	$78.45_{+0.85}$
	+Guidance	$76.23_{+2.26}$	$69.33_{+2.60}$	$79.23_{+1.63}$
	+Both	$\mathbf{76.77}_{+2.80}$	$\mathbf{70.03}_{+3.30}$	$\mathbf{80.60}_{+3.00}$
DVS-Gesture	Baseline	87.85	80.56	90.63
	+Smooth	$89.93_{+2.08}$	$82.99_{+2.43}$	$91.32_{+0.69}$
	+Guidance	$91.32_{+3.47}$	$84.84_{+4.28}$	$93.06_{+2.43}$
	+Both	$\mathbf{93.23}_{+5.38}$	$\mathbf{85.30}_{+4.74}$	$\mathbf{94.44}_{+3.81}$

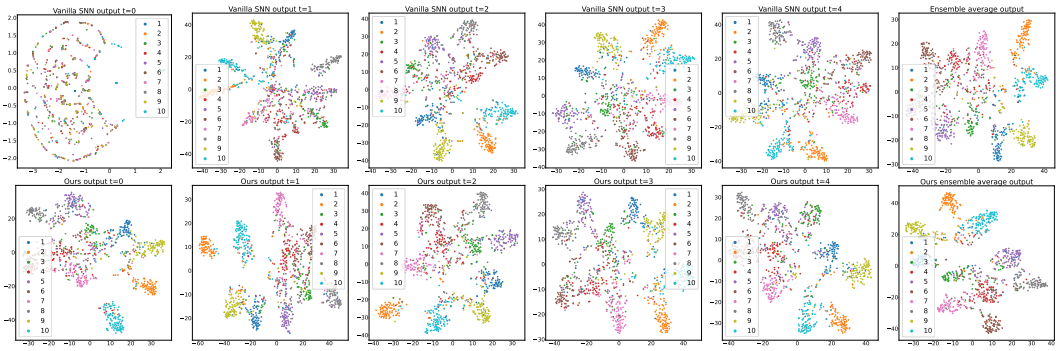


Figure 4: Two-dimensional t-SNE visualization on the CIFAR10-DVS dataset. **Top**: The output of the vanilla SNN varies greatly across timesteps, and the overall output is confusing, making it difficult to distinguish between classes. **Bottom**: The output of our SNN is more stable across timesteps and more distinguishable across classes, especially for the first two timesteps.

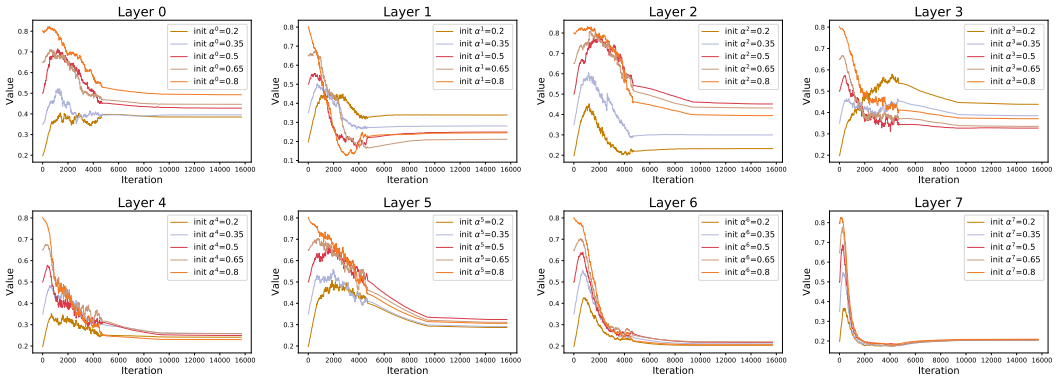


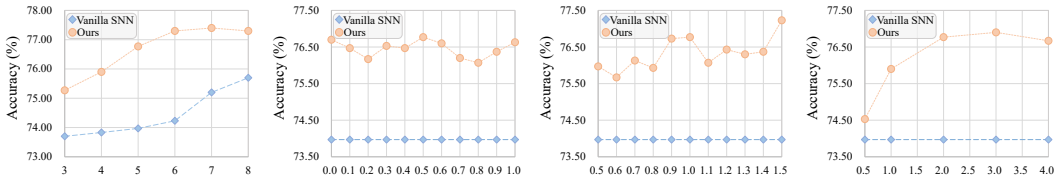
Figure 5: The optimization trend of α during training. α with different initial values gradually converge with training iterations, indicating that our method is insensitive to the initial value of α.

4.2 Output Visualization

We have visualized the output of the SNN in Fig. 4 with 2D t-distributed stochastic neighbor embedding (t-SNE) to show the improvement in output stability and distinguishability with our method. In Fig. 4(Top), the output of the vanilla SNN at each individual timestep varies widely, and the output of the first two timesteps in particular is confusing. This results in a poor distinguishability of its ensemble average output, which limits the recognition performance. Our SNN on the other hand, has more stable outputs across timesteps, and in particular the outputs generated at the first two timesteps are also well distinguished, as shown in Fig. 4(Bottom). Benefiting from the stability across timesteps, our final output is more spread across different classes of clusters and more compactly distributed within the clusters, yielding better performance. Please see Appendix A.4 for more visualization comparisons.

