Shrinking Your TimeStep: Towards Low-Latency Neuromorphic Object Recognition with Spiking Neural Networks

Yongqi Ding, Lin Zuo*, **Mengmeng Jing, Pei He, Yongjun Xiao**

School of Information and Software Engineering
University of Electronic Science and Technology of China
linzuo@uestc.edu.cn

Abstract

Neuromorphic object recognition with spiking neural networks (SNNs) is the cornerstone of low-power neuromorphic computing. However, existing SNNs suffer from significant latency, utilizing 10 to 40 timesteps or more, to recognize neuromorphic objects. At low latencies, the performance of existing SNNs is significantly degraded. In this work, we propose the Shrinking SNN (SSNN) to achieve low-latency neuromorphic object recognition without reducing performance. Concretely, we alleviate the temporal redundancy in SNNs by dividing SNNs into multiple stages with progressively shrinking timesteps, which significantly reduces the inference latency. During timestep shrinkage, the temporal transformer smoothly transforms the temporal scale and preserves the information maximally. Moreover, we add multiple early classifiers to the SNN during training to mitigate the mismatch between the surrogate gradient and the true gradient, as well as the gradient vanishing/exploding, thus eliminating the performance degradation at low latency. Extensive experiments on neuromorphic datasets, CIFAR10-DVS, N-Caltech101, and DVS-Gesture have revealed that SSNN is able to improve the baseline accuracy by $6.55\% \sim 21.41\%$. With only 5 average timesteps and without any data augmentation, SSNN is able to achieve an accuracy of 73.63% on CIFAR10-DVS. This work presents a heterogeneous temporal scale SNN and provides valuable insights into the development of high-performance, low-latency SNNs.

Introduction

Brain-inspired spiking neural networks (SNNs) mimic the mammalian brain and transmit information between neurons via discrete spikes (Maass 1997). The all-or-zero spikes and the event-driven nature make SNNs an extremely low-energy computational paradigm (Deng et al. 2020). In addition, the temporal dynamics inherent within the spiking neurons endow SNNs with superior temporal feature extraction ability (Zuo et al. 2020; Ponghiran and Roy 2022; Ding et al. 2023). Neuromorphic data represent information in the form of 0-1 event streams similar to spikes. Computation of neuromorphic data based on SNNs holds the promise of high-performance and low-power neuromorphic computing.

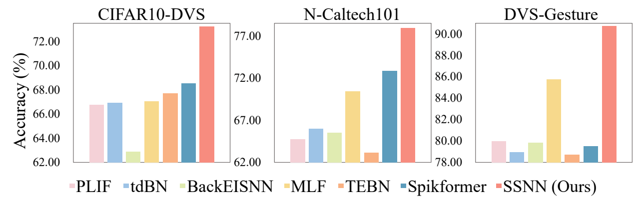


Figure 1: Comparative results of the proposed SSNN and existing SNNs (with an average timestep of 5). Results show that SSNN exceeds existing SNNs by a large margin.

To improve the performance of SNNs, researchers have made efforts in various aspects such as fine-grained spiking neurons (Fang et al. 2021b; Yao et al. 2022; Ding et al. 2022), modeling of biological neural properties (Zhao, Zeng, and Li 2022; Sun et al. 2023), and optimized training strategies (Deng et al. 2022; Guo et al. 2022b). Unfortunately, existing SNNs still require high latency to recognize neuromorphic objects. For instance, on the neuromorphic benchmark datasets CIFAR10-DVS (Li et al. 2017), N-Caltech101 (Orchard et al. 2015), and DVS-Gesture (Amir et al. 2017), 10 timesteps or more are necessary to achieve satisfactory accuracy (Feng et al. 2022; Yao et al. 2022). With fewer timesteps, these methods are not working well, as shown in Fig. 1. Therefore, achieving high-performance and low-latency simultaneously needs further exploration.

Several recent works (Li, Jones, and Furber 2023; Li et al. 2023) have introduced dynamic strategies into SNNs. By employing the sample-wise timestep, the average timestep used for recognition is greatly reduced. Although this sample-wise timestep potentially limits parallel inference (batch size of 1 in (Li, Jones, and Furber 2023)), it demonstrates the potential of SNNs at very low latencies. Another work (Kim et al. 2022) pointed out the phenomenon of temporal information concentration: during the training of SNNs, the effective information is gradually aggregated to the earlier timestep. The above work reveals the high temporal redundancy of existing SNNs, which raises a question: *is it possible to reduce redundant timesteps in SNNs without sacrificing performance and parallelism?* To this end, we propose the Shrinking SNN (SSNN) for low-latency, high-performance, parallelizable neuromorphic ob-

*Corresponding author: Lin Zuo.

ject recognition. Specifically, inspired by the diversity of timescales in biological neurons (Cavanagh, Hunt, and Kennerley 2020), we divide the SNN into multiple stages with progressively shrinking timesteps. To mitigate information loss due to timestep shrinkage, a simple yet effective temporal transformer is employed to smoothly transform the temporal scale of information, preserving the most valuable information with negligible additional overhead. Throughout inference, the average timestep can be significantly reduced with trivial performance degradation and without reducing parallelism.

SNNs trained from scratch with surrogate gradients suffer from two major issues: (1) mismatch between surrogate and true gradients; and (2) gradient vanishing/exploding associated with binary spikes. To further improve the performance of low-latency SNNs, we focus on these two issues by adding multiple early classifiers to SNNs to facilitate training without reducing inference efficiency. Rather than vanilla SNNs that rely only on the final global loss, losses at multiple early classifiers provide more immediate gradient feedback signals. These immediate feedback signals effectively mitigate these two major issues affecting the performance of SNNs. Meanwhile, these early classifiers also contribute to the optimization of the temporal transformer, allowing valid information to be retained more completely during timestep shrinkage.

