SCIENCE CHINA
Information Sciences

• Supplementary File •

Spatial-temporal knowledge fusion for continual learning in spiking neural networks

Jiashuo WANG[1,2], Qingao CHAI[1,2], Rui YAN[3] & Huajin TANG[1,2*]

[1]*College of Computer Science and Technology, Zhejiang University, Hangzhou 310000, China*
[2]*The State Key Lab of Brain-Machine Intelligence, Zhejiang University, Hangzhou 310000, China*
[3]*College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310000, China*

Appendix A Method

Appendix A.1 Preliminaries

Appendix A.1.1 *Class incremental learning definition*

Class Incremental Learning aims to build models capable of learning a sequence of tasks, where new classes are introduced incrementally without accessing the full data from previous tasks. Formally, given a sequence of tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_t$, where each task $\mathcal{T}_i$ contains a disjoint label set $\mathcal{Y}_i$ such that $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $i \neq j$, the model is required to classify inputs over all the classes observed so far based on the current task data $\mathcal{D}_t = \{(x, y) \mid x \in \mathcal{X}, y \in \mathcal{Y}_t\}$, without access to any task identity information.

Our method adopts a rehearsal-based strategy by maintaining a small memory buffer $\mathcal{M} \subset \bigcup_{i=1}^{t-1} \mathcal{D}_i$, which stores a representative subset of samples from previous tasks. During the learning of the current task $\mathcal{T}_t$, the model is trained jointly on $\mathcal{D}_t \cup \mathcal{M}$, effectively mitigating catastrophic forgetting.

Appendix A.1.2 *Leaky integrate-and-fire model*

For computational efficiency, we adopt the Leaky Integrate-and-Fire (LIF) model, a commonly used neuron model in SNNs. This model is described by the following differential equations:

$$\tau \frac{du}{dt} = -u + I, u < v_{\text{th}}, \tag{A1}$$

$$\text{fire a spike } \& \ u = u_{\text{reset}}, u \geqslant v_{\text{th}}, \tag{A2}$$

where u is the membrane potential, I is the pre-synaptic input, τ is the membrane time constant, u_{reset} denotes the reset potential, and v_{th} is the given firing threshold. Specifically, the neuron fires a spike whenever u reaches v_{th}, resetting u to u_{reset}.

To explicitly model the relationship between input current and output spikes, we adopt an iterative version of the LIF neuron model [1] based on Eq. A1, as described below:

$$\boldsymbol{I}^l[t] = \boldsymbol{W}^l \boldsymbol{s}^{l-1}[t], \tag{A3}$$

$$\boldsymbol{h}^l[t] = \alpha \boldsymbol{u}^l[t-1] + \boldsymbol{I}^l[t], \tag{A4}$$

$$\boldsymbol{s}^l[t] = \Theta\left(\boldsymbol{h}^l[t] - v_{\text{th}}\right), \tag{A5}$$

$$\boldsymbol{u}^l[t] = \boldsymbol{h}^l[t] \odot (1 - \boldsymbol{s}^l[t]) + u_{\text{reset}} \boldsymbol{s}^l[t], \tag{A6}$$

where $\boldsymbol{I}^l[t]$ is the input current at time-step t, computed as the dot product of the weights $\boldsymbol{W}^l$ and the spikes from the previous layer $\boldsymbol{s}^l[t]$. $\odot$ denotes the Hadamard product. At each time step t, the spiking neuron in l-th layer integrates the input current $\boldsymbol{I}^l[t]$ and updates its membrane potential. $\boldsymbol{h}^l[t]$ and $\boldsymbol{u}^l[t]$ denote the membrane potential after neuronal dynamics and after the trigger of a spike, α is the decay factor. $\Theta(x)$ is the Heaviside function that satisfies $\Theta(x) = 1$ when $x \geqslant 0$ and $\Theta(x) = 0$ otherwise.

Appendix A.1.3 *Surrogate gradients*

Due to the non-differentiable nature of the heaviside function, surrogate gradient methods are commonly employed in SNNs to enable effective training via backpropagation [2–4]. In this work, we adopt the surrogate gradients shaped like triangles:

$$\Theta'(\boldsymbol{u}) \approx \frac{1}{\gamma^2} \max\left(0, \gamma - |\boldsymbol{u}(t) - v_{\text{th}}|\right), \tag{A7}$$

where γ is a constraint factor that defines the range within which the gradient is activated.

* Corresponding author (email: htang@zju.edu.cn)

Appendix A.2 Overall structure

Our model is composed of two learning modules (Figure 1), similar to [5, 6]. Both modules are implemented using spiking neural networks. The slow learner focuses on general representation learning via self-supervised methods, while the fast learner rapidly adapts to new information through a dynamic structural network. To enable the reuse and extension of experiential knowledge across different tasks, general representations learned by the slow learner are fused with features extracted by the fast learner across spatial and temporal scales. The fused representations are then fed back to the fast learner, enabling rapid adaptation to task changes. It is worth noting that in this framework, only the fast learner is retained for inference, whereas the slow learner and the fusion modules serve solely as training time auxiliaries, functioning as a teacher that enhances the fast learner's representations through knowledge distillation. Since the fast learner consists entirely of spiking operations, the inference pathway remains fully spike-based, making the framework well suited for deployment on neuromorphic hardware. In addition, our method uses the same memory buffer as other approaches [7], storing both the input samples from previous tasks and their corresponding labels.

For the fast learner, we employ the dynamically structured SNNs, using DER [8] as the underlying framework. Upon the arrival of a new task t, the previously trained feature extractor Φ_{t-1} is frozen, and a new module ϕ is added to learn features specific to the new classes. Each task introduces an independent module, and these modules are progressively aggregated to form a super-feature representation, as shown in Eq. A8. In addition to the classification loss $\mathcal{L}_{\text{cls}}$, an auxiliary loss term $\mathcal{L}_{\text{aux}}$ is introduced to enhance the new module's ability to recognize novel classes, following the DER training paradigm.

$$\Phi_t(\boldsymbol{x}) = [\Phi_{t-1}(\boldsymbol{x}),\, \phi(\boldsymbol{x})]. \tag{A8}$$

For the slow learner, in order to acquire general features that are invariant to input variations but sensitive to semantic representations, we employ a standard backbone network optimized with a self-supervised learning loss. Specifically, we adopt the InfoNCE loss [9], which aims to maximize the mutual information between representations, ensuring that the learned features capture the underlying structural relationships within the data. The corresponding loss function is defined as follows:

$$\mathcal{L}_{\text{SSL}} = -\log \frac{\exp\left(f\left(\boldsymbol{z}_i^A, \boldsymbol{z}_i^B\right)/\tau_s\right)}{\sum_j \exp\left(f\left(\boldsymbol{z}_i^A, \boldsymbol{z}_j\right)/\tau_s\right)}, \tag{A9}$$

where $\boldsymbol{z}^A$ and $\boldsymbol{z}^B$ are features obtained from the slow learner and f denotes the similarity function, for which we use cosine similarity. Furthermore, the update strategy proposed in [10] is adopted to continuously acquire general representations from the data.