4.3 Smoothing Coefficient Analysis

By default, α is initialized to 0.5. To analyze the influence of the initialization value on α, we visualize the optimization trend of α in each layer of VGG-9 trained on CIFAR10-DVS in Fig. 5, where the initial values of α are taken from $\{0.2,0.35,0.5,0.65,0.8\}$. As can be seen in Fig. 5, α gradually converges at each layer during the training iterations. In particular, α in the last four layers gradually converges to almost the same optimal value, indicating that our method is robust to initial values. The first few layers of α do not converge to the same optimal value, which we attribute to the accumulation of errors in the surrogate gradient. The SNN uses the surrogate gradient instead of the derivative of the spike activity, which introduces a gradient error in backpropagation, and this error accumulates the further ahead the layer is. Eventually, these gradient errors cause the parameters of the earlier layers to be underoptimized, so that α does not converge to the same optimal value. However, compared to the 73.97% accuracy of the vanilla SNN, these different α initialization values achieved average accuracies of 76.20%, 76.80%, 76.77%, 75.75%, and 76.85%, respectively, both of which provide significant performance gains.



(a) Timestep (b) Drop probability (c) Coefficient γ (d) Temperature

Figure 6: Influence of hyperparameters on performance. (a) Our method consistently outperforms vanilla SNN across different timesteps. (b) (c) Our method is insensitive to the drop probability of the guidance loss and the coefficient γ. (d) Our method is insensitive to the temperature hyperparameter within a reasonable range ($T_{KL} \geq 1$).

Table 3: Comparative results with existing methods on neuromorphic datasets. * denotes self-implementation results with open-source code. † denotes knowledge transfer from static data.

Dataset	Method	Architecture	T	Accuracy (%)
CIFAR10-DVS	Ternary Spike Guo et al. (2024)	ResNet-20	10	78.70
	SLTT Meng et al. (2023)	VGG-11	10	77.17
	NDOT Jiang et al. (2024a)	VGG-11	10	77.50
	SSNN Ding et al. (2024)	VGG-9	5	73.63
	SLT Anumasa et al. (2024)	VGG-9	5	74.23*
	CLIF Huang et al. (2024)	VGG-9	5	74.97*
	SpikingResformer Shi et al. (2024)	SpikingResformer-Ti	5	77.60*
	SDT Yao et al. (2023)	Spiking Transformer-2-256	5	72.53*
	TRT Zuo et al. (2024b)	Spiking Transformer-2-256	5	75.55
	Ours	VGG-9	5	76.77
		SpikingResformer-Ti	5	**80.60**
DVS-Gesture	SSNN Ding et al. (2024)	VGG-9	5	90.74
	TRT Zuo et al. (2024b)	VGG-9	5	91.67
	SLT Anumasa et al. (2024)	VGG-9	5	89.35*
	SpikingResformer Shi et al. (2024)	SpikingResformer-Ti	5	90.63*
	SDT Yao et al. (2023)	Spiking Transformer-2-256	5	92.24*
	Ours	VGG-9	5	93.23
		SpikingResformer-Ti	5	**94.44**
N-Caltech101	TCJA-TET-SNN Zhu et al. (2024)	CombinedSNN	14	82.50
	EventMix Shen et al. (2023)	ResNet-18	10	79.47
	TIM Shen et al. (2024)	Spikformer	10	79.00
	NDA Li et al. (2022)	VGG-11	10	78.20
	Knowledge-Transfer He et al. (2024)	VGGSNN	10	91.72*†
	SSNN Ding et al. (2024)	VGG-9	5	77.97
	TEBN Duan et al. (2022)	VGG-9	5	81.24*
	Ours	VGG-9	5	82.71
		VGGSNN	10	**93.68**†

4.4 Influence of Hyperparameters

To explore the influence of hyperparameters on the performance of the proposed method, we show in Fig. 6 the performance of VGG-9 on CIFAR10-DVS with different hyperparameter settings.

We increased the timestep from 3 to 8 and found that the overall performance of the model also gradually increased and then saturated, and the average accuracy reached a maximum of 77.4% when the timestep was 7, as shown in Fig. 6(a). Compared to the vanilla SNN, our method consistently shows better performance.

The influence of the drop probability of the guidance loss and the coefficient γ on the performance is shown in Fig. 6(b) and Fig. 6(c), and the results show that our method is not sensitive to these hyperparameters and consistently outperforms the vanilla SNN. This indicates that our method can significantly improve model performance without intentionally adjusting the hyperparameters.

In Fig. 6(d), we investigate the influence of the temperature hyperparameter T_{KL} in the guidance on the performance, taking values set to $\{0.5, 1, 2, 3, 4\}$. The results show that the performance of our method fluctuates only slightly when $T_{KL} > 1$, and degrades when $T_{KL} = 0.5$ (still outperforming the vanilla SNN). We argue that this is due to the fact that too small a T_{KL} causes the softened subnetwork logit to be too sharp, making it difficult to pull together subnetwork outputs that are already too dissimilar. These experiments show that our method is not sensitive to specific values as long as the hyperparameters are within reasonable ranges, and thus offers great robustness.

Table 4: Comparative results (%) on static datasets. * denotes self-implementation results.