To verify the proposed method, we conducted extensive experiments on neuromorphic datasets CIFAR10-DVS, N-Caltech101, and DVS-Gesture. At very low latencies, the performance of SSNN substantially exceeds that of existing SNNs, as shown in Fig. 1. In summary, the contributions of this work are as follows:

- We propose to reduce temporal redundancy through timestep shrinkage towards low-latency SSNN, which represents a novel SNN paradigm with heterogeneous temporal scales.
- We add multiple early classifiers to the SNN to facilitate training, which alleviates the mismatch between the surrogate gradient and the true gradient, as well as the gradient vanishing/exploding. In this way, SSNN simultaneously achieves low latency and high performance.
- Extensive experiments on the neuromorphic benchmark datasets CIFAR10-DVS, N-Caltech101, and DVS-Gesture validate the superior performance of our method. At an average timestep of 5, SSNN is able to improve the baseline accuracy by $6.55\% \sim 21.41\%$. Compared to existing SNNs, SSNN achieves remarkable advantages at low latencies.

Background

Spiking Neuron

Spiking neurons are in an iterative process of charging, firing, and resetting. When a spiking neuron i receives input current from the presynaptic neurons, it incorporates the input current into the membrane potential for charging:

$$H_i^l(t) = f(U_i^l(t-1), I_i^l(t)), \tag{1}$$

where $I_i^l(t)$ represents the input current, consisting of the spikes fired by the presynaptic neurons and the corresponding synaptic efficacy; l and t denote the layer and the timestep, respectively; $N(l-1)$ indicates the number of neurons in layer $l-1$; H and U denote the membrane potential of the neuron just after receiving the input current and firing spike, respectively; $f(\cdot)$ is the charging function, which varies across spiking neurons.

Once the membrane potential $H_i^l(t)$ reaches the firing threshold ϑ, the spiking neuron will fire a spike and deliver it to the postsynaptic neurons:

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

After the neuron fires a spike, the membrane potential is reduced by the same magnitude as the threshold ϑ, known as soft reset:

$$U_i^l(t) = r(H_i^l(t), S_i^l(t)) = H_i^l(t) - S_i^l(t)\vartheta. \tag{3}$$

In this paper, the most commonly used leaky integrate-and-fire (LIF) (Feng et al. 2022) neuron is used as an example. The charging function of the LIF neuron is:

$$H_i^l(t) = \left(1 - \frac{1}{\tau}\right) U_i^l(t-1) + I_i^l(t), \tag{4}$$

where τ is the membrane potential time constant that controls the amount of membrane potential leakage.

Surrogate Gradient-Based SNN Training Method

As can be seen from Eq. (2), the spike activity is not differentiable, and therefore the BP algorithm cannot work directly for SNNs. During backpropagation, the surrogate gradient-based method replaces the ill gradient of the spike activity with a smooth surrogate gradient function $h(\cdot)$. This method makes it feasible to train SNNs using the Backpropagation Through Time (BPTT) algorithm.

Specifically, the SNN calculates the spike according to Eq. (2), and the derivative of the spike w.r.t. the membrane potential according to Eq. (5):

$$\frac{\partial S_i^l(t)}{\partial H_i^l(t)} \approx \frac{\partial h(H_i^l(t), \vartheta)}{\partial H_i^l(t)}, \tag{5}$$

where $h(H_i^l(t), \vartheta)$ is the surrogate gradient function. In this paper, the rectangular function (Wu et al. 2018) is used:

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

where a serves as the hyperparameter controlling the shape of the rectangular function and is set to 1.

Method

In this section, we describe the proposed SSNN for low-latency, high-performance neuromorphic object recognition. We first present two key strategies in SSNN: timestep shrinkage and early classifier, and then elaborate on the overall SSNN framework.

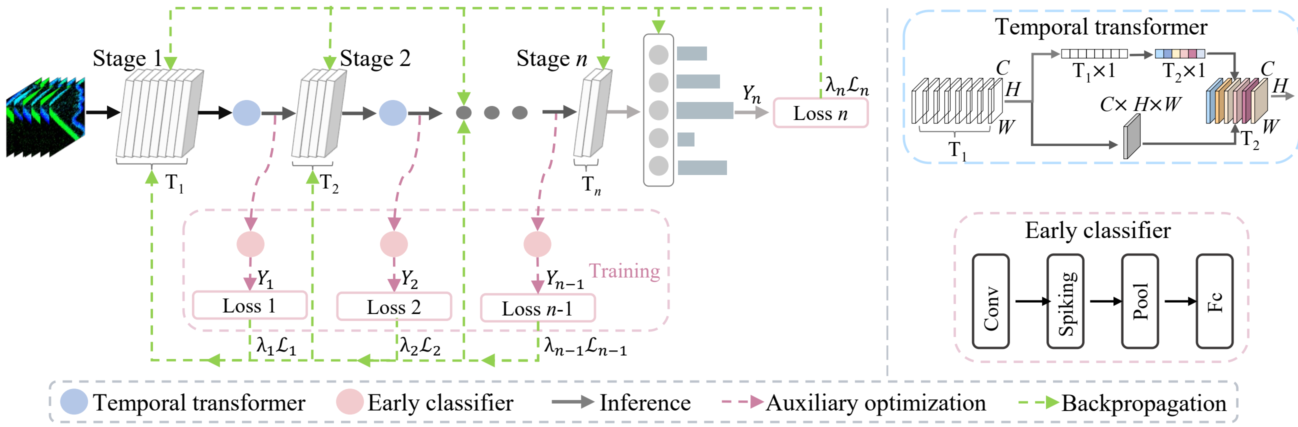


Figure 2: Overview of SSNN. The SSNN is divided into n stages with gradually shrinking timesteps $\{T_1, T_2, \cdots, T_n\}$, and the temporal transformer transforms the temporal scale of the information. During training, an early classifier is added after each stage of the SSNN except the last one. The predictions generated by each early classifier are used to facilitate the optimization of the parameters by calculating the losses along with the ground truth (green arrows).