Appendix A.3 Spatial-temporal feature fusion

The fast learner is designed to quickly learn from samples and adapt to task changes by leveraging representations learned by the slow learner. To achieve this, the general representations obtained from the slow learner are first fused with the features extracted by the fast learner along the spatial dimensions.

Appendix A.3.1 *Spatial interaction*

In complex scenarios, features from different classes may exhibit high similarity at certain spatial scales, which can lead to inter-class interference. As a result, semantically similar categories are more likely to be confused, reducing the model's discriminative performance when distinguishing between closely related classes. By integrating multi-scale features from different stages of the feature extractor, it becomes possible to exploit both low-level spatial details and high-level semantic abstractions. This enhances the model's sensitivity to local variations and helps alleviate the interference caused by inter-class feature similarity. To effectively fuse features across multiple stages and obtain more discriminative representations, we propose a spatial multi-scale fusion module (SMF). The fused feature of the i-th spatial blocks $\boldsymbol{F}_a^i$ is calculated as follows:

$$\boldsymbol{F}_a^i = \text{SMF}(\boldsymbol{F}_f^i, \boldsymbol{F}_s^i), \tag{A10}$$

$$\phi(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \phi_3(\boldsymbol{x}), \phi_4(\boldsymbol{x})], \tag{A11}$$

where $\boldsymbol{F}_f^i$ and $\boldsymbol{F}_s^i$ denote the features obtained from the i-th block $\phi_i(\boldsymbol{x})$ in fast and slow learner, respectively.

Specifically, we adopt cross-attention (CA) for SMF. This approach effectively integrates two feature maps with distinct information content, while focusing on key information. The computation can be denoted as:

$$\boldsymbol{F}_{\text{a}} = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\text{T}}}{\sqrt{d}}\right)\boldsymbol{V}, \tag{A12}$$

$$\boldsymbol{Q} = \boldsymbol{F}_{\text{f}}\boldsymbol{W}_{\text{Q}}, \quad \boldsymbol{K} = \boldsymbol{M}\boldsymbol{W}_{\text{K}}, \quad \boldsymbol{V} = \boldsymbol{M}\boldsymbol{W}_{\text{V}}, \tag{A13}$$

$$\boldsymbol{M} = \text{concat}(\boldsymbol{F}_{\text{f}}, \boldsymbol{F}_{\text{s}}). \tag{A14}$$

where concat is concatenation and the query $\boldsymbol{Q}$ is computed from $\boldsymbol{F}_f$, while the key $\boldsymbol{K}$ and value $\boldsymbol{V}$ are obtained from $\boldsymbol{M}$ using learnable projection matrices $\boldsymbol{W}_{\text{Q}}$, $\boldsymbol{W}_{\text{K}}$, and $\boldsymbol{W}_{\text{V}}$.

To enable the fast learner to quickly adapt to task changes, the fused features are fed back into the fast learner, and we employ logits-level knowledge distillation [11, 12] to further facilitate the learning process of the fast learner. Specifically, we use the logits obtained from the multi-scale fused features as the teacher to guide the training of the fast learner and the KL divergence is used to calculate the distance between them:

$$\mathcal{L}_{\text{SKD}} = \sum_i \mathcal{L}_{\text{KD}}(\boldsymbol{o}_a^i, \bar{\boldsymbol{o}}_f^i), \tag{A15}$$

$$\mathcal{L}_{\mathrm{KD}}(p,q) = -\tau_t^2 \sum_i \frac{\exp\left(p_i/\tau_t\right)}{\sum_j \exp\left(p_j/\tau_t\right)} \log\left(\frac{\frac{\exp(p_i/\tau_t)}{\sum_j \exp(p_j/\tau_t)}}{\frac{\exp(q_i/\tau_t)}{\sum_j \exp(q_j/\tau_t)}}\right), \tag{A16}$$

where $\boldsymbol{o}_a^i$ and $\bar{\boldsymbol{o}}_f^i$ denote the logits after the fully connected layer from the fused feature and the feature from the fast learner, respectively, and τ_t is the temperature scaling factor.

Appendix A.3.2 *Temporal interaction*

Considering the unique spatiotemporal characteristics of SNNs, existing distillation methods primarily focus on spatial feature alignment. However, the output of SNNs is inherently dynamic across the temporal dimension. To address this, we propose to fuse the general representations with the features extracted by the fast learner along the temporal dimension. By performing distillation on the logits across different timesteps T, we align the representations in the time dimension, thereby enhancing the model's representational capacity and improving performance over the entire temporal range.

Given the output of the fast learner at each timestep, denoted as $\boldsymbol{o}_f^i(t)$, the overall loss combining the spatial knowledge distillation loss L_{SKD} and the temporal alignment is defined as follows:

$$\mathcal{L}_{\mathrm{STKD}} = \frac{1}{T} \sum_i \sum_t \mathcal{L}_{\mathrm{KD}}(\boldsymbol{o}_a^i, \boldsymbol{o}_f^i(t)). \tag{A17}$$

The pseudocode of the proposed method is shown in Algorithm 1.