Method	Architecture	#Param (M)	T	CIFAR10	CIFAR100
CLIF Huang et al. (2024)	ResNet-18	11.21	4	94.89	77.00
RMP-Loss Guo et al. (2023a)	ResNet-19	12.54	4	95.51	78.28
NDOT Jiang et al. (2024a)	VGG-11	9.23	4	94.86	76.12
TAB Jiang et al. (2024b)	ResNet-19	12.54	4	94.76	76.81
SLT-TET Anumasa et al. (2024)	ResNet-19	12.54	4	95.18	75.01
Spikformer Zhou et al. (2023)	Spiking Transformer-4-384	9.28	4	95.19	77.86
SpikingResformer Shi et al. (2024)	SpikingResformer-Ti	10.79	4	95.93*	78.23*
SDT Yao et al. (2023)	Spiking Transformer-2-512	10.21	4	95.60	78.40
Ours	SpikingResformer-Ti	10.79	4	**96.16**	**79.22**

4.5 Comparison with Existing Methods

Neuromorphic object/gesture recognition. Table 3 shows the comparative results on neuromorphic datasets. On CIFAR10-DVS and DVS-Gesture, our VGG-9 achieves an accuracy of 76.77% and 93.23%, respectively, with only 5 timesteps. Using the Transformer architecture, we achieved accuracies of 80.60% and 94.44%, respectively, significantly exceeding other methods. In addition, we conducted experiments on N-Caltech101 and achieved 82.71% accuracy using VGG-9. Note that we only used vanilla LIF neurons and the general training method, which can further improve the performance when combined with other methods. For example, employing the knowledge transfer strategy He et al. (2024), we were able to increase our accuracy to 93.68%.

Table 5: Comparative results (%) on point cloud classification.

Method	Type	T	ModelNet10	ModelNet40
PointNet++ Qi et al. (2017)	ANN	-	95.50	92.16
Converted SNN Lan et al. (2023)	SNN	16	92.75	89.45
Spiking PointNet Ren et al. (2023)	SNN	2	92.98	88.46
P2SResLNet Wu et al. (2024)	SNN	1	-	89.20
Ours	SNN	2	**94.54**	**91.13**
		1	94.39	89.82

Static object recognition and 3D point cloud classification. In addition to neuromorphic datasets, we also conducted experiments on static object recognition and 3D point cloud classification tasks to demonstrate the generalizability of our method. The comparative results for static datasets are shown in Table 4, where we achieved 96.16% and 79.22% accuracy for CIFAR10 and CIFAR100, respectively, outperforming the other methods. For the point cloud classification task, we performed experiments on the ModelNet10/40 datasets Wu et al. (2015) using the lightweight PointNet++ architecture Qi et al. (2017), and the comparative results are shown in Table 5. With $T = 2$, our method achieves 94.54% and 91.13% accuracy, respectively, outperforming other spiking models. Since our method relies on the ensemble of temporal subnetworks, it is not able to produce gains when trained with $T = 1$. To evaluate the one-timestep inference performance, we directly infer the trained two-timestep model with one timestep. This preserves the benefits of our method while avoiding additional training overhead. The results show that even with only one timestep, we can achieve accuracies of 94.39% and 89.82%, which still outperform existing SNN models. Experiments on large-scale ImageNet, object detection, and neuromorphic speech recognition are presented in Appendices A.5, A.6, and A.7.

5 Conclusion

In this paper, we rethink the SNN from an ensemble learning perspective and identify a key factor that is generally overlooked but has a large impact on performance: excessive differences in membrane potential distributions can lead to unstable outputs across timesteps, thereby affecting overall performance. To mitigate this, we propose membrane potential smoothing and temporally adjacent subnetwork guidance to facilitate consistency of initial membrane potentials and outputs at each timestep, respectively, thereby improving ensemble stability and performance. Meanwhile, membrane potential smoothing creates additional pathways for forward information propagation and gradient backpropagation, mitigating the temporal gradient vanishing and providing dual gains. We have conducted extensive experiments on neuromorphic/static object/gesture/speech recognition, object detection, and 3D point cloud classification tasks using various architectures to confirm the effectiveness and versatility of our method. These experiments show that our method consistently outperforms other existing methods. We expect that our work will inspire the community to further analyze the spatio-temporal properties of SNNs.

REFERENCES

Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

Arnon Amir et al. A low power, fully event-based gesture recognition system. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7388–7397, 2017.

Srinivas Anumasa, Bhaskar Mukhoty, Velibor Bojkovic, Giulia De Masi, Huan Xiong, and Bin Gu. Enhancing training of spiking neural network with stochastic latency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10900–10908, 2024.

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*, 2022.

Biswadeep Chakraborty, Beomseok Kang, Harshit Kumar, and Saibal Mukhopadhyay. Sparse spiking neural network: Exploiting heterogeneity in timescales for pruning recurrent SNN. In *The Twelfth International Conference on Learning Representations*, 2024.

Benjamin Cramer, Yannik Stradmann, Johannes Schemmel, and Friedemann Zenke. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(7):2744–2757, 2022.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *International Conference on Learning Representations*, 2021.

Yongqi Ding, Lin Zuo, Kunshan Yang, Zhongshu Chen, Jian Hu, and Tangfan Xiahou. An improved probabilistic spiking neural network with enhanced discriminative ability. *Knowledge-Based Systems*, 280:111024, 2023.

Yongqi Ding, Lin Zuo, Mengmeng Jing, Pei He, and Yongjun Xiao. Shrinking your timestep: Towards low-latency neuromorphic object recognition with spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 11811–11819, 2024.

Yiting Dong, Dongcheng Zhao, and Yi Zeng. Temporal knowledge sharing enable spiking neural network learning from past and future. *IEEE Transactions on Artificial Intelligence*, 5(7):3524–3534, 2024.

Chaoteng Duan, Jianhao Ding, Shiyan Chen, Zhaofei Yu, and Tiejun Huang. Temporal effective batch normalization in spiking neural networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Daniel Elbrecht, Shruti R. Kulkarni, Maryam Parsa, J. Parker Mitchell, and Catherine D. Schuman. Evolving ensembles of spiking neural networks for neuromorphic systems. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1989–1994, 2020.

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 (ICCV)*, pp. 2661–2671, October 2021b.

Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023a.