Timestep Shrinkage

As shown in Fig. 2, for SNNs with arbitrary architecture and layers, we divide it into n stages. The timestep required for each stage keeps shrinking as the network layers get deeper, that is, $T_1 > T_2 > \cdots > T_n$. This practice follows the basic principle that the input part of the SNN requires a large timestep to extract sufficiently valuable features from the input, while the remaining part of the SNN uses fewer timesteps to reduce the inference latency. At the same time, the gradual shrinkage of the timestep, rather than a sudden and drastic reduction, allows the information transmitted to be effectively retained when the temporal scale is reduced, thus preventing performance degradation.

Temporal Transformer For stages with different timesteps, the temporal dimension of the data processed is different. For example, assume that the timesteps of the two stages are T_1 and T_2, and the data that can be processed are $\boldsymbol{I}_1 \in \mathbb{R}^{T_1 \times C \times H \times W}$ and $\boldsymbol{I}_2 \in \mathbb{R}^{T_2 \times C \times H \times W}$, where C, H, and W denote the channel, height, and width of the intermediate feature map, respectively. To allow smooth transmission of information between two stages with different temporal scales, the dimensionality of the output of the pre-stage must be converted to an acceptable dimension for the post-stage and then transmitted to the post-stage. Therefore, how to transform information between different temporal scales and maximize the preservation of valid information is a non-trivial problem. To enable the smooth transformation of information across temporal scales, we propose a lightweight and tractable yet effective temporal transformer. Denoting the output of the pre-stage as $\boldsymbol{O}_1 \in \mathbb{R}^{T_1 \times C \times H \times W}$, the temporal transformer is described as follows, and the graphical representation is shown at the top right of Fig. 2.

First, the timestep-wise global average descriptor of $\boldsymbol{O}_1$ is calculated within T_1 timesteps:

$$\boldsymbol{O}_1^{avg} = \frac{1}{C \times H \times W} \sum_{i=1}^{C} \sum_{j=1}^{H} \sum_{k=1}^{W} \boldsymbol{O}_{1,i,j,k}, \tag{7}$$

where $\boldsymbol{O}_1^{avg} \in \mathbb{R}^{T_1 \times 1}$ is the timestep-wise global average descriptor. Then, the temporal score $\boldsymbol{d} \in \mathbb{R}^{T_2 \times 1}$ for T_2 timesteps can be obtained using a nonlinear transformation:

$$\boldsymbol{d} = \text{softmax}(\boldsymbol{W}\boldsymbol{O}_1^{avg}), \tag{8}$$

where $\boldsymbol{W} \in \mathbb{R}^{T_2 \times T_1}$ is the learnable weight of the nonlinear transformation. The softmax function keeps the sum of temporal scores to 1 to ensure the complete assignment of information. The temporal score $\boldsymbol{d}$ serves as the basis for reassigning $\boldsymbol{O}_1$ in T_2 timesteps. Calculate the sum of $\boldsymbol{O}_1$ over T_1 timesteps as the total information and assign it to T_2 timesteps based on $\boldsymbol{d}$ to obtain $\boldsymbol{I}_2 \in \mathbb{R}^{T_2 \times C \times H \times W}$:

$$\boldsymbol{I}_{2,t} = \boldsymbol{O}_1^{total} \odot \boldsymbol{d}_t = \sum_{t'=1}^{T_1} \boldsymbol{O}_{1,t'} \odot \boldsymbol{d}_t, \tag{9}$$

where $\boldsymbol{O}_1^{total} \in \mathbb{R}^{C \times H \times W}$ is the total information; $\odot$ denotes the multiplication with the broadcast mechanism.

The $\boldsymbol{I}_2$ obtained after the above transformation has a post-stage compatible temporal dimension and can be used as input to the post-stage to continue the forward inference. In this way, the timestep can be continuously shrunk to an ultra-low value without causing significant information loss during the shrinking process.

Average Timestep and Overhead Analysis Assuming that the SNN is divided into n stages, each stage containing n_i computational units (a convolutional layer and a layer of spiking neurons), requiring timesteps of $\{T_1, T_2, \cdots, T_n\}$, then the average timestep for inference using the SNN with timestep shrinkage can be approximated as:

$$T_{avg} = \frac{\sum_i^n n_i T_i}{\sum_i^n n_i}. \tag{10}$$

Note that the fully connected layer used for classification (which runs at the minimum timestep T_n) is not considered in Eq. (10), so the actual average timestep is less than T_{avg}. For an SNN that does not use timestep shrinkage, the average timestep is equal to the timestep of each stage. In subsequent experiments, the SNNs with and without timestep shrinkage are compared at the same average timestep to confirm the effectiveness of timestep shrinkage.

