Rethinking Pretraining as a Bridge From ANNs to SNNs

Yihan Lin, Yifan Hu, Shijie Ma, Dongjie Yu, and Guoqi Li, Member, IEEE

Abstract—Spiking neural networks (SNNs) are known as typical kinds of brain-inspired models with their unique features of rich neuronal dynamics, diverse coding schemes, and low power consumption properties. How to obtain a high-accuracy model has always been the main challenge in the field of SNN. Currently, there are two mainstream methods, i.e., obtaining a converted SNN through converting a well-trained artificial NN (ANN) to its SNN counterpart or training an SNN directly. However, the inference time of a converted SNN is too long, while SNN training is generally very costly and inefficient. In this work, a new SNN training paradigm is proposed by combining the concepts of the two different training methods with the help of the pretrain technique and BP-based deep SNN training mechanism. We believe that the proposed paradigm is a more efficient pipeline for training SNNs. The pipeline includes pipe-S for static data transfer tasks and pipe-D for dynamic data transfer tasks. State-of-the-art (SOTA) results are obtained in a large-scale event-driven dataset ES-ImageNet. For training acceleration, we achieve the same (or higher) best accuracy as similar leaky-integrate-and-fire (LIF)-SNNs using 1/8 training time on ImageNet-1K and 1/2 training time on ES-ImageNet and also provide a time-accuracy benchmark for a new dataset ES-UCF101. These experimental results reveal the similarity of the functions of parameters between ANNs and SNNs and also demonstrate various potential applications of this SNN training pipeline.

Index Terms—Event-driven dataset, neural network (NN) analysis, pretraining technique, spiking NN (SNN), transfer learning.

I. INTRODUCTION

THE development of neural networks (NN) is a hot spot for both academia and industry. Although great success has proved the value of artificial NNs (ANNs), they are divorced from the biological basis of real NNs and depend heavily on high-performance computation. The appearance of brain-inspired SNNs fills in this blank. In 1997, spiking neuron networks based on leaky-integrate-and-fire (LIF) were proposed [3] as the third generation of NN. The modeling of spiking neurons imitates the propagation mechanism of the electrical signal of biological neurons using spikes to transmit information. Spiking neurons have a solid biological basis and spatiotemporal information coding complexity. SNNs are good at event-driven and energy-constrained tasks [4],[5].

Although SNNs can simulate real neurons and show extraordinarily low power consumption, the directly trained SNN models are often too small in scale limited by training algorithms and long training time, and the small-scale networks are often of poor performance, and hard to be applied in real-world applications with a huge amount of data. Some algorithms provide the solution for specific network structures [7], [8], which still have a limitation on applications. Therefore, a more general algorithm for obtaining large-scale SNNs efficiently is urgently needed.

Back to the motivation of SNNs, researchers hope that SNNs could provide a nice simulation of biological NNs. Although the powerful training algorithms endow SNNs with learning ability, the critical problem is that the biological instinct, which is brought by the prior knowledge stored in deoxyribonucleic acid to recognize the world, is not reflected by SNNs. The pretrain techniques inspire us, as many existing well-trained ANN models provide a large amount of prior knowledge. It is also well known that pretraining is an important technique to increase the scale of the model and accelerate the training process. At present, most of the commonly used SNNs have incorporated the parameter structures of ANNs, so researchers have made efforts to reuse the weights of a well-trained ANN to obtain a converted SNN without training [9]. The performance of converted-SNNs is close to the original ANNs, but they are not widely used due to the data constraints (only suitable for static data) and the long inference time. In computer vision tasks, it has been proven that an ANN can help to train an SNN with a similar structure by reusing the weights [10].

Inspired by the results, we design a training pipeline with two branches named pipe-D (pipe-dynamic) and pipe-S (pipe-static) depending on the type of data. We prefer to use pipe-S on static data and pipe-D on dynamic data. These pipelines are further illustrated in Fig. 1. Leaky-integrate-and-analog-fire (LIAF) SNN [1] and spiking-activation-ANN (SANN) are introduced as the intermediaries of parameter transition. Here, the LIAF-SNN is a variant model of LIF-SNN, whose
amplitude of the spikes is represented by continuous value, so it does not suffer from the inaccurate gradient during training as LIF-SNN. The SANN is a kind of ANNs using threshold-based binary activations, which also have the common characteristics of both LIF-SNNs and ANNs.

We believe that the pretrained parameters will help SNNs with difficult tasks, even when the distributions and modes of the data change. Moreover, our experiments prove that such a training pipeline can significantly facilitate the training of SNNs, accelerate convergence, and improve generalization performance without any other tricks. The main contributions of this work are as follows.

1) A new training pipeline (including two branches, i.e., pipe-D and pipe-S) for SNNs is proposed in this article, which uses pretrained ANN models to accelerate the training of SNNs and improve SNNs’ performance. To the best of our knowledge, it is the first attempt to propose a complete pretrain-finetune framework for training large-scale LIF-SNNs and it would become a bridge from ANNs to SNNs, further promoting the application of SNNs.

2) We obtain many high-performance SNN models while significantly reducing the training time consumption in various tasks. State-of-the-art (SOTA) results are provided in this article on ES-ImageNet (52.25%/43.74% for LIAF/LIF-SNN) and we also obtain competitive test accuracy on ImageNet-1K, CIFAR10-DVS, and ES-UCF101 with significant training acceleration, saving several times to dozens of time on different tasks.

3) We also provide the ES-UCF101 dataset, which is a new lightweight event stream video classification dataset for long-time sparse spatial–temporal feature extraction. At the same time, we provide a time-accuracy benchmark on it to test the efficiency of training.