Wei Fang, Zhaofei Yu, Zhaokun Zhou, Ding Chen, Yanqi Chen, Zhengyu Ma, Timothée Masquelier, and Yonghong Tian. Parallel spiking neurons with high efficiency and ability to learn long-term dependencies. *Advances in Neural Information Processing Systems*, 36, 2023b.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International conference on machine learning*, pp. 1607–1616. PMLR, 2018.

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 2022.

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 (ICCV)*, pp. 17391–17401, October 2023a.

Yufei Guo, Weihang Peng, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Xuhui Huang, and Zhe Ma. Joint a-snn: Joint training of artificial and spiking neural networks via self-distillation and weight factorization. *Pattern Recognition*, 142:109639, 2023b.

Yufei Guo, Yuanpei Chen, Xiaode Liu, Weihang Peng, Yuhan Zhang, Xuhui Huang, and Zhe Ma. Ternary spike: Learning ternary spikes for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12244–12252, 2024.

Xiang He, Dongcheng Zhao, Yang Li, Guobin Shen, Qingqun Kong, and Yi Zeng. An efficient knowledge transfer strategy for spiking neural networks from static to event domain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 512–520, 2024.

Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

JiaKui Hu, Man Yao, Xuerui Qiu, Yuhong Chou, Yuxuan Cai, Ning Qiao, Yonghong Tian, Bo XU, and Guoqi Li. High-performance temporal reversible spiking neural networks with $\mathcal{O}(l)$ training memory and $\mathcal{O}(1)$ inference cost. In *Forty-first International Conference on Machine Learning*, 2024.

Yulong Huang, Xiaopeng LIN, Hongwei Ren, Haotian FU, Yue Zhou, Zunchang LIU, biao pan, and Bojun Cheng. CLIF: Complementary leaky integrate-and-fire neuron for spiking neural networks. In *Forty-first International Conference on Machine Learning*, 2024.

Somayeh Hussaini, Michael Milford, and Tobias Fischer. Ensembles of compact, region-specific regularized spiking neural networks for scalable place recognition. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4200–4207, 2023.

Haiyan Jiang, Giulia De Masi, Huan Xiong, and Bin Gu. NDOT: Neuronal dynamics-based online training for spiking neural networks. In *Forty-first International Conference on Machine Learning*, 2024a.

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*, 2024b.

kang you, Zekai Xu, Chen Nie, Zhijie Deng, Qinghai Guo, Xiang Wang, and Zhezhi He. SpikeZIP-TF: Conversion is all you need for transformer-based SNN. In *Forty-first International Conference on Machine Learning*, 2024.

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, pp. 11270–11277, 2020.

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

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 (ICCV)*, pp. 9211–9220, October 2023.

Jun Ho Lee, Jae Soon Baik, Tae Hwan Hwang, and Jun Won Choi. Learning from data with noisy labels using temporal self-ensemble. *arXiv preprint arXiv:2207.10354*, 2022.

Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, 11, 2017.

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*, pp. 631–649. Springer, 2022.

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. *arXiv preprint arXiv:2407.20708*, 2024.

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

Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Towards memory- and time-efficient backpropagation for training spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6166–6176, October 2023.

Georgiana Neculae, Oliver Rhodes, and Gavin Brown. Ensembles of spiking neural networks, 2021.

Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9, 2015.

Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Ensemblesnn: Distributed assistive stdp learning for energy-efficient recognition in spiking neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2629–2635, 2017.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

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. In *Artificial Neural Networks and Machine Learning – ICANN 2024*, pp. 422–436, 2024.

Dayong Ren, Zhe Ma, Yuanpei Chen, Weihang Peng, Xiaode Liu, Yuhan Zhang, and Yufei Guo. Spiking pointnet: Spiking neural networks for point clouds. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Lior Rokach. Ensemble-based classifiers. *Artificial intelligence review*, 33:1–39, 2010.

Yangjun Ruan, Saurabh Singh, Warren Richard Morningstar, Alexander A. Alemi, Sergey Ioffe, Ian Fischer, and Joshua V. Dillon. Weighted ensemble self-supervised learning. In *ICLR*, 2023.

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*, 11:682, 2017.

Mert Bulent Sariyildiz, Philippe Weinzaepfel, Thomas Lucas, Diane Larlus, and Yannis Kalantidis. Unic: Universal classification models via multi-teacher distillation, 2024.

Guobin Shen, Dongcheng Zhao, and Yi Zeng. Eventmix: An efficient data augmentation strategy for event-based learning. *Information Sciences*, 644:119170, 2023.

Sicheng Shen, Dongcheng Zhao, Guobin Shen, and Yi Zeng. Tim: An efficient temporal interaction module for spiking transformer, 2024.

Xinyu Shi, Zecheng Hao, and Zhaofei Yu. Spikingresformer: Bridging resnet and vision transformer in spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5610–5619, June 2024.

Qiaoyi Su, Yuhong Chou, Yifan Hu, Jianing Li, Shijie Mei, Ziyang Zhang, and Guoqi Li. Deep directly-trained spiking neural networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6555–6565, October 2023.

Mohammad Reza Taesiri, Giang Nguyen, Sarra Habchi, Cor-Paul Bezemer, and Anh Totti Nguyen. Imagenet-hard: The hardest images remaining from a study of the power of zoom and spatial biases in image classification. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Luke Taylor, Andrew J King, and Nicol Spencer Harper. Addressing the speed-accuracy simulation trade-off for adaptive spiking neurons. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Linh Tran, Bastiaan S Veeling, Kevin Roth, Jakub Swiatkowski, Joshua V Dillon, Jasper Snoek, Stephan Mandt, Tim Salimans, Sebastian Nowozin, and Rodolphe Jenatton. Hydra: Preserving ensemble diversity for model distillation. *arXiv preprint arXiv:2001.04694*, 2020.