The number of learnable parameters introduced by the temporal transformer is positively correlated with the timesteps of the two adjacent stages. Within the whole SNN, the number of additional introduced parameters $w_{add} = \sum_{1}^{n-1} T_i T_{i+1}$. For neuromorphic data, the timestep before and after the shrinkage is typically less than or equal to 10. The number of stages n that divide an SNN is generally kept between 3 and 5. Accordingly, w_{add} is kept to the order of a hundred, which is negligible compared to the millions of total parameters in the SNN.

Early Classifier

SNNs trained with the surrogate gradient bypass the non-differentiability of the spike activity, but the mismatch between the surrogate gradient and the true gradient limits the performance of the SNN. On the other hand, binary spikes make SNNs suffer from more severe gradient vanishing/exploding than artificial neural networks (Fang et al. 2021a). To improve the performance of SSNN and eliminate the performance degradation at low latency, we mitigate these two major problems by adding multiple early classifiers during training. These losses at the early classifiers deliver more immediate gradient signals, effectively alleviating both problems while contributing to the optimization of the temporal transformer.

During training, the output of each stage, except the last one, is passed to an early classifier for auxiliary optimization (rose arrows in Fig. 2). Each early classifier consists of a convolutional layer, a spiking neuron layer, and a fully connected layer, similar to (Teerapittayanon, McDanel, and Kung 2016), capable of predicting the object based on intermediate features generated by the corresponding stage (the bottom right of Fig. 2). In this paper, early classifiers have the same structure and can actually be customized depending on the difficulty of the task and the location of the early classifiers. Alternatively, a globally shared early classifier can be used after each stage to reduce the overhead during training. We use the same structure of early classifiers rather than a deliberate setting in order to facilitate a clearer focus on the effect of this strategy rather than the structure of each early classifier. The output of the last stage predicts the object directly through the fully connected layer and without passing it to the early classifier. Thus, each stage will produce a prediction specific to the input and calculate the loss with the ground truth.

The output is decoded in a rate-based manner, that is, the average of multiple timestep outputs is used to calculate the loss along with the ground truth. The total loss is expressed

Algorithm 1: Training framework for SSNN

Require: input $\boldsymbol{X}$, label $\hat{\boldsymbol{Y}}$, the number of stages n of SNN, the timestep of each stage $\{T_1, T_2, \cdots, T_n\}$.
Ensure: Update network parameters.

```
Initialize network parameters W;
I_1 = X;
for i = 1 to n − 1 do
    O_i ← stage_i(I_i); // Calculate stage output
    I_{i+1} ← Eq. (7-9); // Transform output and shrink timestep
    Y_i ← EC_i(I_{i+1}); // Calculate the output of early classifier
    L_i ← L_i( (1/T_{i+1}) Σ_t^{T_{i+1}} Y_{i,t}, Ŷ ); // Calculate the loss of early classifiers
end for
O_n ← stage_n(I_n);
Y_n ← fc(O_n);
L_n ← L_n( (1/T_n) Σ_t^{T_n} Y_{n,t}, Ŷ );
L_total ← Eq. (11);
Update parameters W based on the BPTT algorithm.
```

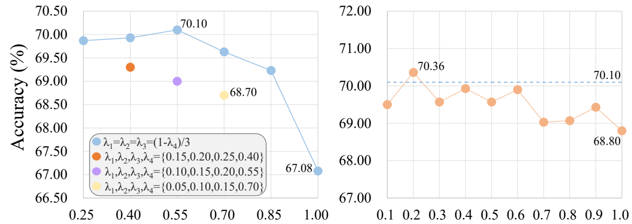
as the weighted sum of the losses at each stage:

$$\mathcal{L}_{total} = \sum_{i}^{n} \lambda_i \mathcal{L}_i \Big(\frac{1}{T_i} \sum_{t}^{T_i} \boldsymbol{Y}_{i,t}, \hat{\boldsymbol{Y}}\Big), \sum_{i}^{n} \lambda_i = 1, \tag{11}$$

where $\boldsymbol{Y}_{i,t}$ is the output of the i-th stage through early classifier at timestep t; $\hat{\boldsymbol{Y}}$ is the ground truth; $\mathcal{L}_i$ denotes the loss of the i-th stage; λ_i is the coefficient corresponding to the loss. In this paper, all losses are set to the cross-entropy (CE) loss. It is worth noting that this strategy is compatible with specially designed loss functions such as TET (Deng et al. 2022) and IM-Loss (Guo et al. 2022a). The total loss is backpropagated to each layer in the SNN based on the surrogate gradient function and the BPTT algorithm. The losses of each stage act on the optimization of the parameters in all its preceding layers. Compared with methods that rely only on the final output, these immediate gradients mitigates the impact of gradient mismatch accumulation and gradient vanishing/exploding.

The output of the first stage is based only on the extracted primary features, and the corresponding loss can only facilitate the updating of local parameters. The later stages produce output on the basis of high-level features, and their losses provide broader perspectives (green arrows in Fig. 2). In the subsequent experiments, we experimentally investigate the effect of different loss coefficients λ on the performance and show that this strategy is not sensitive to λ with great generalization.

Additional Overhead Analysis The early classifiers added after each stage are only used for auxiliary optimization and do not affect the inference process. Therefore, this strategy do not incur additional computational overhead during inference.



(a) Ablation of λ_4. **(b)** Ablation of $\lambda_1=\lambda_2=\lambda_3$ with fixed $\lambda_4=1$.

Figure 3: Influence of λ on performance. The accuracy remains at [68.70%,70.36%] as long as λ_1, λ_2 and λ_3 are not zero, indicating that our method is not sensitive to λ.