Algorithm A1 Fast-slow learning with spatial-temporal knowledge fusion

Input: Number of tasks T, data stream $\mathcal{D}$.
Initialize: slow learner θ, $\mathcal{M} \leftarrow \{\}$.
for $\mathcal{D}_t \in \mathcal{D}$ **do**
  Add new feature extractor ϕ to fast learner to form expanded feature extractor Φ_t (Eq. A8).
  while Training **do**
    Sample training data: $(x_t, y_t) \sim \mathcal{D}_t$, $(x_m, y_m) \sim \mathcal{M}$, and interleave $\boldsymbol{x} \leftarrow (x_t, x_m)$.
    Extract $\boldsymbol{F}_f$ and $\boldsymbol{F}_s$: features of $\boldsymbol{x}$ from fast and slow learners.
    Fuse $\boldsymbol{F}_f$ and $\boldsymbol{F}_s$ across multiple spatial blocks to obtain $\boldsymbol{F}_a$ (Eq. A10).
    Compute $\mathcal{L}_{\mathrm{STKD}}$ across spatial-temporal dimension (Eq. A17).
    Compute $\mathcal{L}_{\mathrm{SSL}}$ (Eq. A9) and the loss for continual acquisition of slow learner: $\mathcal{L}_{\mathrm{cs}}$.
    Evaluate overall loss for fast learner: $\mathcal{L}_{\mathrm{fast}} = \mathcal{L}_{\mathrm{cls}} + \lambda_{\mathrm{aux}}\mathcal{L}_{\mathrm{aux}} + \lambda_{\mathrm{fuse}}\mathcal{L}_{\mathrm{STKD}}$.
    Evaluate loss for slow learner: $\mathcal{L}_{\mathrm{slow}} = \mathcal{L}_{\mathrm{SSL}} + \mathcal{L}_{\mathrm{cs}}$.
    Optimize the trainable parameters for fast and slow learners by backward propagation.
    Reset the membrane potential of spiking neurons.
    Update memory buffer $\mathcal{M} \leftarrow (\mathcal{M}, (x_t, y_t))$.
  end while
end for

Appendix A.4 Sparse activations

Although dynamic structural methods help reduce task interference when introducing new modules, they often lead to a rapid increase in the number of parameters, significantly raising the computational burden. Evidence suggests that the brain employs a mechanism known as sparse coding, in which sensory events are represented by the strong activation of a relatively small set of neurons [13, 14]. The efficient use of such sparsity for information representation allows new associations to be learned more rapidly while minimizing interference with previously stored knowledge and maintaining sufficient representational capacity [15]. Inspired by this principle, we adopt a k-winners-take-all (k-WTA) mechanism [16] to induce sparse activations, thereby enhancing learning efficiency and reducing computational cost.

In the context of k-WTA activation, the entire output of a network layer is first taken as input, and the K neurons with the highest membrane currents are activated, while the remaining neurons are suppressed. This mechanism ensures that only the top K values in the input vector are retained and propagated to the next layer, with all other elements set to zero prior to transmission.

The update rule, firing rule, and reset rule are consistent with those of the LIF neuron model (Eq. A3, Eq. A5 and Eq. A6). The k-WTA mechanism is applied before the input current is passed to the neurons, as follows:

$$\boldsymbol{M}^l[t] = TopK(\boldsymbol{I}^l[t]), \tag{A18}$$

$$\hat{\boldsymbol{I}}^l[t] = \boldsymbol{I}^l[t] \odot \boldsymbol{M}^l[t], \tag{A19}$$

$$\boldsymbol{h}^l[t] = \alpha \boldsymbol{u}^l[t-1] + \hat{\boldsymbol{I}}^l[t], \tag{A20}$$

where $TopK(\cdot)$ denotes a function that uses the k-Winners-Take-All selection to generate a mask $\boldsymbol{M}^l[t]$ that filters out the K largest neuronal input currents. $\hat{\boldsymbol{I}}^l[t]$ is the actual input current to the l-th layer after the k-WTA mechanism.

The sparse activation-based LIF-SNN layer first extracts spatial features from the input $\boldsymbol{s}^{l-1}[t]$ of the previous layer, after which the first K units with higher activation are retained to obtain $\hat{\boldsymbol{I}}^l[t]$, and then the LIF module integrates the previous time-step input $\boldsymbol{u}^l[t-1]$ and spatial features $\hat{\boldsymbol{I}}^l[t]$ into $\boldsymbol{h}^l[t]$, and then generates spikes $\boldsymbol{s}^l[t]$ of the next layer and the next time-step $\boldsymbol{u}^l[t]$ by fire and reset mechanism.

Appendix B Experiments

Appendix B.1 Experimental settings

Experiments are conducted on the CIFAR100 [17] and ImageNet100 [18] datasets, which are widely adopted benchmarks in class-incremental learning research and provide a reliable basis for fair comparison with existing methods. The CIFAR100 dataset [17] consists of 100 classes, each containing 600 color images of resolution 32×32 pixels, with 500 samples per class used for training and 100 for evaluation. The ImageNet100 dataset is a subset of ImageNet1K [18], comprising 100 selected classes, each with 1,300 training samples and 500 evaluation samples. To comprehensively evaluate the proposed method under diverse incremental learning conditions, evaluation on CIFAR100 is performed under two protocols: CIFAR100-B0 and CIFAR100-B50. In the CIFAR100-B0 protocol, all 100 classes are learned sequentially from scratch, with the class set evenly partitioned into 5, 10, or 20 incremental steps (corresponding to 20, 10, or 5 new classes per step, respectively). This protocol is designed to evaluate the model's ability to continuously accumulate knowledge from an empty initialization, where each incremental step contributes equally to the final representation. In the CIFAR100-B50 protocol, the model is initially trained on 50 base classes, followed by 5 or 10 incremental steps covering the remaining 50 classes (corresponding to 10 or 5 new classes per step, respectively). This protocol is designed to evaluate the model's ability to preserve and extend a well-established feature representation when new classes arrive. For ImageNet100, the B0 protocol is adopted with training starting from scratch in batches of 10 classes, in order to evaluate the scalability of the proposed method to higher-resolution and more complex data. Both top-1 and top-5 accuracy are reported for ImageNet100.

The motivation for adopting configurations with varying numbers of incremental steps (i.e., 5, 10, and 20 steps) lies in evaluating the model's robustness under progressively more challenging conditions. As the number of incremental steps increases, the number of classes per task decreases and the total number of incremental updates grows, which makes catastrophic forgetting more pronounced and inter-task discrimination more difficult. The 20-step setting, in particular, represents a longer-step scenario that more closely reflects practical continual learning deployment, where the model must cope with a long sequence of small incremental updates.

The overall experimental workflow follows a standard class incremental learning pipeline. When a new task arrives, the model has access only to the training samples of the new classes (specifically, 500 samples per new class on CIFAR100 and 1,300 samples per new class on ImageNet100), together with a fixed-size memory buffer of 2,000 exemplars selected from previously learned classes via the herding strategy [7], while the remaining historical data is no longer accessible. Upon the arrival of the new task, a new feature extractor is appended to the fast learner to learn features specific to the new classes, while all previously trained feature extractors are frozen to preserve learned knowledge. The model is then trained jointly on the new task data and the memory buffer for a fixed number of epochs. During each training iteration, the input samples are passed through both the fast and slow learners to obtain task-specific features and general representations, which are then fused across multiple spatial blocks via the cross-attention based fusion module. The fused representations are fed back to guide the fast learner through the spatial-temporal knowledge distillation loss, which aligns the outputs of the fast learner with those of the fused features across all timesteps. After training on the current task is completed, the memory buffer is updated to include representative samples from the new task while keeping its total size fixed. At inference time, only the fast learner is retained, and the model is required to classify inputs over all classes observed so far without access to any task identity information, reflecting the realistic constraints of class incremental learning.