II. RELATED WORK

A. Pretrain Technique and Transfer Learning

Pretrain techniques are often applied in situations where large amounts of generalized data are available, but task-specific domain data are scarce and difficult to support practical training. This technique significantly reduces the training time on specific tasks and improves performance. When the pretrain and finetune processes are conducted on a different dataset, we call it transfer learning. In this article, we adopt the ideas from feature-based transfer learning. In CV tasks, most of the algorithms need a feature extractor. Because some of the low-level features of images are reusable, the optimization of the target task can be facilitated by reusing the learned low-level features from other (large) datasets like ImageNet [11]. Yosinski et al. [12] investigate the effects of freezing parameters of different layers on the transferability and representation ability. In tasks with similar features, adversarial optimization is often applied to extract shared features between source and target tasks by removing domain-specific properties, which enhances the robustness [13].

B. SNN Training Algorithms

ANN-to-SNN conversion can be regarded as the primary method of transferring pretrained ANN models to SNNs.
The early conversion methods use the mean-firing-rate approximation of LIF neurons to approach ANN’s activation value during training [9] and many normalization methods are proposed for better conversion [14], [15]. Nowadays, advanced conversion methods can equip a converted SNN with performance comparable to the corresponding ANN. However, these methods can only be used in the static-data scenario. The long inference time also prevents the converted SNNs from wide applications. The direct training method is another way to get an SNN, where spatiotemporal back-propagation (STBP) [6] is a frequently-used BP-based method. STBP employs gradient approximation to avoid the discontinuity of the spiking activation function and applies backpropagation through time (BPTT) [16] on SNNs. But the inaccurate gradient propagation in the BP-based method makes it difficult to obtain a high-performance LIF-SNN, and the multitime-step data flow slows down the process of gradient propagation, leading to an unbearable long training time.

In order to obtain advanced SNNs more efficiently in various scenarios, we consider adopting STBP and the pretrain technique jointly. There are some early explorations in this field. Wu et al. [10] provide a method that guides the BP process by the gradient of ANN and some spiking characteristics of SNN. The hybrid neural state machine (H-NSM) applies the ANN-SNN hybrid architectures, where the ANN and SNN are trained together using various rules under different conditions [17]. There is also an early trial on transfer learning from continuous rate RNN to spiking RNN using weight scaling technique [18]; however, the finetune step is absent in this work and the NN’s scale is limited. In a word, these attempts are still simple or of limited applications. Rathi et al. [19] provide a more inspiring method that combines spike timing dependent BP with a converted SNN, but is still limited in its applicability and performance, which urges the proposal of a more universal and efficient ANN-to-SNN pretraining framework.

III. METHOD

A. Neuron Models

The LIF-SNN model we utilize is simplified from the continuous LIF model [3]. The LIF model considers the influence of the leaky current of neurons, where the membrane voltage can be described as

\[ U(t + 1) = U(t)e^{-\frac{1}{\tau}} + X(t). \]  

(1)

Here, \( U(t) \) denotes the membrane voltage, \( \tau \) denotes a time coefficient, and \( X(t) \) is the weighted input current. When a neuron fires a spike, its membrane will be reset to 0. To construct a multilayer SNN in a computer, we define the discrete form of the LIF model in the \( t \)th layer as

\[ X^i(t) = W^i \ast O^{i-1}(t) \]  

(2)

\[ U^i(t) = \begin{cases} U^i_{\text{init}} + X^i(t), & t = 0 \\ U^i(t-1)e^{-\frac{1}{\tau}(1-\tau)}(1-O^i(t-1)) + X^i(t), & t > 0 \end{cases} \]  

(3)

\[ O^i(t) = F(U^i(t)) \]  

(4)

where \( W^i \) is the weight matrix and \( O(t) \) is the spike train at timestamp \( t \). \( X \) is the input spike train, for example, \( \{1,0,0,\ldots,1,0,1\} \) and \( T \) is the simulation time. \( U^i_{\text{init}} \) denotes the initial membrane potential at the \( i \)th layer. Rate coding is often used in LIF-SNN, where we will calculate the average fire rate of neurons in the output layers. \( F \) is the spike activation function and we often use a threshold-based step function \( F \) like

\[ O^i(t) = F(U^i(t)) = \begin{cases} 1, & U^i(t) > U_{\text{th}} \\ 0, & \text{others} \end{cases} \]  

(5)

where the \( U_{\text{th}} \) denotes threshold voltage. As it is the only nondifferentiable term in the LIF model, we use the gradient approximation proposed in STBP [6]

\[ F'(U^i(t)) = \begin{cases} \frac{1}{2 \ast \text{lens}}, & U_{\text{th}} + \text{lens} > U^i(t) > U_{\text{th}} - \text{lens} \\ 0, & \text{others} \end{cases} \]  

(6)

LIAF models [1] are proposed in the conception of combining the dynamic features of ANN and SNN. The model can be described by the following:

\[ X^i(t) = W^i \ast O^{i-1}(t) \]  

(7)

\[ U^i(t) = \begin{cases} U^i_{\text{init}} + X^i(t), & t = 0 \\ U^i(t-1)e^{-\frac{1}{\tau}(1-S^i(t-1))} + X^i(t), & t > 0 \end{cases} \]  

(8)

\[ S^i(t) = F(U^i(t)) \]  

(9)

\[ O^i(t) = G(U^i(t)). \]  

(10)

By replacing \( F \) by the continuous activation function \( G \) in (4), which can be chosen as ReLU, sigmoid, or other commonly used functions, LIAF neurons obtain both the accuracy of ANN neurons and the bio-characteristics of SNN neurons. Consequently, LIAF is well suited to be used with pretraining.