Lihao Wang and Zhaofei Yu. Autaptic synaptic circuit enhances spatio-temporal predictive learning of spiking neural networks. In *Forty-first International Conference on Machine Learning*, 2024.

Ziming Wang, Runhao Jiang, Shuang Lian, Rui Yan, and Huajin Tang. Adaptive smoothing gradient learning for spiking neural networks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35798–35816. PMLR, 23–29 Jul 2023.

Qiaoyun Wu, Quanxiao Zhang, Chunyu Tan, Yun Zhou, and Changyin Sun. Point-to-spike residual learning for energy-efficient 3d point cloud classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 6092–6099, 2024.

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.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Qu Yang, Jibin Wu, Malu Zhang, Yansong Chua, Xinchao Wang, and Haizhou Li. Training spiking neural networks with local tandem learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Man Yao, JiaKui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo XU, and Guoqi Li. Spike-driven transformer. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

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. In *The Twelfth International Conference on Learning Representations*, 2024.

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, 2024.

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.

Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng YAN, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations*, 2023.

Rui-Jie Zhu, Malu Zhang, Qihang Zhao, Haoyu Deng, Yule Duan, and Liang-Jian Deng. Tcja-snn: Temporal-channel joint attention for spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2024.

Lin Zuo, Yongqi Ding, Mengmeng Jing, Kunshan Yang, and Yunqian Yu. Self-distillation learning based on temporal-spatial consistency for spiking neural networks. *arXiv preprint arXiv:2406.07862*, 2024a.

Lin Zuo, Yongqi Ding, Wenwei Luo, Mengmeng Jing, Xianlong Tian, and Kunshan Yang. Temporal reversed training for spiking neural networks with generalized spatio-temporal representation. *arXiv preprint arXiv:2408.09108*, 2024b.

A Appendix

A.1 Integrating Membrane Potential Smoothing in LIF Neurons

In this paper, membrane potential smoothing is integrated into commonly used LIF neurons. The neuron dynamics after integration can be expressed as:

$$U_i^l(t) = \left(1 - \frac{1}{\tau}\right) H_i^l(t-1), \text{leakage} \tag{12}$$

$$\tilde{H}_i^l(t) = \alpha^l \tilde{H}_i^l(t-1) + (1-\alpha^l) U_i^l(t), \text{smoothing} \tag{13}$$

$$H_i^l(t) = \tilde{H}_i^l(t) + I_i^l(t), \text{charge} \tag{14}$$

$$S_i^l(t) = \begin{cases} 1, & H_i^l(t) \geq \vartheta \\ 0, & H_i^l(t) < \vartheta \end{cases}, \text{fire spike} \tag{15}$$

$$H_i^l(t) = H_i^l(t) - S_i^l(t)\vartheta, \text{reset} \tag{16}$$

where $\tilde{H}$ is the smoothed membrane potential. Note that $\tilde{H}(t-1)$ does not exist when $t = 0$, at which point the dynamics of the neuron are the same as the original LIF dynamics. When $t = 1$, $\tilde{H}(t-1) = \tilde{H}(0)$ is set to the initialized value of membrane potential H, which is 0 in this paper.

A.2 PyTorch-Style Code

For reproducibility, we provide Pytorch-style code for the drop function $drop(\cdot)$, which randomly drops guidance losses in the Algorithm 2 inspired by Sariyildiz et al. (2024).

Algorithm 2 PyTorch-style code for randomly dropping guidance losses

```
# losses: Guidance loss between T-1 temporally adjacent sub-networks.
# losses.shape: [T-1]
# p: random discard probability.
def drop(losses,p):
   T,B,C,H,W = x.shape
   w = torch.ones((losses.shape[0]))
   index = torch.argmax(losses)
   for i in range(losses.shape[0]):
      if i == index:
         continue
      else:
         p = random.random()
         if p < prob:
            w[i] = 0
   w.div_(w.sum())
   return w * kd_loss
```

A.3 Experimental Details

A.3.1 Datasets

In this paper, we perform experiments on the neuromorphic datasets CIFAR10-DVS, DVS-Gesture, and N-Caltech101, as well as the static image datasets CIFAR10, CIFAR100, and the 3D point cloud datasets ModelNet10 and ModelNet40.

CIFAR10-DVS Li et al. (2017) is a benchmark dataset for neuromorphic object recognition, which contains 10,000 event samples of size 128×128. The dimension of each event sample x is $[t, p, x, y]$, where t is the time stamp, p is the polarity of the event, indicating the increase or decrease of the pixel value, and $[x, y]$ are the spatial coordinates. There are 10 classes of samples in CIFAR10-DVS, and we divide each class of samples into training and test sets in the ratio of 9:1 to evaluate the model performance, the same as the existing work Zuo et al. (2024b); Shi et al. (2024); Yao et al. (2023).

DVS-Gesture Amir et al. (2017) dataset contains event samples for 11 gestures, of which 1176 are used for training and 288 are used for testing. The dimension of each event sample is $[t, p, x, y]$ and the spatial dimension size is 128×128.

N-Caltech101 Orchard et al. (2015) contains event stream data for 101 objects, each sample with a spatial size of 180×240. There are 8709 samples in N-Caltech101, and we divide the training set and the test set at a ratio of 9:1.