Appendix B.1.1 *Evaluation metrics*

To comprehensively assess both the learning plasticity and memory stability of the proposed method, we adopt the following metrics. The accuracy on all learned tasks after learning task t is denoted as A_t. The average incremental accuracy $Avg = \frac{1}{T}\sum_{t=1}^{T} A_t$ reflects the overall performance across all incremental stages, while the last-step accuracy $Last = A_T$ indicates the final performance after the entire sequence of tasks has been learned. In addition, to directly characterize the extent of catastrophic forgetting, we further report Backward Transfer (BWT) and Forgetting under the CIFAR100-B0-10steps setting. Let $a_{t,i}$ denote the accuracy on the i-th task after the model has been trained up to task t (with $i \leqslant t$). BWT measures the average influence of learning new tasks on previously learned ones, defined as

$$\mathrm{BWT} = \frac{1}{T-1}\sum_{i=1}^{T-1}\left(a_{T,i} - a_{i,i}\right), \tag{B1}$$

where a value closer to zero indicates less interference. Forgetting measures the gap between the best historical accuracy and the final accuracy on each old task, defined as

$$\text{Forgetting } = \frac{1}{T-1}\sum_{i=1}^{T-1}\left(\max_{1\leqslant t\leqslant T-1} a_{t,i} - a_{T,i}\right), \tag{B2}$$

where a smaller value indicates stronger memory retention.

Appendix B.1.2 *Implementation details*

Our proposed method is implemented using PyTorch 2.6.0, SpikingJelly 0.0.0.0.15 [19] and PyCIL [20–22] framework. All experiments are conducted on NVIDIA RTX 4090 GPU. For the compared methods, some results are taken from DER [8], while others are adopted from their respective original papers. Following DER [8], the ANN-based method compared in the experiments adopts ResNet-18 as the feature extractor. DNE [23] and Dytox [24] use a 6-layer Vision Transformer [25] backbone. In our SNN-based method setting, for training on the CIFAR100 dataset, we utilize spiking ResNet-18 and spiking ResNet-19 [26] trained from scratch as feature extractors. For ImageNet100 dataset, we adopt spiking ResNet-18, also trained from scratch. We set the random seed to 1993 and apply AutoAugment [27] and Cutout [28] as data augmentation during training. As for sparse activations, the value of K in the k-WTA mechanism is not a globally fixed constant, but is determined on a per-layer basis. Specifically, for the l-th spiking layer containing N^l neurons, we set $K^l = \lfloor r \cdot N^l \rfloor$, where r is a single scalar hyperparameter shared across all layers, referred to as the sparse ratio. In our experiments, the sparse ratio

Table B1 Hyperparameters of networks.

Parameters	Meanings	Value
α	Decay factor of membrane potential	0.5
v_{th}	Potential threshold	1
u_{reset}	Reset potential	0
γ	Constraint factor of surrogate function	1
τ_s	InfoNCE temperature	0.2
τ_t	KL divergence temperature	2
λ_{aux}	Weighting factor of auxiliary loss	2
λ_{fuse}	Weighting factor of fusion	0.2
$\lvert\mathcal{M}\rvert$	Replay buffer size	2000
r	sparse ratio	0.6

Table B2 Comparison on CIFAR100-B0 benchmarks.

Model	**Methods**	**CIFAR100-B0-5steps**		**CIFAR100-B0-10steps**		**CIFAR100-B0-20steps**	
		Avg(%)	**Last(%)**	**Avg(%)**	**Last(%)**	**Avg(%)**	**Last(%)**
ANN	iCaRL [7]	71.14	59.70	65.27	50.74	61.20	43.75
	UCIR [29]	62.77	47.30	58.66	43.39	58.17	40.63
	BiC [30]	73.10	62.10	68.80	53.54	66.48	47.02
	WA [31]	72.81	60.80	69.46	53.78	67.33	47.31
	PODNet [32]	66.70	51.70	58.03	41.05	53.97	35.02
	RPSNet [33]	70.50	60.60	68.60	57.05	-	-
	DER [8]	76.80	68.30	75.36	65.22	74.09	62.48
	Dytox [24]	-	-	75.54	62.06	75.04	60.03
	TCIL [34]	<u>77.70</u>	<u>69.60</u>	<u>77.30</u>	66.40	<u>75.10</u>	63.50
	DNE-1head [23]	-	-	74.03	<u>69.10</u>	73.27	<u>67.61</u>
SNN	DSD-SNN [35]	-	-	60.47	-	57.39	-
	ALADE-SNN* [36]	**78.67**	69.67	75.42	63.13	72.73	59.43
	Ours (Spiking ResNet-18)†	77.02	**70.69**	**76.85**	**68.34**	**75.54**	**66.20**
	Ours (Spiking ResNet-19)	78.98	72.31	77.93	70.30	76.42	67.48

* Spiking ResNet-19 backbone

† Our method with a smaller Spiking ResNet-18 backbone surpasses the SNN state-of-the-art (ALADE-SNN) and achieves performance comparable to the best ANN methods. Bold and underlined values indicate the best results in SNN and ANN categories, respectively.

is set to $r = 0.6$, meaning that approximately 60% of the neurons with the highest membrane currents are retained at each layer while the remaining neurons are suppressed. This layer-adaptive design allows the sparsity to scale with the varying widths of different layers, avoiding over-sparsification in narrow layers and under-sparsification in wide layers, while keeping the overall sparsity level controlled by a single interpretable hyperparameter. Additionally, we use the Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9 and a weight decay of 2e-4. Furthermore, for both datasets, the time window size is set to 4. The batch sizes are set to 64 and 96 for CIFAR100 and ImageNet100, respectively. In addition, we adopt the herding strategy [7] to select representative samples from previous tasks to be stored in the memory buffer for rehearsal. The detailed parameters are shown in Table B1.