The network backbones used in the experiments are ResNets [20]. These different networks maintain the same network structure as much as possible, where we replace the corresponding artificial network layers with similar spiking network layers. In the residual structure, the corresponding multilayer structure is designed specifically for SNNs, which is shown in Fig. 2.

B. Training Pipelines

We design dataset-specific pipelines for the pretrain-finetune paradigm.

1) Pipe-S (ANN → SANN → LIF): When the target dataset is a static image dataset, we first train an ANN and then fine-tune it with spiking activation. After that, the weights from the source SANN are copied to the target SNN and get fine-tuned with the extension of the temporal dimension. Here, the SANN can be treated as an SNN without iterations in time.

2) Pipe-D (ANN → LIAF-SNN → LIF-SNN): For dynamic datasets like the DVS-dataset and the video dataset, we adopt a DVS-to-gray reconstruction method (without extra data) or an additional static source dataset to obtain a pretrained ANN model, then fine-tune an LIAF-SNN...
model on the target dynamic dataset. Finally, we fine-tune LIF-SNN from the LIAF version.

In brief, the training pipeline is based on a pretrained ANN model, then introduces spiking-activation and iterative neuron mechanism serially. **Pipe-D** is more general and also suitable for static datasets and summarized in Algorithm 2. **Pipe-S** is faster, but it is unfriendly to dynamic tasks, which is described in Algorithm 1.

### C. Transfer Process Smoothing

For both **pipe-D** and **pipe-S**, we have introduced different smoothing methods when the activation or some layers are changed. These smoothing methods increase the efficiency of the pipeline.

1) **Progressively Sharpened ReLU**: In the view of the potential obstacle caused by the nondifferentiable spiking activation function in **pipe-S**, we incorporate the second step into the first one and offer a smoother procedure to obtain an SANN. A progressively SReLU [2] is adopted as

\[
F(U) = \begin{cases} 
0, & U < \alpha \\
1, & U > \beta \\
\frac{U - \alpha}{\beta - \alpha}, & \text{others}
\end{cases}
\]  

and we assert that \( \alpha < \beta \) and \( \alpha + \beta = 2U_{th} \). As \( \alpha \) increases linearly from 0 to \( U_{th} \) with training epochs, the function will approach the spiking activation in the end, while its derivative is set unchanged as

\[
F'(U) = \begin{cases} 
\frac{1}{2U_{th}}, & 0 \leq U \leq U_{th} \\
0, & \text{others}
\end{cases}
\]  

2) **Warmup Step**: For transfer learning tasks where network structures are changed (where we often use **pipe-D**), it is recommended to add several warmup (WU) epochs (and usually one epoch is enough) between step 1 and step 2. During the WU epochs, we freeze the parameter of some layers and only retrain the other layers from scratch. The unfrozen layers are often the shallow layers or the layers whose shapes of weight are changed. Without WU epochs, the noise information brought by the randomly initialized layers may disturb the well-initialized layers, which sometimes disrupts the pretrain-finetune process. According to the observation, WU can also

---

**Algorithm 1** Pipe-S for Transfer Learning on Static Data

**Require**: ANN, StaticDataset

**Ensure**: trained LIF-SNN

Pretrain an ANN on StaticDataset with ReLU activation (optional).

for \( e = 0 \rightarrow \text{MaxEpoch} \) do

for sample in StaticDataset do

\( B\text{PandUpdate}(SANN, \text{sample}) \)

end for

update activations of SANN via Eq. (11-12)

end for

Copy the weights of SANN to LIF-SNN.

for \( e = 0 \rightarrow \text{MaxEpoch} \) do

if \( e \leq \text{WaUpEpoch} \) then

Freeze some weights of LIF-SNN. (optional)

else

Thaw all weights of LIF-SNN.

end if

for sample in DynamicDataset do

\( B\text{PandUpdate}(LIF-SNN, \text{sample}) \)

end for

Acc = Validate(LIF-SNN)

Save the best model.

end for

---

**Algorithm 2** Pipe-D for Transfer Learning on Dynamic Data

**Require**: ANN, DynamicDataset

**Ensure**: trained LIF-SNN

Pretrain ANN on StaticDataset.

Copy the weights of ANN to LIAF-SNN.

for \( e = 0 \rightarrow \text{MaxEpoch} \) do

for sample in DynamicDataset do

\( B\text{PandUpdate}(LIF-SNN, \text{sample}) \)

end for

Copy the weights of LIAF-SNN to LIF-SNN

for \( e = 0 \rightarrow \text{MaxEpoch} \) do

if \( e \leq \text{WaUpEpoch} \) then

Freeze some weights of LIF-SNN. (optional)

else

Thaw all weights of LIF-SNN.

end if

for sample in DynamicDataset do

\( B\text{PandUpdate}(LIF-SNN, \text{sample}) \)

end for

Acc = Validate(LIF-SNN)

Save the best model.

end for
IV. EXPERIMENTS

We conduct our experiments in three types of datasets, which are the traditional RGB dataset ImageNet-1K [11], the static-data-converted/recorded dynamic vision sensor (DVS) dataset ES-ImageNet [21], and the CIFAR10-DVS [22], and dynamic-data-converted ES-UCF101, respectively. They are different in the form of data, especially the encoding mechanisms. The latter three datasets share data coding characteristics more compatible with SNNs’ spiking data flow. We have conducted extensive experiments on those datasets and compared the classification accuracy with the SOTA results of SNNs.