Since the event data is of high temporal resolution, we use the SpikingJelly Fang et al. (2023a) framework to integrate each event sample into T event frames, where T corresponds to the timestep of the SNN. In addition, event frames are downsampled to a spatial size of 48×48 before being input to the SNN.

CIFAR10 and CIFAR100 Krizhevsky et al. (2009) contain 60,000 static images for object recognition, of which 50,000 are used for training and 10,000 for testing. The size of each image is 32×32. There are 10 categories of images in CIFAR10 and 100 categories in CIFAR100.

ModelNet10 Wu et al. (2015) and ModelNet40 Wu et al. (2015) are benchmark datasets for 3D point cloud classification. ModelNet10 contains point cloud data for 4899 objects in ten categories; ModelNet40 contains data for 12311 objects in 40 categories. For point cloud data preprocessing, we follow Ren et al. (2023). We uniformly sample 1024 points on the mesh faces and input them into the SNN after normalizing them to a unit sphere.

A.3.2 Training Setting

Our experiments are based on the PyTorch package, using the Nvidia RTX 4090 GPU. By default, we use the VGG-9 Zuo et al. (2024b) architecture on the neuromorphic datasets. For ResNet-18, we use the architecture settings described in Ding et al. (2024). We trained the model for 100 epochs using a stochastic gradient descent optimizer with an initial learning rate of 0.1 and a tenfold decrease every 30 epochs. We trained the VGG-9 and ResNet-18 models without using any data augmentation techniques, and the weight decay value was 1e-3. The batch size during training is 64. The firing threshold ϑ and membrane potential time constant τ of spiking neurons were 1.0 and 2.0, respectively.

When our method is used for the Transformer architecture SNN, we use the SpikingResformer Shi et al. (2024) architecture. At this point, our training strategy is exactly the same as in the original paper, and we use the officially released training code directly.

Our method can be combined with the knowledge transfer strategy He et al. (2024) to maximize performance gains, and we conducted experiments on N-Caltech101 using the officially released code. At this point, our training strategy is exactly the same as A, but we set the batch size to 48 to reduce the memory overhead.

For the 3D point cloud classification task, we use the spiking version of the lightweight PointNet++ Qi et al. (2017) architecture. We directly use the code released by Ren et al. (2023), and thus our training strategy is exactly the same as that of Ren et al. (2023).

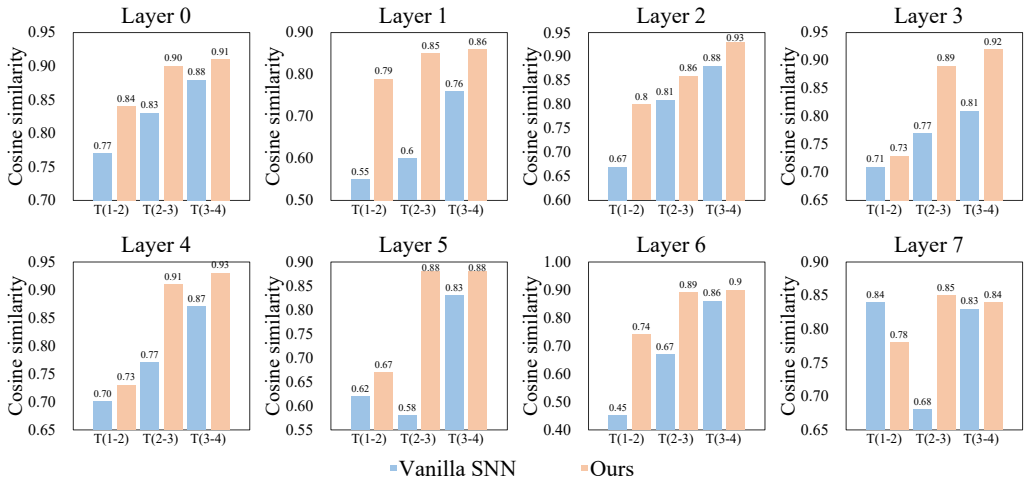


Figure 7: Comparison of the cosine similarity of membrane potential distributions for adjacent timesteps between the vanilla SNN and our method. $T(a-b)$ denotes the similarity of the distribution between timestep a and timestep b. Our method significantly improves the similarity of the membrane potential distribution across timesteps, thus facilitating the ensemble performance.

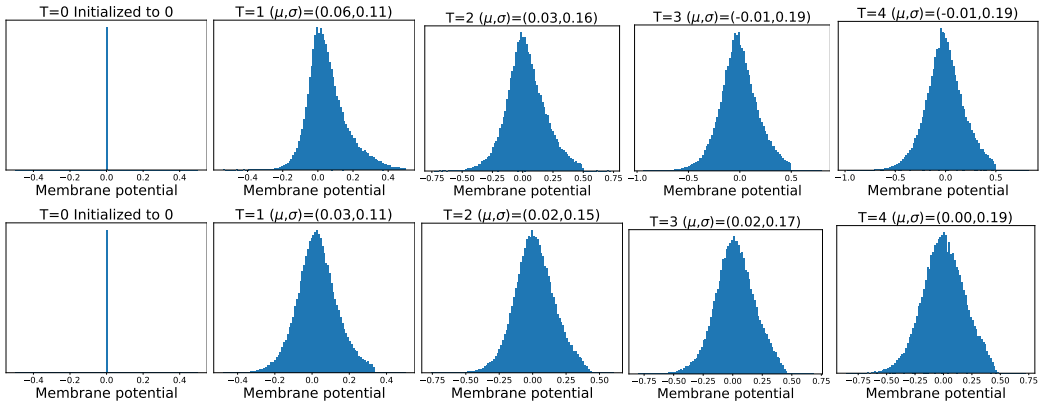


Figure 8: Additional visualization of the membrane potential distribution on CIFAR10-DVS for comparison ((μ, σ) denotes the mean and standard deviation of the distribution). **Top**: Vanilla SNN. **Bottom**: Our method allows for a more stable distribution with smaller differences across timesteps.