Appendix B.2 Comparison results

Appendix B.2.1 *CIFAR100-B0*

Table B2 presents the comparison results on the CIFAR100-B0 benchmark. A comprehensive introduction to the baseline models can be found in Appendix A. In most cases, our method significantly outperforms various competing approaches. In the 10-steps incremental setting, using Spiking ResNet-18 as the feature extractor at each step, our method achieves an average accuracy improvement of 1.43% and a last step accuracy improvement of 5.21% over the state-of-the-art spiking CL method ALADE-SNN, which uses Spiking ResNet-19. Under the more challenging 20-steps setting, the average and last step accuracy improvements increase to 2.81% and 6.77%, respectively. As the number of tasks increases, the problem of catastrophic forgetting becomes more pronounced, leading to substantial performance drops in most existing methods. In contrast, our approach demonstrates greater stability and resistance to forgetting, exhibiting considerably smaller declines in accuracy. It maintains strong performance even in more challenging longer-step scenarios. These results suggest that our method is more effective at accumulating general knowledge throughout the learning process.

In comparison to ANN-based methods, our approach also achieves compelling performance. Under the 10-steps setting, our method slightly lags behind TCIL in terms of average accuracy, but achieves a 1.94% higher last step accuracy. Under the 20-steps setting, our method outperforms the best ANN-based results by 0.44% in average incremental accuracy and 2.70% in last step accuracy. These results demonstrate the effectiveness of our method in mitigating forgetting and maintaining model stability by fusing the general representations from the slow learner with the features from the fast learner at spatial-temporal scales and feeding this knowledge back

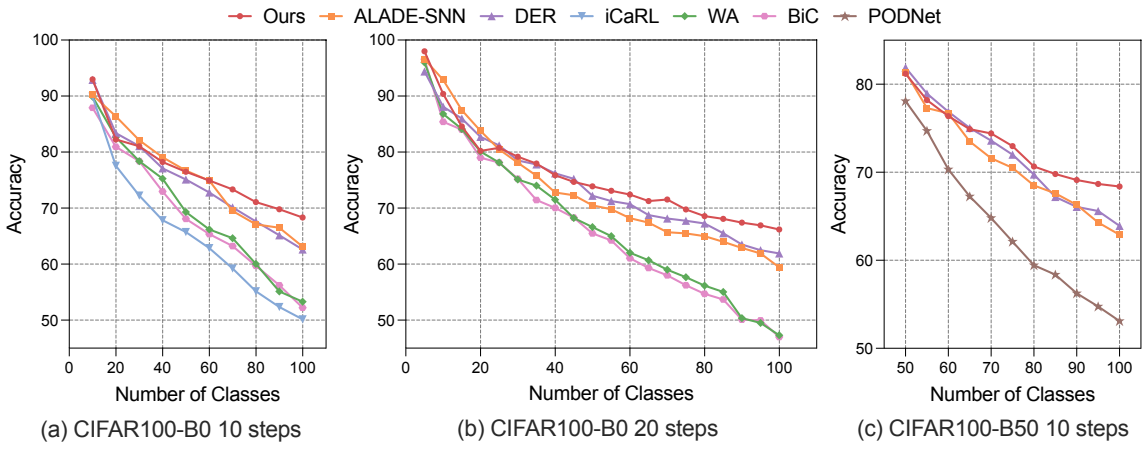


Figure B1 Accuracy curve on CIFAR100-B0-10steps, CIFAR100-B0-20steps and CIFAR100-B50-10steps.

to the fast learner.

As shown in Figure B1(a), although our method slightly underperforms the runner-up method during the initial 5 steps, it consistently outperforms all other methods in subsequent steps, ultimately achieving the highest average and last step accuracy. Furthermore, as illustrated in Figure B1(b), the performance gap between our method and other approaches continues to grow as new classes are introduced, further validating the effectiveness of our proposed method.

It should be noted that, under the 5-steps setting, our method using Spiking ResNet-18 slightly underperforms ALADE-SNN. When using the same Spiking ResNet-19 backbone as ALADE-SNN, our method achieves 0.31% higher average incremental accuracy and 2.64% higher last step accuracy. Consistent gains are also observed under other task settings.

Table B3 Comparison of BWT and Forgetting for different methods under the CIFAR100-B0-10steps setting.

	Ours	ALADE-SNN [36]	DER [8]	iCaRL [7]	WA [31]	BiC [30]
BWT(%)	-5.63	-6.78	-9.87	-21.58	-16.82	-19.51
Forgetting(%)	11.26	16.46	18.83	21.58	16.82	19.51

To more directly characterize the extent of catastrophic forgetting, we further compare our method with representative SNN-based and ANN-based baselines in terms of BWT and Forgetting under the CIFAR100-B0-10steps setting, as reported in Table B3. BWT reflects the average influence of learning new tasks on previously acquired knowledge, where a value closer to zero indicates less interference. Forgetting measures the average gap between the best historical accuracy and the final accuracy on each old task, where a smaller value indicates stronger memory retention.

As shown in Table B3, our method achieves a BWT of -5.63% and a Forgetting of 11.26%, substantially outperforming all compared baselines on both metrics. Compared with the state-of-the-art SNN-based method ALADE-SNN [36], our method reduces the BWT magnitude by 1.15% and the Forgetting by 5.20%, indicating that the proposed spatial-temporal feature fusion effectively preserves previously acquired knowledge when new tasks arrive. Compared with the dynamically expandable ANN baseline DER [8], our method also achieves a clear improvement of 4.24% in BWT and 7.57% in Forgetting, which suggests that feeding back the general representations from the slow learner to the fast learner provides stronger resistance to interference between tasks. The classical replay-based and bias-correction-based methods, including iCaRL [7], WA [31], and BiC [30], exhibit much larger BWT magnitudes and Forgetting values, further confirming the advantage of our method in mitigating catastrophic forgetting. These results collectively demonstrate that our method exhibits stronger memory stability across the entire incremental learning process.

Appendix B.2.2 *CIFAR100-B50*

As shown in Table B4, under the CIFAR100-B50 5-steps setting, our method achieves a slightly lower average accuracy than ALADE-SNN when using Spiking ResNet-18 as the backbone, but surpasses it by 2.59% in last step accuracy. In the more challenging 10-steps setting, our method outperforms ALADE-SNN by 1.65% in average accuracy and 5.08% in last step accuracy, demonstrating consistent superiority over baselines. The accuracy trends across incremental steps are illustrated in Figure B1(c). After switching to the same backbone as ALADE-SNN, our method achieves further improvements under both the 5 and 10-steps settings, reaching the best overall performance.

Appendix B.2.3 *ImageNet100-B0*

Table B5 shows the comparison results on the ImageNet100-B0-10steps setting. We report the Top-1 and Top-5 accuracy for ANN-based methods, SNN-based methods, and an oracle model using Spiking ResNet-18. SNN-based approaches generally underperform their ANN counterparts, which may be attributed to the relatively small time window used in our setting (T=4), limiting the generalization capabilities of SNNs. Nevertheless, our method achieves significantly higher average accuracy and last step accuracy than the other two SNN-based methods, demonstrating the effectiveness of our method.