A. Experiment 1: Static–Static Experiments

ImageNet-1K (or ILSVRC2012) [23] is a well-known large-scale visual dataset. It contains about 1.3M image samples collected from the real scene with 1000 different categories. We have trained the basic ResNet-18 version CNN/SCNN/LIF-SNN/LIAF-SNN on it using pipe-S and pipe-D. The hyperparameter setting for pipe-D and pipe-S are in Table I.

1) Training Result: For pipe-S, at the beginning of the pretrained stage, the direct transcription of the convolutional NNs (CNNs) weights to its SNN counterpart is of little help, but as the activation evolves into the step function while being finetuned, the corresponding CNN grows into a well-behaved spiking-activation CNN (SCNN). When making inferences with strict spiking activation, the model and its parameters obtain a top-1 accuracy of 50.25% on ImageNet. Then, surprisingly, the accuracy increases to 58.08% after only one epoch of finetuning with extended temporal dimension, which indicates the importance of further adapting the pretrained model to spatial–temporal dynamics. After the subsequent decay of the learning rate, it turns out that only one-tenth of the training epochs are required for the finetune stage at its fastest pace, to achieve the same accuracy as that of the model trained from scratch. A training procedure (e.g., of 30 epochs) with moderate learning rate decay will yield a higher accuracy, as in Table II. Through pretraining to learn the extraction ability of spatial features and finetuning to learn effective representation in spatial–temporal dynamics, our pipe-S provides a much faster and more effective way to train an SNN. More details can be found in the Appendix. We test pipe-D later, and it also shows a significant acceleration effect and boosts the training of LIF-SNN. However, for static tasks, the redundancy of pipe-D may be somehow harmful for gradient propagation and leads to a slightly worse result.

2) Results Analysis: Structural similarity index measure (SSIM) is used to measure the similarity between different weights, which is defined as

\[
L(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \quad (13)
\]

where \(\mu_x\) and \(\mu_y\) are the mean value and the unbiased standard deviation of the data \(x\), respectively. \(L(x, y)\), \(C(x, y)\), and \(S(x, y)\) measure the similarity between two images by luminance, contrast, and structure, respectively. In order to measure the structural similarity between the weights, which are four-dimension tensors having out channels and in channels, we extend the SSIM to mSSIM, which is simply defined as

\[
mSSIM(W_1, W_2) = \sum_{i=0}^{in} \sum_{j=0}^{oc} \frac{\text{SSIM}(W_1[i, j, :, :], W_2[i, j, :, :])}{ic \times oc} \quad (17)
\]

where \(ic\) and \(oc\) are numbers of inChannel and outChannel. For each tensor \(W_1[i, j, :, :]\) and \(W_2[i, j, :, :]\), we choose \(\text{windowsSize} = 3\) to calculate the SSIM, where the window is used to measure local structural similarity. With the
TABLE II
PIPE-S AND PIPE-D EXPERIMENT RESULTS (TOP-1 ACCURACY) ON IMAGENET

| Network Structure | Pretrain Method | Acc. (1st Epoch) | Best Acc. (Epoch) |
|-------------------|----------------|------------------|-------------------|
| SEW-R18(T=4) [5]  | pipe-S step2  | —                | 50.25(100)        |
| pure-LIF-R18(T=6) | pipe-D step2  | 49.83            | 64.04(42)         |
| MS-R18(T=6) [14]  | pipe-D step3  | 58.08            | 62.81(30)         |
| tdBN+LIF-R34(T=6) | pipe-D step3  | 48.27            | 60.32(32)         |
| TET-LIF-R34(T=6)  |               |                  |                   |
| TS-VGG16*(T=16)   |               |                  |                   |
| LIF-R34(T=2500)   |               |                  |                   |
| IF-R34*(T=2500)   |               |                  |                   |

a R18/R34 denoted basic ResNet-18/34 structure.

The visualizaiton of the weight of deeper layers and the comparison among them are provided in Fig. 8 in the Appendix. To give a more rigorous analysis, we also draw the mSSIM curves for each network pair in Fig. 3(b) when these networks converge, which supports our pipelines. For most layers of these networks, the highest and lowest similarities occur in LIF-SNN and LIAF-SNN pair and LIF-SNN and CNN pair, respectively, and the similarity between LIAF-SNN and CNN lies between the two. This is why we employ intermediate LIAF/SANN models in the pipelines, and the gap between LIF-SNN and CNN often leads to training failure, as shown in Fig. 9 in the Appendix. Another observation is that mSSIM curves show an upward trend with the increase of depth, which indicates that it may be necessary to warm the first few layers at the beginning of finetuning.

B. Experiment 2: Static–Dynamic Experiments

ES-ImageNet is an event-stream (ES) classification dataset converted from ImageNet-1K, but their data modes are different. The samples in ES-ImageNet are sequential data of quads \((x, y, t, p)\), where \((x, y)\) is the topological coordinates of the pixel, \(t\) is the time of spike generation, and \(p\) is the ternary polarity of the spike \(\in \{-1, 0, 1\}\). We usually use a two-channel event frame format to structure these data to form event frame stream data, which is similar to video data. The ES-ImageNet consists of about 1.3M ES samples, with each sample having eight-time steps and about a 5% none-zero rate. Its former SOTA result is 47.47% obtained by a LIAF-SNN with ResNet-34 backbone [21]. CIFAR10-DVS is another dynamic dataset, which is converted from CIFAR10 by using a DAVIS camera [29] to record image motions on an LCD monitor [22]. The samples are also organized as \((x, y, t, p)\) quads. The data distribution of CIFAR10-DVS is much different from that of ImageNet or ES-ImageNet. The pipeline would be tested on CIFAR10-DVS using the pretrained models obtained on ImageNet-1K or ES-ImageNet, hoping that the benefits of the pipeline can also be observed on transfer tasks with a larger gap. We choose \(T = 8\) for pipe-D in these experiments, while \(T\) is extended to 20 in one case of the CIFAR10-DVS experiment.