Method	CIFAR10-DVS	N-Caltech101	DVS-Gesture
VGG-9	67.08	66.45	80.29
+TS	$71.53_{+4.45}$	$75.42_{+8.97}$	$85.42_{+5.13}$
+EC	$70.10_{+3.02}$	$70.13_{+3.68}$	$84.49_{+4.20}$
SSNN	$\mathbf{73.63}_{+6.55}$	$\mathbf{77.97}_{+11.52}$	$\mathbf{90.74}_{+10.45}$
ResNet-18	58.60	52.81	65.17
+TS	$61.97_{+3.37}$	$62.62_{+9.81}$	$72.68_{+7.51}$
+EC	$64.30_{+5.70}$	$57.29_{+4.48}$	$77.43_{+12.26}$
SSNN	$\mathbf{68.10}_{+9.50}$	$\mathbf{67.51}_{+14.70}$	$\mathbf{86.58}_{+21.41}$

Table 1: Average test accuracy (%) for ablation studies of timestep shrinkage (TS), early classifier (EC), and SSNN.

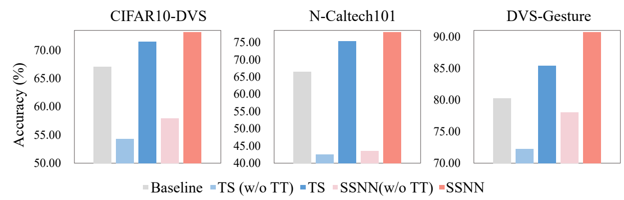


Figure 4: Influence of the temporal transformer (TT). The performance of SNNs is severely degraded and inferior to the baseline without the temporal transformer.

The SSNN Framework

SSNN achieves low-latency inference through timestep shrinkage and high performance with multiple early classifiers, as shown in Fig. 2. The temporal transformer transforms the output of the pre-stage to make it compatible with the temporal scale of the post-stage. With the immediate gradient of the early classifiers, it can facilitate the optimization of the learnable parameters in the temporal transformer, thus favoring the preservation of information during temporal scale transformation. Specifically, the output of each stage is transformed by the temporal transformer first, and then the transformed information is fed to both the early classifier and the post-stage. In this way, the input received by each early classifier has been shrunk in timestep. The training framework for SSNN is presented in Algorithm 1.

Experiments

Experiments were conducted on CIFAR10-DVS (Li et al. 2017), N-Caltech101 (Orchard et al. 2015), and DVS-Gesture (Amir et al. 2017). The preprocessing of neuromorphic data is similar to MLF (Feng et al. 2022), integrating the event streams into frames and then downsampling without any data augmentation. All experiments were repeated three times with different random seeds to reduce randomness. Two architectures, VGG-9 and ResNet-18, are used in the experiments to demonstrate the generalizability of the proposed methods. Both of these network structures are evenly divided into four stages.

Ablation Studies

Influence of λ. We first explore the influence of λ under two settings: (a) $\lambda_1 = \lambda_2 = \lambda_3 = (1 - \lambda_4)/3$; (b) The coefficients of the losses gradually increase as the layers get deeper, i.e., $\lambda_1 < \lambda_2 < \lambda_3 < \lambda_4$. The experiments were performed on CIFAR10-DVS using VGG-9 with a timestep of 5. The experimental results are shown in Fig. 3(a). When $\lambda_4 = 1, \lambda_1 = \lambda_2 = \lambda_3 = 0$, only the final loss works (auxiliary optimization does not work) with an accuracy of 67.08%. As λ_1, λ_2 and λ_3 increases, the auxiliary losses gradually contribute to the optimization of the parameters and the performance improves. When $\lambda_4 \in [0.25, 0.70]$, the accuracy fluctuates between 68.70% and 70.10%, which indicates that our method is not sensitive to λ (as long as λ_1, λ_2 and λ_3 are not 0) and has great generalizability. In addition, we step over the constraint that the weighted sum is 1 and fix $\lambda_4 = 1$ to make $\lambda_1 = \lambda_2 = \lambda_3$ to further explore the influence of λ. As shown in Fig. 3(b), the accuracy is kept between 68.80% and 70.36%, which is quite stable. In subsequent experiments, we fixed $\lambda_1 = \lambda_2 = \lambda_3 = 0.15, \lambda_4 = 0.55$ for stable results.

Comparison with the baseline SNN. We compare the proposed method with the baseline SNN on three datasets. The timesteps of the four stages of VGG-9 and ResNet-18 are set to $\{8, 6, 4, 2\}$ and $\{8, 6, 4, 1\}$, and their average timesteps are 5 and 4.94, respectively. The baseline timestep is set to 5. As shown in Table 1, the baseline does not perform well on the three datasets, indicating that the vanilla training method cannot achieve satisfactory performance at low latency. By separately employing timestep shrinkage and early classifier, there is a noticeable improvement in the accuracy, demonstrating the effectiveness of these methods in enhancing the performance of low-latency SNNs. The optimal performance is obtained by SSNN, which considerably outperforms the baseline. When inference is performed on DVS-Gesture with ResNet18, the accuracy of SSNN even exceeds that of the baseline by 21.41%.