Table B4 Comparison on CIFAR100-B50 benchmarks.

Model	Methods	B50-5steps		B50-10steps	
		Avg(%)	**Last(%)**	**Avg(%)**	**Last(%)**
ANN	iCaRL [7]	65.06	56.07	58.59	49.43
	UCIR [29]	64.28	51.87	59.92	48.11
	BiC [30]	66.62	55.12	60.25	48.72
	WA [31]	64.01	52.74	57.86	48.06
	PODNet [32]	67.25	55.94	64.04	51.65
	DER [8]	73.21	66.25	72.81	65.61
	TCIL [34]	<u>74.90</u>	<u>68.60</u>	<u>73.70</u>	<u>66.40</u>
SNN	ALADE-SNN* [36]	**74.50**	66.90	71.50	63.30
	Ours (Spiking ResNet-18)	74.32	**69.49**	**73.15**	**68.38**
	Ours (Spiking ResNet-19)	75.23	70.29	75.21	69.79

* Spiking ResNet-19 backbone

Table B5 Comparison on ImageNet100-B0 benchmark.

Model	Methods	ImageNet100-B0-10steps			
		Top-1		**Top-5**	
		Avg(%)	**Last(%)**	**Avg(%)**	**Last(%)**
ANN	BiC [30]	-	-	90.60	84.40
	RPSNet [33]	-	-	87.90	74.00
	WA [31]	-	-	91.00	84.10
	DER [8]	<u>77.20</u>	<u>66.70</u>	<u>93.20</u>	87.50
SNN	Spiking ResNet-18 bound	-	77.52	-	93.34
	DER-SNN†	69.94	64.16	89.13	85.04
	ALADE-SNN†	74.06	65.20	91.44	87.20
	Ours	**74.74**	**66.50**	**92.17**	**87.60**

† Reproduction results

Table B6 Ablation study.

Spatial Interaction	Temporal Interaction	**Avg(%)**	**Last(%)**
✗	✗	72.87	67.59
✓	✗	73.07	67.78
✗	✓	72.91	67.89
✓	✓	**73.15**	**68.38**

Appendix B.3 Ablation study

To evaluate the effectiveness of each component in our method, we conduct ablation studies on the CIFAR100-B50-10steps setting. Table B6 summarizes the performance of different combinations of our method's components. When spatial interaction is removed, only the final block's spatial fusion is retained. The results demonstrate that the proposed spatial interaction alleviates task confusion caused by feature similarity across classes at certain scales, by integrating multi-scale features. Moreover, adding temporal interaction further enhances performance, as it leverages the intrinsic temporal attributes of SNNs and facilitates consistent model performance across all timesteps. In this challenging longer-step setting, our method achieves 73.15% average accuracy and 68.38% last-step accuracy, demonstrating the effectiveness of the proposed components as also confirmed by the ablation results.

Appendix B.4 Empirical analysis of model behavior

Appendix B.4.1 *Confusion matrix.*

To provide a more intuitive comparison, we visualize the confusion matrices of our method and DER [8] under the CIFAR100-B0-10steps setting, as illustrated in Figure B2. It can be observed that DER suffers from a classification bias toward newly introduced classes. Specifically, many samples belonging to old classes are incorrectly recognized as instances of new classes, which results in task confusion and a noticeable degradation in overall performance. This phenomenon reflects the limitation of DER in maintaining stable knowledge of previously learned classes. In contrast, the confusion matrix of our proposed method presents a substantially brighter and clearer diagonal structure, which indicates that most samples are correctly classified into their respective categories. The reduction of off-diagonal elements further suggests that our method effectively alleviates inter-class confusion and mitigates the bias toward new

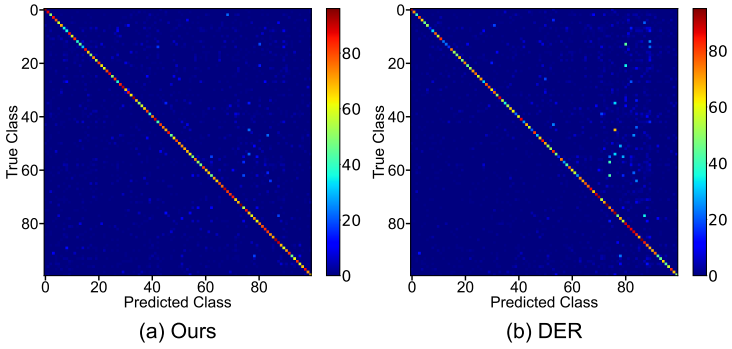


Figure B2 Confusion matrix comparison.

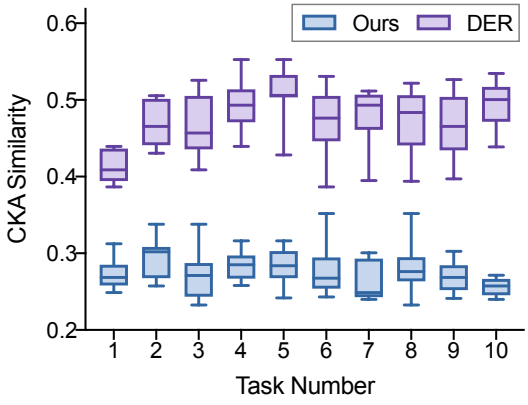


Figure B3 Representational similarity.

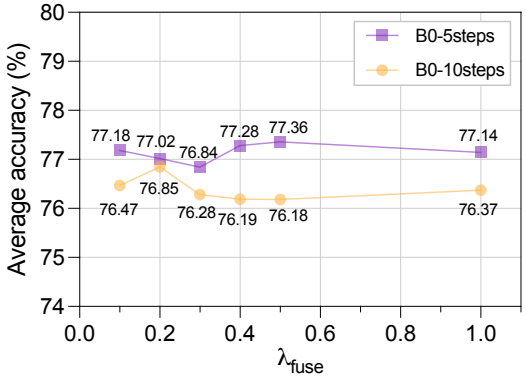


Figure B4 Hyper-parameter λ_{fuse} sensitivity under the CIFAR100-B0-10steps and CIFAR100-B0-5steps settings.

tasks. It demonstrates the robustness of our approach in preserving old knowledge while adapting to new information, thereby achieving a better balance between stability and plasticity.