1) Training Result: We conduct an ablation experiment on the dataset with pipe-D. It should be noted that, because there is a change in the number of the data’s channels (RGB frame is three and event frame is two), we introduce a WU step before fine-tuning LIAF-SNN, where we only train the first layer of the network. The training results on ES-ImageNet with ResNet-18 structure can be found in Table III. The experimental results show that such a training pipeline is also effective in some cross-mode tasks and with external data the network can obtain the highest accuracy. Because the cost of training SNN (especially the time) is several times to tens of times larger than that of ANN, the whole pipeline saves a lot of training costs compared with direct training.

In order to facilitate our comparison with the existing work, we use the same network structure as the original paper that proposed the first benchmark of ES-ImageNet. More training details are provided in Table IV.

We then use the pretrain model obtained on ES-ImageNet and ImageNet-1K to apply pipe-D on CIFAR10-DVS. The input size of the sample is resized to \(42 \times 42\) by downsampling for one case and keeping the original \(128 \times 128\) for another. WU is used for the last layer, where the output dimension is changed from \(1000\) to \(10\). The results are shown in Table V. More details can be found in Table VI.

2) Results Analysis: The parameter of three converged models is visualized in Fig. 4, and we can still find similar patterns in the parameters of the first convolutional layer (although the number of input channels is different). That layer of LIAF is initialized randomly in step 2 of pipe-D. The feature maps of three different convolution layers are also visualized in Fig. 5, where the feature maps of SNN are the average value of the output spike-frame sequences, i.e., the rate decoding process. The similarity between feature maps of different networks is intractable to quantify because they are influenced by the data. However, it is still important to inspect if similar weights could generate similar feature maps with the same data. And in experiment 2, the case is more complicated when ANNs are trained with RGB data and SNNs are trained with the converted event data, so it is worthwhile to figure out how many features will be shared by these two data modes and utilized by different NNs at the same time.

According to the visualization of weights in Fig. 4 and feature maps in Fig. 5, there is an interesting phenomenon that the dynamic among these layers is similar. However, it is
Fig. 3. (a) Weight maps of CNN, LIAF-SNN, and LIF-SNN in ImageNet experiment in pipe-D. We place these convolution kernels in the order of OutChannels, where each $7 \times 7$ square contains $[\text{ith OutChannels}, :, :]$. The color of the weight presents the absolute value of the weight, which may be influenced by the membrane update mechanism of the LIF/LIAF neuron in the training process. It should be noted that here we use the same ResNet-18 backbone as the CNNs to construct SNNs. The shortcut connection of the dotted line indicates the change in size of the feature maps. (b) mSSIM curves. mSSIM is to measure the similarity between weights. The mSSIM curves indicate that when these networks converge, there is the highest structural similarity in the weight between LIF-SNN and LIAF-SNN, followed by LIAF-SNN and CNN, and finally, LIF-SNN and CNN. At the same time, the weighted similarity increases with the growth of depth.

| Network¹ | Pretrain Method² | Acc. (1st Epoch) | Best Acc. (Epoch) |
|----------|------------------|------------------|-------------------|
| LIF-R18 [21] | — | — | 42.54(50) |
| LIF-R18 [21] | — | — | 39.89(50) |
| CNN-R18 [21] | — | — | 41.03 |
| LIF-R34 [21] | — | — | 43.42(50) |
| ERC-R18² [28] | — | — | 44.25(30) |
| LIF-R18⁴ | step2 without WU | 14.33 | 46.06(11) |
| LIF-R18⁵ | step2 | 19.74 | 50.54(16) |
| LIF-R18⁶ | step2 | 29.92 | 52.25(16) |
| LIF-R18 | step3 + LIF-18⁷ | 15.60 | 43.47(11) |
| LIF-R18 | step3 + LIF-18⁸ | 17.68 | 43.74(18) |

¹ LIF-R18⁵: The pretrained ANN is obtained on the reconstructed Gray dataset from ES-ImageNet with $Acc = 42.94\%$.
² WU are adopted for pipe-D.
³ ERC-R18²: The pretrained ANN is obtained on ResNet-18.
⁴ All the SNNs’ simulation time is $T = 8$.