Effectiveness of the temporal transformer. To evaluate the role of the temporal transformer, we compare the performance of timestep shrinkage and SSNN with and without the temporal transformer. Assuming T_2 is the shrunken timestep, the information of the first T_2 timesteps is taken directly for subsequent processing in the absence of the temporal transformer. VGG-9 was used for the experiments and the average timestep was set to 5. The experimental results are shown in Fig. 4. In the absence of the temporal

Dataset	Method	Network	Average timestep	Accuray (%)
CIFAR10-DVS	DSR (Meng et al. 2022)	VGG-11†	20	77.27
	GLIF (Yao et al. 2022)	7B-wideNet	16	76.80
	TET (Deng et al. 2022)	VGGSNN	10	77.33
	SLTT (Meng et al. 2023)	VGG-11	10	77.17
	SLSSNN (Xu, Liu, and Yang 2023)	VGG-16	8	77.50
	AutoSNN (Na et al. 2022)	AutoSNN†	8	72.50
	STBP-tdBN (Zheng et al. 2021)	VGG-9*	5	66.93
	PLIF (Fang et al. 2021b)	VGG-9*	5	66.77
	BackEISNN (Zhao, Zeng, and Li 2022)	VGG-9*	5	62.90
	MLF (Feng et al. 2022)	VGG-9*	5	67.07
	TEBN (Duan et al. 2022)	VGG-9*	5	67.70
	Spikformer (Zhou et al. 2023)	Spikformer*†	5	68.55
	SSNN (Ours)	VGG-9	5	**73.63**
		VGG-9	8	**78.57**
N-Caltech101	EventMix (Shen, Zhao, and Zeng 2022)	ResNet18†	10	79.47
	tdBN+NDA (Li et al. 2022)	VGG-11†	10	78.20
	STBP-tdBN (Zheng et al. 2021)	VGG-9*	5	66.01
	PLIF (Fang et al. 2021b)	VGG-9*	5	64.73
	BackEISNN (Zhao, Zeng, and Li 2022)	VGG-9*	5	65.53
	MLF (Feng et al. 2022)	VGG-9*	5	70.42
	TEBN (Duan et al. 2022)	VGG-9*	5	63.13
	Spikformer (Zhou et al. 2023)	Spikformer*	5	72.83
	SSNN (Ours)	VGG-9	5	**77.97**
		VGG-9	8	**79.25**
DVS-Gesture	teacher default-KD (Xu et al. 2023)	ResNet18	16	96.88
	STBP-tdBN (Zheng et al. 2021)	VGG-9*	5	78.94
	PLIF (Fang et al. 2021b)	VGG-9*	5	79.98
	BackEISNN (Zhao, Zeng, and Li 2022)	VGG-9*	5	79.86
	MLF (Feng et al. 2022)	VGG-9*	5	85.77
	TEBN (Duan et al. 2022)	VGG-9*	5	78.70
	Spikformer (Zhou et al. 2023)	Spikformer*	5	79.52
	SSNN (Ours)	VGG-9	5	**90.74**
		VGG-9	8	**94.91**

Table 2: Comparative results with existing methods. * denotes self-implementation results. † indicates using data augmentation.

transformer, shrinking the timestep leads to degraded performance, and even the accuracy of SSNN is lower than the baseline. This illustrates that information must be rationally reassigned when the temporal scale has changed, and that direct truncation of information will lead to dramatic loss of information. Considerable performance gains are achieved by employing the temporal transformer, which demonstrates the crucial role of the temporal transformer.

Influence of stage division and stage-wise timesteps. We divided VGG-9 into 2, 3, 4, and 5 stages and set different stage-wise timesteps by keeping the average timestep at 5 to explore the influence of these factors on the timestep shrinkage strategy. The 8 convolution layers in VGG-9 are evenly divided into 2 and 4 stages. In case of division into 3 or 5 stages, each stage contains the number of convolution layers as $\{3,3,2\}$ or $\{2,2,2,1,1\}$. In total, we explored a total of 11 different settings, see Fig. 5. As shown in Fig. 5, the accuracy is in the range of 84.38% and 88.43%, without drastic changes as the way of division and stage-wise timesteps vary. In both settings, the accuracy far exceeds the baseline accuracy of 80.29%. This verifies that our method is not sensitive to these settings and has great generalizability. It is worth noting that when divided into two stages with timesteps of 9 and 1, it achieved a surprising accuracy of 86.34%. This illustrates that the timestep shrinkage strategy can work with flexible timestep settings, even for substantial temporal scale shrinkage.

Comparison with Existing Methods

CIFAR10-DVS As shown in Table 2, SSNN outperforms all other methods at an average timestep of 5. Even though Spikformer (Zhou et al. 2023) uses a more complex transformer structure and data augmentation, the accuracy of our SSNN is still 5.08% higher than that of Spikformer. When the average timestep is increased to 8, SSNN achieves an accuracy of 78.57%, surpassing SLTT (Meng et al. 2023),

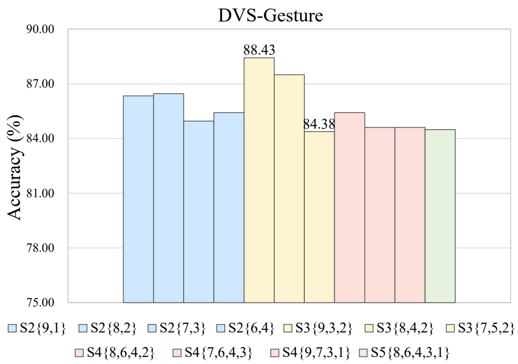


Figure 5: Influence of stage division and stage-wise timesteps, $Sn\{t_1, t_2, ..., t_n\}$ denotes division into n stages and stage-wise timestep of $\{t_1, t_2, ..., t_n\}$. Results indicate that our method is not sensitive to different stage divisions and stage-wise timesteps.