Appendix B.4.2 *Representational similarity.*

We analyze the similarity of representations learned across different tasks by our method and DER [8] using Centered Kernel Alignment (CKA) [37], as illustrated in the box plot in Figure B3. The spike-based representation of a layer is obtained by calculating its firing rate. For two representations, $X \in \mathbb{R}^{b\times d}$ and $Y \in \mathbb{R}^{b\times d}$, CKA computes $\text{CKA}(\text{X},\text{Y}) = \frac{\text{HSIC}(\text{XX}^{\text{T}},\text{YY}^{\text{T}})}{\sqrt{\text{HSIC}^{\text{T}}(\text{XX}^{\text{T}},\text{XX}^{\text{T}})}\sqrt{\text{HSIC}^{\text{T}}(\text{YY}^{\text{T}},\text{YY}^{\text{T}})}}$ for Hilbert-Schmidt Independence Criterion(HSIC), using a linear kernel. As illustrated in the figure, our method exhibits a lower average CKA value compared to DER. This suggests that, from the perspective of the CLS theory, by providing additional descriptions of general features, our method helps the model capture inter-task differences from a complementary angle. As a result, it encourages the model to obtain diverse representations of tasks across learning stages. By increasing the dissimilarity between tasks, the model is better equipped to mitigate interference, thereby reducing forgetting.

Appendix B.4.3 *Impact of hyper-parameters.*

In Figure B4, we analyze the effect of varying balancing factor values on the performance under the CIFAR100-B0-10steps and CIFAR100-B0-5steps settings. As illustrated, when the weighting factor of fusion $\lambda_{\text{fuse}} \in \{0.1, 0.2, 0.3, 0.4, 0.5, 1\}$, the average accuracy remains relatively stable across different settings, indicating that the method is not sensitive to this hyper-parameter. For consistency and clarity, we set $\lambda_{\text{fuse}} = 0.2$ in all experiments.

Appendix B.5 Energy consumption estimation

SNNs transmit information through event-driven discrete spikes on neuromorphic hardware, offering significantly improved energy efficiency compared to ANNs. The spikes (0 or 1) and weight multiplications in SNNs can be efficiently implemented using Accumulate (AC) operations, whereas ANNs require more energy-intensive Multiply-Accumulate (MAC) operations. A single 32-bit floating-point AC operation consumes approximately 0.9 pJ, in contrast to 4.6 pJ for a MAC operation [38].

We visualize the spike rate of each layer in the spiking ResNet-18 under the CIFAR100-B0-10steps setting, as shown in Figure B5. In addition, following common practice in the neuromorphic computing community, we estimate the energy consumption of SNNs by computing the total synaptic operations (SOP). Specifically, the SOP for SNNs is determined by the neuronal firing rate, the fan-out f_{out}, and the time window size T. The total SOP with accumulation N_{AC} is computed as follows:

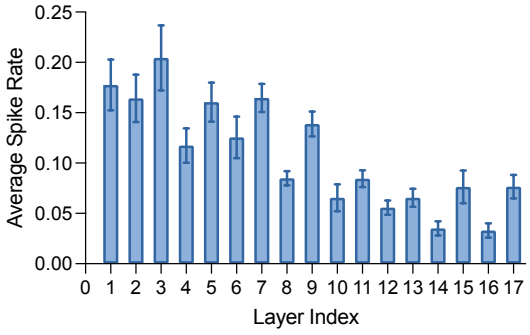


Figure B5 The average spike rate of each layer for the last task under the CIFAR100-B0-10steps setting.

Table B7 Energy consumption comparison.

	DER [8]	Ours	Ours w/ SA
Avg(%) ↑	75.36	76.85	76.48
Last(%) ↑	65.22	68.34	68.78
MAC(M)	5554.69	18.21	18.21
AC(M)	0.00	2372.37	2224.15
Energy(mJ) ↓	25.55	2.22	2.09

$$N_{\mathrm{AC}} = \sum_{t=1}^{T} \sum_{l=1}^{L-1} \sum_{i=1}^{N^l} f_{\mathrm{out}}^l s_i^l[t], \tag{B3}$$

where L is the total number of layers, N^l is the number of neurons in layer l, and f_{out} is the number of outgoing connections.

For ANNs, the corresponding synaptic operation N_{MAC} with more expensive multiply-accumulate is defined as:

$$N_{\mathrm{MAC}} = \sum_{l=1}^{L-1} \sum_{i=1}^{N^l} f_i^l. \tag{B4}$$

Following [39], we calculate the average number of AC and MAC operations per sample during forward propagation over a time window of T ($T = 4$). We compare our method with the ANN-based DER, and the results are presented in Table B7. The results show that our method reduces energy consumption 91.32%, demonstrating significantly higher computational efficiency and application potential on neuromorphic hardware that supports sparse spiking computation. Furthermore, the sparse activations (SA) in our model contribute to additional energy savings, enabling the potential for energy-efficient continual learning with only a moderate accuracy trade-off.

It should be noted that the above analysis reflects estimated rather than measured energy consumption. Following the common methodology in the neuromorphic computing literature, our estimation accounts only for the energy of arithmetic synaptic operations (AC and MAC) under a fixed per-operation energy assumption [38], and does not model hardware-dependent factors of real deployment. In practice, on-chip memory access, inter-core data movement, spike routing and communication, static/leakage power, and the specific dataflow and mapping of a given neuromorphic chip can all contribute non-trivially to the total energy, and these overheads vary considerably across hardware platforms. Consequently, the absolute energy values and the reported reduction ratio should be interpreted as an indication of the theoretical efficiency advantage afforded by sparse, event-driven AC-based computation, rather than as a guaranteed end-to-end power saving on any particular device.

Appendix C Discussion

The practical deployment of SNNs on neuromorphic hardware in dynamic environments necessitates effective class-incremental learning methods. However, existing SNN-based CIL methods still lag behind their ANN counterparts, particularly in longer-step scenarios, and the critical issue of task confusion remains underexplored. Motivated by the complementary learning systems theory in neuroscience, and drawing on insights into the brain's rapid adaptability through structured connections and reconfigurable modules, we propose a fast–slow learning framework for class-incremental learning in SNNs. In this framework, the slow learner focuses on acquiring general representations, while the fast learner captures novel knowledge. We integrate the general representations from the slow learner with the features extracted by the fast learner across spatial and temporal scales, and feed the fused representations back to the fast learner to enable rapid adaptation to task changes. Extensive experiments on CIFAR100 and ImageNet100 validate the effectiveness of our approach. Across standard and longer-step settings, our method consistently alleviates task confusion and mitigates catastrophic forgetting, achieving superior performance over existing SNN-based CIL baselines, especially in more challenging longer-step scenarios. Moreover, it reaches accuracy comparable to ANN counterparts while consuming only ∼10% of their energy. These results demonstrate that our framework effectively narrows the performance gap between SNN-based and ANN-based CIL methods while preserving the energy efficiency advantage of SNNs, contributing to the advancement of effective class-incremental learning methods for SNNs.