For further validation, we also conduct experiments on a real-dynamic video dataset, called UCF101 [32]. There are 13,232 video clips in UCF101, the train-test split is: 9990 — 3330. There is an attempt to convert the RGB-stream dataset to a DVS dataset named UCF101-DVS [33] using DAVIS-camera, and their corresponding method RG-CNN achieved 63.2% validation accuracy. For simplicity and
TABLE IV
HYPERPARAMETER SETTING FOR ES-IMAGENET EXPERIMENTS (PIPE-D)

| Part       | Names     | Value    |
|------------|-----------|----------|
| SNN        | Thresh    | 0.5      |
|            | Decay     | 0.5      |
| SGD+M-step1| weight_decay | 1e-4    |
|            | momentum  | 0.9      |
| ADAM-step2 | weight_decay | 1e-4    |
|            |           |          |
| Activation | lens in Eq.(6) | 0.5      |
|            | ANN/LIAF act | ReLU     |
| StepLR     | N epoch   | 10       |
|            | α         | 0.2      |
| Others     | Max Epoch | 40(early stop) |
|            | Loss      | CELoss   |

TABLE V
PIPE-D TRANSFER LEARNING RESULTS ON CIFAR10-DVS

| Network       | Pretrain Method | Acc. |
|---------------|-----------------|------|
| GCN [50]      |                 | 54.00|
| LIF-SCNN [6]  |                 | 60.50|
| tdbN-RseNet18 (T=10) [7] |        | 67.80|
| Wide-7B-Net (T=8) [8] |       | 70.20|
| PLIF (T=20) [31] |             | 74.80|
| LIF-R18* From scratch |            | 53.64|
| LIF-R18* pipe-D (reuse ImageNet Exp.) |         | 62.89|
| LIF-R18* pipe-D (reuse ES-ImageNet Exp.) |       | 66.02|
| LIF-R18* From scratch |             | 64.17|
| LIF-R18* pipe-D (reuse ES-ImageNet Exp.) |     | 70.52|
| LIF-R18* pipe-D (reuse ES-ImageNet Exp.) |   | 72.50|

* T = 8, event frames are obtained by down-sampling to 42 × 42.
* T = 8, event frames keep the resolution as original 128 × 128.
* + T = 20 with 128 × 128-resolution input.

excellent compatibility with SNNs, we use the software simulation method mentioned in ES-ImageNet [21] and generate a new lightweight ES dataset, named ES-UCF101. Compared with the large volume of UCF101-DVS (28.4 GB in ZIP format), the volume of our ES dataset is relatively small, with only 1.66 GB in ZIP format, which is especially suitable for testing sparse event recognition algorithms. We set the event ratio close to 0.15 by dynamically adjusting the contrast threshold. Here, the event-ratio = (#e_pos + #e_neg)/(H × W), where #e_pos and #e_neg denotes the number of positive and negative events in each event frame, respectively. We also use pipe-D here, and T is selected as 15. More details can be found in the Appendix.

1) Training Result: We implement a video classification algorithm on UCF101 for the ANN case. One CNN encoder (CNN-Enc) for feature extraction and one RNN (long-short term memory (LSTM)) decoder (RNN-Dec) for integrating temporal information is adopted, which is similar to LRCN [34]. In order to alleviate the disturbance of other factors, only RGB images or event frames are fed into the networks, rather than integrating optical flow, which would degrade our overall experimental results but does not affect the demonstration of our pretrain pipeline because here we mainly focus on relative performance gains compared with naively training LIF-SNN. In this experiment, LIF-SNN and LIF-AFN-SNN act as the spatial–temporal feature encoders only. We first train the ANN encoder and decoder on UCF101. And then LIF-CNN encoders and LSTM decoders are finetuned on ES-UCF101 from pretrained models or scratch. Finally, we fine-tune LIF-CNN encoders and LSTM decoders on ES-UCF101 from pretrained LIF-AFN-CNN encoders and LSTM decoders, as well as from scratch for comparison. We have conducted comprehensive comparative experiments. The efficiency and effectiveness of pretraining from ANNs to SNNs on such a real-video dataset could be illustrated by the aforementioned Tables VII and VIII.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 5. Feature maps of a sample in ES-Imagenet/ImageNet generated by ResNet18/Spiking-ResNet18. The intensity of the feature maps reflects the channel’s bias, and the patterns of the convolution kernels determine the specific shapes in output feature maps.

TABLE VII
PIPE-D STEP-2 RESULTS ON ES-UCF101, WITH ✓ FOR USING PRETRAINED MODELS, WHILE ✗ FOR NOT. FOR EPOCH COLUMNS, NUMBERS IN ( ) DENOTE TEST ACCURACY(%) IN CORRESPONDING EPOCHS AND FOR BEST ACC. COLUMN, NUMBERS IN ( ) DENOTE CORRESPONDING EPOCHS

| CNN-Enc | RNN-Dec | epoch-1* | epoch-2* | best Acc.(epoch) |
|---------|---------|----------|----------|------------------|
| ✓       | ✓       | 6(20.42) | 16(41.71) | 47.84(35) |
| ✓       | ✗       | 5(22.46) | 14(40.93) | 52.85(56) |
| ✗       | ✓       | 4(21.35) | 16(41.41) | 47.84(51) |
| ✓       | ✓       | 3(22.76) | 11(41.32) | 50.12(49) |

* epoch-1 denotes the first epoch when test accuracy reaches 20%.

* epoch-2 denotes the first epoch when test accuracy reaches 40%.

2) Results Analysis: Results in Table VII demonstrate that, at step 1 of pipe-D, pretraining CNN-Enc while training RNN-Dec from scratch could accelerate the training and boost the performance of LIAF-Enc on ES-UCF101 by a large margin. Nevertheless, pretraining RNN-Dec, unfortunately, may degrade the results. A possible explanation is that ANNs and SNNs share similar visual feature extraction patterns spatially, but they have different temporal modeling features. More specifically, LIAF and LIF are capable of addressing spatiotemporal features inherently in a collaborative manner, while 2D-CNN can only process visual inputs individually. As Table VIII shows, when transferred from LIAF to LIF, pretraining both LIAF-Enc and RNN-Dec performs better, for LIAF and LIF share a similar spatiotemporal mechanism. However, using the RNN-Dec pretrained with the CNN-Enc at step 1 may degrade the final performance, which may be because the RNN-Dec is highly dependent on the distribution of the output of CNN-Enc. The hyperparameter setting for pipe-D and pipe-S can be found in Table IX.