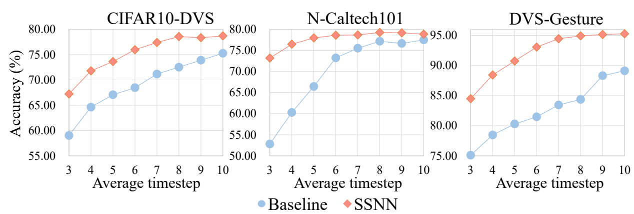


Figure 6: Influence of average timestep on performance. SSNN consistently outperforms the baseline and shows significant advantages, especially at low latencies.

TET (Deng et al. 2022), GLIF (Yao et al. 2022), and DSR (Meng et al. 2022) at higher latencies.

N-Caltech101 With an average timestep of 5, our SSNN achieves an accuracy of 77.97%, which is 5.32% higher than that of Spikformer (Zhou et al. 2023). EventMix (Shen, Zhao, and Zeng 2022) and NDA (Li et al. 2022) achieved accuracies of 79.47% and 78.20% at timestep 10, respectively, using specially designed data augmentation methods. Our SSNN achieves an accuracy of 79.25% at an average timestep of 8 without data augmentation, which exceeds NDA (Li et al. 2022) at a timestep of 10 and is only 0.22% worse than EventMix (Shen, Zhao, and Zeng 2022).

DVS-Gesture When the timestep is 5, only MLF (Feng et al. 2022) can achieve 85.77% accuracy, while the other methods have poor recognition performance. At this point, our SSNN was able to achieve an accuracy of 90.74%, exceeding MLF (Feng et al. 2022) by 4.97%. For an average timestep of 8, SSNN is able to achieve an accuracy of 94.91%, which is only 1.97% lower than that of teacher default-KD (Xu et al. 2023) at a timestep of 16, while the required inference latency is reduced by half.

Influence of Average Timestep

Here, we evaluate the influence of average timestep on the performance of SSNN. The experiments were conducted using the VGG-9 with controlled average timesteps ranging from 3 to 10, and the experimental results are shown in Fig. 6. When the average timestep is slightly larger, SSNN exceeds the baseline by a small margin; as the average timestep gradually decreases, the performance gap between the two grows more significant. It is worth noting that the performance of SSNN at low latency (e.g., 4/5/6) exceeds that of the baseline at high latency (e.g., 7/8/10, which is the timestep of the first stage when the average timestep is 4/5/6), revealing that our SSNN effectively avoids the performance degradation associated with timestep shrinkage.

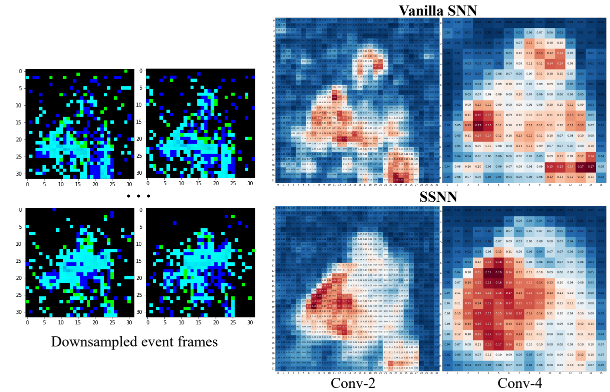


Figure 7: Visualization of spike firing rate on DVS-Gesture. Compared to the vanilla SNN, SSNN can accurately focus on the hand region, which is crucial for gesture recognition.

Visualization

To understand more clearly the feature extraction capability of SSNN, similar to (Yao et al. 2023), we visualize the spike firing rates of the second and fourth convolutional layers in both vanilla SNN and SSNN. As shown in Fig. 7, the vanilla SNN cannot focus on the waving hand and the focus area is scattered. In contrast, SSNN can exactly capture the hand region (which is critically important for gesture recognition) and therefore can extract features more accurately and perform better for recognition.

Conclusion

In this work, we propose SSNN for low-latency, high-performance neuromorphic object recognition. On the one hand, SSNN reduces temporal redundancy by progressively shrinking the timestep to achieve low-latency inference. To alleviate the information loss during timestep shrinkage, a simple yet effective temporal transformer is employed to smoothly transform the temporal scale of the information. On the other hand, SSNN mitigates the gradient mismatch and gradient vanishing/exploding by using multiple early classifiers during training to improve the recognition performance under low latency. Extensive experiments have confirmed the effectiveness of the proposed SSNN. At very low latency, the performance of SSNN far exceeds that of existing SNNs. We expect that our work can contribute to the study of heterogeneous temporal scale SNNs and inspire the development of ultra-low latency, high-performance SNNs.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62276054, 61877009), and the Sichuan Science and Technology Program (Grant No. 2023YFG0156).

References

Amir, A.; et al. 2017. A Low Power, Fully Event-Based Gesture Recognition System. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7388–7397.

Cavanagh, S. E.; Hunt, L. T.; and Kennerley, S. W. 2020. A Diversity of Intrinsic Timescales Underlie Neural Computations. *Frontiers in Neural Circuits*, 14.

Deng, L.; et al. 2020. Rethinking the performance comparison between SNNS and ANNS. *Neural Networks*, 121: 294–307.

Deng, S.; Li, Y.; Zhang, S.; and Gu, S. 2022. Temporal Efficient Training of Spiking Neural Network via Gradient Reweighting. In *International Conference on Learning Representations*.

Ding, J.; et al. 2022. Biologically Inspired Dynamic Thresholds for Spiking Neural Networks. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Ding, Y.; Zuo, L.; Yang, K.; Chen, Z.; Hu, J.; and Xiahou, T. 2023. An improved probabilistic spiking neural network with enhanced discriminative ability. *Knowledge-Based Systems*, 280: 111024.