While experimental results demonstrate the superior performance of our method in class incremental learning, several limitations remain to be addressed. First, extending the proposed framework to more complex and large-scale datasets could further demonstrate its scalability. Second, exploring hardware-efficient optimizations could help narrow the gap between theoretical design and practical deployment on neuromorphic chips. Finally, incorporating additional biologically inspired mechanisms, such as memory consolidation and adaptive plasticity, may further enhance the robustness and adaptability of class-incremental learning in SNNs. Taken together, these directions highlight avenues for future work, and we believe the proposed framework represents a meaningful step toward effective and energy-efficient class-incremental learning for SNNs in dynamic environments.

References

1 Wu Y, Deng L, Li G, et al. Spatio-temporal backpropagation for training high-performance spiking neural networks. Frontiers in neuroscience, 2018, 12: 331
2 Neftci E O, Mostafa H, Zenke F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. IEEE Signal Processing Magazine, 2019, 36: 51–63
3 Fang W, Yu Z, Chen Y, et al. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In: Proceedings of the IEEE/CVF international conference on computer vision, 2021. 2661–2671
4 Lian S, Shen J, Liu Q, et al. Learnable surrogate gradient for direct training spiking neural networks. In: Proceedings of IJCAI, 2023. 3002–3010
5 Pham Q, Liu C, Hoi S. Dualnet: Continual learning, fast and slow. Advances in Neural Information Processing Systems, 2021, 34: 16131–16144
6 Zheng B, Zhou D W, Ye H J, et al. Task-agnostic guided feature expansion for class-incremental learning. In: Proceedings of the Computer Vision and Pattern Recognition Conference, 2025. 10099–10109
7 Rebuffi S A, Kolesnikov A, Sperl G, et al. icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2017. 2001–2010
8 Yan S, Xie J, He X. Der: Dynamically expandable representation for class incremental learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021. 3014–3023
9 Chen T, Kornblith S, Norouzi M, et al. A simple framework for contrastive learning of visual representations. In: Proceedings of International conference on machine learning. PMLR, 2020. 1597–1607
10 Fini E, Da Costa V G T, Alameda-Pineda X, et al. Self-supervised models are continual learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022. 9621–9630
11 Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015
12 Lan W, Cheung Y m, Xu Q, et al. Improve knowledge distillation via label revision and data selection. IEEE Transactions on Cognitive and Developmental Systems, 2025, 17: 1377–1388
13 Olshausen B A, Field D J. Sparse coding of sensory inputs. Current opinion in neurobiology, 2004, 14: 481–487
14 McHugh S B, Lopes-dos Santos V, Gava G P, et al. Adult-born dentate granule cells promote hippocampal population sparsity. Nature neuroscience, 2022, 25: 1481–1491
15 Sarfraz F, Arani E, Zonooz B. Sparse coding in a dual memory system for lifelong learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2023. 9714–9722
16 Maass W. On the computational power of winner-take-all. Neural computation, 2000, 12: 2519–2535
17 Krizhevsky A, Hinton G, et al. Learning multiple layers of features from tiny images. 2009
18 Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database. In: Proceedings of 2009 IEEE conference on computer vision and pattern recognition, 2009. 248–255
19 Fang W, Chen Y, Ding J, et al. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. Science Advances, 2023, 9: eadi1480
20 Zhou D W, Wang F Y, Ye H J, et al. Pycil: a python toolbox for class-incremental learning. SCIENCE CHINA Information Sciences, 2023, 66: 197101
21 Zhou D W, Wang Q W, Qi Z H, et al. Class-incremental learning: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024, 46: 9851–9873
22 Zhou D W, Sun H L, Ning J, et al. Continual learning with pre-trained models: A survey. In: Proceedings of IJCAI, 2024. 8363-8371
23 Hu Z, Li Y, Lyu J, et al. Dense network expansion for class incremental learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023. 11858–11867
24 Douillard A, Ramé A, Couairon G, et al. Dytox: Transformers for continual learning with dynamic token expansion. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022. 9285–9295
25 Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In: Proceedings of International Conference on Learning Representations, 2021
26 Fang W, Yu Z, Chen Y, et al. Deep residual learning in spiking neural networks. Advances in Neural Information Processing Systems, 2021, 34: 21056–21069
27 Cubuk E D, Zoph B, Mane D, et al. Autoaugment: Learning augmentation policies from data. arXiv preprint arXiv:1805.09501, 2018
28 DeVries T, Taylor G W. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552, 2017
29 Hou S, Pan X, Loy C C, et al. Learning a unified classifier incrementally via rebalancing. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019. 831–839
30 Wu Y, Chen Y, Wang L, et al. Large scale incremental learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019. 374–382
31 Zhao B, Xiao X, Gan G, et al. Maintaining discrimination and fairness in class incremental learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020. 13208–13217
32 Douillard A, Cord M, Ollion C, et al. Podnet: Pooled outputs distillation for small-tasks incremental learning. In: Proceedings of European Conference on Computer Vision. Springer, 2020. 86–102
33 Rajasegaran J, Hayat M, Khan S H, et al. Random path selection for continual learning. Advances in neural information processing systems, 2019, 32
34 Huang B, Chen Z, Zhou P, et al. Resolving task confusion in dynamic expansion architectures for class incremental learning. In: Proceedings of the AAAI conference on artificial intelligence, 2023. 908–916
35 Han B, Zhao F, Zeng Y, et al. Enhancing efficient continual learning with dynamic structure development of spiking neural networks. In: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, 2023. 2993–3001
36 Ni W, Shen J, Xu Q, et al. Alade-snn: Adaptive logit alignment in dynamically expandable spiking neural networks for class incremental learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2025. 19712–19720
37 Kornblith S, Norouzi M, Lee H, et al. Similarity of neural network representations revisited. In: Proceedings of International conference on machine learning. PMLR, 2019. 3519–3529
38 Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 2015, 28
39 Wang Z, Zhang Y, Lian S, et al. Toward high-accuracy and low-latency spiking neural networks with two-stage optimization. IEEE Transactions on Neural Networks and Learning Systems, 2023