TABLE VIII
PIPE-D STEP-3 RESULTS ON ES-UCF101, WITH ✓ FOR USING PRETRAINED MODELS AT THIS STEP, WHILE ✗ FOR NOT. FOR EPOCH COLUMNS, NUMBERS IN ( ) DENOTE TEST ACCURACY (%) IN CORRESPONDING EPOCHS AND FOR BEST ACC. COLUMN, NUMBERS IN ( ) DENOTE CORRESPONDING EPOCHS

| Step 1 | Step 2 |
|--------|--------|
| CNN Enc | RNN Enc | LIAF Dec | RNN Dec | epoch-1 | epoch-2 | best Acc. (epoch) |
| ✓       | ✗       | 11(20.00) | ——      | 39.70(55) |
| ✓       | ✗       | 11(21.11) | 47(40.03) | 40.84(49) |
| ✓       | ✓       | 5(20.63)  | 26(40.36) | 45.14(54) |
| ✓       | ✓       | 4(20.99)  | 18(41.56) | 46.94(56) |

No step 1 ✓ ✗ 3(21.71) 10(40.60) 47.72(60)
No step 1 ✗ ✓ 3(21.65) 18(40.15) 46.94(48)
No step 1 ✓ ✓ 1(35.32) 3(40.27) 49.76(47)

V. DISCUSSION
A. Weight Structure and Inference Performance
The experiments analysis in experiment 1 and experiment 2 provides some evidence to support that SNNs and ANNs may depend on similar spatial features (generated by similar weights). Similar weight patterns are found in convolution kernels in SNNs and ANNs, which means that although the mechanisms of neurons are different, the preference for spatial features is similar, and the features extracted by those convolutional kernels would be similar. However, the gap between the LIF neuron and ANN activation still exists.
and brings subtle but crucial differences, so we are bound to find some smoothing method to fill up the gap, finding the balance between speed and performance. The results in experiment 3 show that the ability of ANNs and SNNs to represent high-level features may be different when dealing with spatial–temporal information streams. This phenomenon emphasizes the importance of intermediate models to smooth the training pipeline, especially when SNNs are trained on dynamic datasets.

B. Performance Improvement

As shown in Table II–VIII and the training curves drawn in Fig. 6, the pretrained SNNs can achieve high validation accuracy after only a few training epochs. In Spiking-ResNet18’s experiments on ES-ImageNet, after 16 epochs of retraining, the network can obtain 52.25% validation accuracy, which exceeds all previous experimental records. The LIF-SNN result of 43.74% is also the highest record for LIF-SNNs. Dozens of repeated experiments are conducted to verify if transfer learning is applicable to LIF-SNN with larger changes in modality and distribution of data (RGB images versus real DVS data), as shown in Fig. 6(c). These observations indicate that similar feature extraction ability with high-performance ANN may boost SNNs. The ANN training speed and the ability of feature encoding are adequately utilized in the pipelines.

In experiment 1 and experiment 2, the training of SNN is still improved with the absence of external data. We speculate that previous training methods and structures are not so perfect for giving full play to the capability of SNNs. The inaccurate gradient estimation caused by the limited expression ability of SNN affects the training progress and restricts the parameter space. However, our pipeline takes advantage of the superior parameter space exploration ability of ANNs. In experiment 2, we also introduced external information. Although ES-ImageNet is converted from ImageNet, it loses a large amount of color and illumination information. The pretrained model may provide prior knowledge about this information, which further improves the SNN’s performance. Furthermore, the smoothness of training may affect the final results. The introduction of the WU process improves the effect of the pretrain models, which is apparent in experiment 2.

C. Training Acceleration

More importantly, the convergence speed of this training pipeline is often several times to tens of times faster than that of the direct training process. In experiment 1, we obtain an LIF-ResNet18 in 30 epochs, where the former record for R18-SNN needs 320 epochs to reach that accuracy. In experiments on ES-ImageNet, less than 20 epochs are used to achieve the best result, while the compared work uses 50 epochs for training. As for experiment 3, we can even save 45 (2 versus 47) epochs to get the same performance, which is \( > 20 \times \) faster than the direct training process.

From a more rigorous perspective, we need to calculate the total training time (including each step in the pipeline). To avoid the interference of training settings, we use \( \text{card h/epoch} \) as the normalized time unit of the measurement.
Here, one hour of training using NVIDIA RTX 2080Ti is taken as 1 card-h. We only compare the training cost on the two most time-consuming experiments: on ImageNet and ES-ImageNet. We first calculate the training time at each step of the training pipelines, which can be found in Table X. And then, we compare the total training time of LIF-R18 trained from scratch, the SOTA R18-SNN (SEW-R18 [8] for ImageNet and EC-R18 [28] for ES-ImageNet), in Fig. 7. In experiment 1, the time cost of pipe-S is 41.6% of that of training LIF from scratch, and 12.36% of that of SEW-R18 (about 8× acceleration), whose epochs read from Fig. 3 of the original paper [8] and training speed tested by our reproduction. It should be noted that we use pipe-S with SReLU, so the first and second steps become a continuous process, and the total time cost is the time to obtain the trained SANN. For experiment 2, we need to take all steps into account. Our training time cost using pipe-D is 58.0% of training LIF from scratch, and 41.5% of that of EC-R18 (about 2× acceleration).