A.4 Additional Visualizations

In this section, we provide additional visualizations to demonstrate the effectiveness of our method in enhancing the similarity of membrane potential distributions across timesteps.

The cosine similarity of the membrane potential distribution across timesteps for the eight convolutional layer neurons in VGG-9 is shown in Fig. 7. Since the membrane potential is initialized to 0, the similarity of the membrane potential distribution between the 0th timestep and the 1st timestep is 0, and we ignore this term in the figure. It can be seen that our method consistently shows a higher similarity compared to the vanilla SNN, which mitigates the difference in the distribution of membrane potentials across timesteps. Although our method has lower similarity than the vanilla SNN for layer 7 timestep 1 and timestep 2, this exception does not affect our overall role in reducing distributional differences.

In addition, additional visualizations of the membrane potential distribution on CIFAR10-DVS are shown in Fig. 8 to illustrate the effect of our method in smoothing the membrane potential distri-

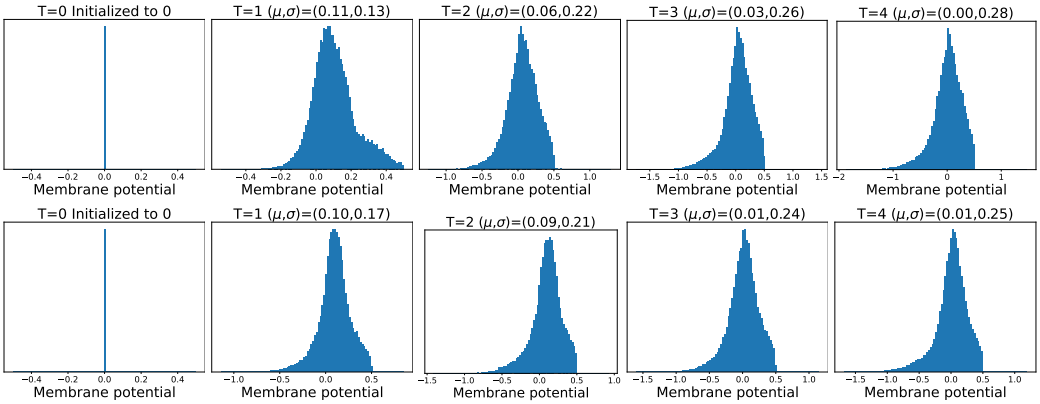


Figure 9: Membrane potential distribution in spiking VGG-9 on DVS-Gesture ((μ, σ) denotes the mean and standard deviation of the distribution). **Top**: Vanilla SNN. **Bottom**: The overall membrane potential distribution of our method is more stable.

bution. The results of the membrane potential visualization on DVS-Gesture are shown in Fig. 9, where again our method shows a more consistent distribution.

A.5 Experiments on Tiny-ImageNet, ImageNet-Hard, and ImageNet

To validate the scalability of our method, we performed experiments on Tiny-ImageNet, ImageNet-Hard Taesiri et al. (2023), and ImageNet.

The comparative results with existing methods on Tiny-ImageNet are shown in Table 6. Our method achieves an accuracy of 58.04% in only 4 timesteps, surpassing other comparative methods.

Table 6: Comparative results on the Tiny-ImageNet dataset.

Method	Architecture	T	Accuracy (%)
Online LTL Yang et al. (2022)	VGG-16	16	56.87
ASGL Wang et al. (2023)	VGG-13	8	56.81
Joint A-SNN Guo et al. (2023b)	VGG-16	4	55.39
Ours	VGG-13	4	**58.04**

ImageNet-Hard is slightly smaller in scale than ImageNet, but more challenging. We take Fang et al. (2023b) as the baseline and employ the same training strategy as Fang et al. (2023b). The comparative results on ImageNet-Hard are shown in Table 7. The results show that our method can still be effective for this challenging dataset.

Table 7: Comparative results with Fang et al. (2023b) on the challenging ImageNet-Hard dataset.

Method	Architecture	T	Top-1 Acc. (%)	Top-5 Acc. (%)
PSN Fang et al. (2023b)	SEW-ResNet18	4	44.32	52.57
Ours	SEW-ResNet18	4	**45.89**	**53.16**

Since ImageNet requires more training resources and time, we show the performance of training 250 epochs in Table 8. The results show that our method is already able to achieve competitive performance even with only 250 epochs of training. We will continue to update the latest experimental results.

A.6 Object Detection Experiment

We perform object detection on PASCAL VOC 2012 and COCO 2017 to evaluate whether the proposed method can provide performance gains on non-classification tasks. When our method is

Table 8: Comparative results with existing methods on ImageNet dataset.

Method	Architecture	T	Accuracy (%)
RecDis-SNN Guo et al. (2022)	ResNet-34	6	67.33
RMP-Loss Guo et al. (2023a)	ResNet-34	4	65.17
SSCL Zhang et al. (2024)	ResNet-34	4	66.78
TAB Jiang et al. (2024b)	ResNet-34	4	67.78
SEW-ResNet Fang et al. (2021a)	SEW-ResNet34	4	63.18
Ours	SEW-ResNet34	4	**69.03**

Table 9: Comparative results with existing methods on PASCAL VOC 2012. $T \times D$ indicates that T timesteps are set and each timestep is expanded D times. D is set to 1 by default in the comparative methods.