Duan, C.; Ding, J.; Chen, S.; Yu, Z.; and Huang, T. 2022. Temporal Effective Batch Normalization in Spiking Neural Networks. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 34377–34390. Curran Associates, Inc.

Fang, W.; Yu, Z.; Chen, Y.; Huang, T.; Masquelier, T.; and Tian, Y. 2021a. Deep Residual Learning in Spiking Neural Networks. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 21056–21069. Curran Associates, Inc.

Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; and Tian, Y. 2021b. Incorporating Learnable Membrane Time Constant To Enhance Learning of Spiking Neural Networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2661–2671.

Feng, L.; Liu, Q.; Tang, H.; Ma, D.; and Pan, G. 2022. Multi-Level Firing with Spiking DS-ResNet: Enabling Better and Deeper Directly-Trained Spiking Neural Networks. In *IJCAI*, 2471–2477. International Joint Conferences on Artificial Intelligence Organization.

Guo, Y.; et al. 2022a. IM-Loss: Information Maximization Loss for Spiking Neural Networks. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 156–166. Curran Associates, Inc.

Guo, Y.; et al. 2022b. 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)*, 326–335.

Kim, Y.; Li, Y.; Park, H.; Venkatesha, Y.; Hambitzer, A.; and Panda, P. 2022. Exploring Temporal Information Dynamics in Spiking Neural Networks. arXiv:2211.14406.

Li, C.; Jones, E.; and Furber, S. 2023. Unleashing the Potential of Spiking Neural Networks by Dynamic Confidence. arXiv:2303.10276.

Li, H.; Liu, H.; Ji, X.; Li, G.; and Shi, L. 2017. CIFAR10-DVS: An Event-Stream Dataset for Object Classification. *Frontiers in Neuroscience*, 11.

Li, Y.; Geller, T.; Kim, Y.; and Panda, P. 2023. SEENN: Towards Temporal Spiking Early-Exit Neural Networks. arXiv:2304.01230.

Li, Y.; Kim, Y.; Park, H.; Geller, T.; and Panda, P. 2022. Neuromorphic Data Augmentation for Training Spiking Neural Networks. In Avidan, S.; Brostow, G.; Cissé, M.; Farinella, G. M.; and Hassner, T., eds., *Computer Vision – ECCV 2022*, 631–649. Cham: Springer Nature Switzerland.

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

Meng, Q.; Xiao, M.; Yan, S.; Wang, Y.; Lin, Z.; and Luo, Z.-Q. 2022. Training High-Performance Low-Latency Spiking Neural Networks by Differentiation on Spike Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12444–12453.

Meng, Q.; Xiao, M.; Yan, S.; Wang, Y.; Lin, Z.; and Luo, Z.-Q. 2023. Towards Memory- and Time-Efficient Backpropagation for Training Spiking Neural Networks. arXiv:2302.14311.

Na, B.; Mok, J.; Park, S.; Lee, D.; Choe, H.; and Yoon, S. 2022. AutoSNN: Towards Energy-Efficient Spiking Neural Networks. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 16253–16269. PMLR.

Orchard, G.; Jayawant, A.; Cohen, G. K.; and Thakor, N. 2015. Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *Frontiers in Neuroscience*, 9.

Ponghiran, W.; and Roy, K. 2022. Spiking Neural Networks with Improved Inherent Recurrence Dynamics for Sequential Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 8001–8008.

Shen, G.; Zhao, D.; and Zeng, Y. 2022. EventMix: An Efficient Augmentation Strategy for Event-Based Data. arXiv:2205.12054.

Sun, Y.; Zeng, Y.; Zhao, F.; and Zhao, Z. 2023. Multi-compartment Neuron and Population Encoding improved Spiking Neural Network for Deep Distributional Reinforcement Learning. arXiv:2301.07275.

Teerapittayanon, S.; McDanel, B.; and Kung, H. 2016. BranchyNet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2464–2469.

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; and Shi, L. 2018. Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks. *Frontiers in Neuroscience*, 12.

Xu, C.; Liu, Y.; and Yang, Y. 2023. SLSSNN: High energy efficiency spike-train level spiking neural networks with spatio-temporal conversion. arXiv:2307.07136.

Xu, Q.; et al. 2023. Biologically inspired structure learning with reverse knowledge distillation for spiking neural networks. arXiv:2304.09500.

Yao, M.; Zhao, G.; Zhang, H.; Hu, Y.; Deng, L.; Tian, Y.; et al. 2023. Attention Spiking Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–18.

Yao, X.; Li, F.; Mo, Z.; and Cheng, J. 2022. GLIF: A Unified Gated Leaky Integrate-and-Fire Neuron for Spiking Neural Networks. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Zhao, D.; Zeng, Y.; and Li, Y. 2022. BackEISNN: A deep spiking neural network with adaptive self-feedback and balanced excitatory–inhibitory neurons. *Neural Networks*, 154: 68–77.

Zheng, H.; Wu, Y.; Deng, L.; Hu, Y.; and Li, G. 2021. Going Deeper With Directly-Trained Larger Spiking Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12): 11062–11070.

Zhou, Z.; et al. 2023. Spikformer: When Spiking Neural Network Meets Transformer. In *The Eleventh International Conference on Learning Representations*.

Zuo, L.; Chen, Y.; Zhang, L.; and Chen, C. 2020. A spiking neural network with probability information transmission. *Neurocomputing*, 408: 1–12.