Method	#Param (M)	$T \times D$	mAP@50 (%)	mAP@50:95 (%)
Spiking-YOLO Kim et al. (2020)	10.2	3500	51.8	-
SpikeYOLO Luo et al. (2024)	13.2	2×4	56.0	37.6
Ours	13.2	2×4	**57.0**	**38.4**

Table 10: Comparative results with existing methods on COCO 2017 val. $T \times D$ indicates that T timesteps are set and each timestep is expanded D times. D is set to 1 by default in the comparative methods.

Method	#Param (M)	$T \times D$	mAP@50 (%)	mAP@50:95 (%)
Spiking-YOLO Kim et al. (2020)	10.2	3500	-	25.7
EMS-YOLO Su et al. (2023)	26.9	4	50.1	30.1
Meta-Spikeformer (YOLO) Yao et al. (2024)	16.8	4	50.3	-
SpikeYOLO Luo et al. (2024)	13.2	2×4	54.7	39.0
Ours	13.2	2×4	**55.2**	**39.5**

Table 11: Experimental results (%) of membrane potential smoothing on time-dependent SHD dataset.

Method	RNN-LIF	RNN-DH-LIF Zheng et al. (2024)
Vanilla	81.87	90.33
+Smoothing	$83.04_{+1.17}$	$89.86_{+0.47}$

used for non-classification tasks (e.g., regression), we can compute the MSE loss between outputs instead of the KL divergence. We take SpikeYOLO Luo et al. (2024) as the baseline and compute the guidance loss using MSE for its predicted coordinates, and the results for the PASCAL VOC 2012 and COCO 2017 datasets are shown in Table 9 and Table 10, respectively (Train 300 epochs on PASCAL VOC 2012 and 100 epochs on COCO 2017). The results show that our method can be applied to the object detection task and improve the performance of the baseline model, and should be similarly applicable to other regression tasks.

A.7 Experiments on the SHD Dataset

In this section, we explore whether the proposed membrane potential smoothing can be applied to more time-dependent tasks. To this end, we performed experimental evaluations on the Spiking Heidelberg digits (SHD) dataset Cramer et al. (2022) with DH-LIF Zheng et al. (2024) as the baseline. The model architecture is a two-layer RNN with eight branches of DH-LIF neurons. Additionally, we also compare it with a vanilla two-layer RNN-LIF. The experimental results are shown in Table 11. The results show that membrane potential smoothing still improves the performance of both baselines, demonstrating the generalizability of the method.

Furthermore, the experiments show that this view of an SNN as an ensemble of multiple subnetworks also holds for time-dependent tasks. Although time-dependent tasks require more time-varying information than static tasks, too much variance can still negatively affect their performance. Our membrane potential smoothing reduces excessive variance across timesteps, improving ensemble stability and overall performance. It is worth noting that smoothing does not completely eliminate variance, thus preserving the necessary dynamic information and allowing its application to time-dependent tasks.

A.8 Generalizability Experiments with Other Neurons

We used LIF, PLIF Fang et al. (2021b), and CLIF Huang et al. (2024) neuron models as baselines on CIFAR10-DVS to evaluate the generalizability of membrane potential smoothing. The experimental results are shown in Table 12. The results show that membrane potential smoothing yields consistent performance gains with great generalization. In addition, Table 11 shows that membrane potential smoothing is equally effective in Spiking RNNs.

Table 12: Experimental results (%) of membrane potential smoothing across multiple neuron models on CIFAR10-DVS.

	LIF	PLIF	CLIF
Original	73.97	74.83	74.97
+Smoothing	$74.80_{+0.83}$	$75.10_{+0.27}$	$75.97_{+1.00}$

A.9 Timestep Hyperparameter and Power Consumption Analysis

The timestep of the SNN is proportional to the training and inference overhead. To balance performance and overhead, the timestep in this paper on the neuromorphic dataset was set to 5. To evaluate the influence of the time-step hyperparameter on the performance, we conducted experiments on DVS-Gesture based on the SpikingResformer Shi et al. (2024) architecture, and the results are shown in Table 13. The results show that even at larger time steps, our method is still able to improve the performance of the baseline model, although the effect gradually decreases as the performance of the model saturates.

In addition, Table 13 compares the power consumption of our method with that of the baseline model. The calculation of the power consumption follows Shi et al. (2024) (Power consumption is positively correlated with the number of spikes). The results show that our method only slightly increases the power consumption when the timestep is small, and our method with lower power consumption when the timestep is large. In particular, the low number of spikes of the SNN at low timesteps tends to lead to an inadequate feature representation. Our method generates slightly more spikes at low timesteps to enhance the representation quality and thus the performance. In contrast, when the timestep is large, the vanilla SNN suffers from redundant spikes, and our method reduces the redundant spikes, thus improving performance and reducing power consumption. Overall, the difference in power consumption between our method and the baseline model is negligible, and therefore hardly affects the power consumption of the SNN model.

Table 13: Comparison of performance and power consumption at different timesteps. Experiments were performed on DVS-Gesture with the SpikingResformer architecture.

		T=4	T=5	T=8	T=10	T=16
Acc. (%)	Baseline	89.93	90.63	92.89	93.58	96.99
	+Smoothing	$91.67_{+1.74}$	$94.44_{+3.81}$	$94.32_{+1.43}$	$94.33_{+0.75}$	$97.23_{+0.24}$
Power (mJ)	Baseline	0.3699	0.4796	0.8119	1.0374	1.5788
	+Smoothing	0.3869	0.4874	0.8079	1.0199	1.5769