The effect of the acceleration is not only limited to a single experiment. We have proved that the pretrained model can be shared along with different tasks by transfer learning experiments (experiment 2), which means most of the repeated pretrain work can be avoided, which actually provides the biggest acceleration for the SNN research. The pretrained models of the most commonly used ANN networks are available since many open-access model libraries are established, while the corresponding LIAF network can also be finetuned easily (and the LIAF pretrained model can also be reused). In a word, we not only provide ANN-to-SNN training pipelines, but also a bridge between the available ANN resources and the advanced SNN research. It encourages us to train more large-scale LIF-SNN models in the future.

D. Limitations

Limited by the computation resources, we are not able to test the super-large-scale pretrained model using this pipeline. Another problem is that the tasks are all CV classification tasks, focusing on feature extraction and high-level features understanding. We hope this technique for LIF-SNN can be extended to more valuable tasks like high-speed detection, tracking, and classification with neuromorphic image data or RGB-neuromorphic hybrid data processing, where there is still lacking either high-quality datasets and high-performance SNN models. In addition, the adjustment of the simulation time of SNN will greatly affect the performance, so the pipeline of fine-tuning needs to be divided into two steps (for both pipe-S and pipe-D). We hope we can build a complete SNN-ANN transfer learning framework in the future, providing analysis of the weights of ANN and SNN in more detail on more tasks to find the similarities and differences in their working mechanism.

VI. CONCLUSION

This work combines the SNNs training algorithm and the pretrain technique, which is, to the best of our knowledge, the first attempt to propose a complete pretrain-finetune framework for training LIF-SNNs. The novel training pipeline can significantly accelerate the convergence speed of SNNs’ training process and may further improve the validation accuracy with extra data provided, especially when training large-scale SNNs on complicated tasks.

The pipelines are tested on three different dataset types, including the RGB image datasets, converted DVS datasets, and directly recorded DVS-dataset. On all the above-mentioned datasets, the algorithm achieves higher convergence speeds than that of direct training. The pipeline provides up to 2× and 8× training acceleration on ES-ImageNet and ImageNet-1K with competitive or better accuracy. When using external data for pretraining, we achieve SOTA results (52.25%) on ES-ImageNet. We also provide a time-accuracy benchmark for our newly created ES-UCF101 dataset.

We plan to apply this hybrid-technique training pipeline to other tasks. This will bring a new prospect for SNNs’ large-scale and practical applications, and open a new door for large-scale SNN training. The code we used is available in the Appendix.

APPENDIX

TRAINING SETTINGS AND RESOURCES

A. Experiments on ImageNet-1K

ILSVRC2012 is a commonly used large-scale image classification dataset, which collects 1281167 RGB images from the ImageNet dataset with 1000 categories and 5000 images for validation.

We visualize the training process of pipe-S and pipe-D in Fig. 6. At the pretraining stage, the sharpened ReLU

---

**TABLE X**

| Tasks | Type | SANN/ANN | LIAF-SNN | LIF-SNN |
|-------|------|----------|----------|---------|
| Exp. 1 ImageNet | pipe-S | 0.767 | — | 8.014 |
| Exp. 2 ES-ImageNet | pipe-D | 0.861 | 13.511 | 13.511 |
| pipe-D | 0.861 | 34.071 | 34.071 |

The unit is card-h/epoch, which indicates the time required to use the maximum memory of one NVIDIA RTX2080Ti GPU for one epoch of training.
Fig. 8. Visualization results of weight in deeper layers of the three different NNs. (SReLU) is chosen as our activation function to provide a smoother approach toward SCNN. In addition to SReLU, the model is tested with strict spiking activation at every epoch of pretraining using the same set of weights, in order to show to what extent the weight can help during different periods. The learning rate is kept constant during this stage. It is because the process of sharpening is linear; otherwise, a decaying rate would result in an incompetent optimizer accompanied by a severe accuracy drop at the latter part of training. In the finetune stage, we choose StepLR as our learning rate scheduler. When the period of learning rate decay is set to two epochs, a tight finetune stage is obtained and it takes only ten epochs for the fine-tuned model to achieve the same accuracy as the model trained from the scratch. When the period of learning is set to ten epochs, the finetuning will be more moderate and the resulting model reports higher accuracy than the original model. In this way, our pipe-S provides a much faster and more effective way to train an SNN.

As the supplementary for Fig. 3, we add more parameter visualization results between CNN, LIAF, and LIF in Fig. 8. We still place these kernels in the order of OutChannels, where each $3 \times 3$ square displays an average intensity of kernels along the inChannel dimension. The results of experiment 1 still hold for the observation. However, the difference in weights of different layers has different effects on the final result, so this observation is only the inspiration for the method, not the proof.

What is more, we have conducted an ablation study on pipe-S, where we use CNN’s weight to initialize the LIF-SNN and fine-tune it. It unfortunately just makes the SNN converge more quickly, as shown in Fig. 9, where the SCNN-initialized SNN has better convergence speed and accuracy. We could find that the SCNN-initialized SNN has more than 50% Acc. without being fine-tuned, which is least affected by the membrane potential mechanism than the other, so we think it is a more robust training process with pipe-S than directly using CNN’s weight. Directly assigning the weight of CNN to SNN often leads to training failure or poor effect, which may indicate that there is still a significant gap between CNN and LIF-SNN and it is why researchers need to design lots of complicated ANN-SNN conversion methods.

B. Experiments on ES-ImageNet and CIFAR10-DVS

ES-ImageNet is a new ES dataset for event-based image classification and validation. The events are generated from the samples in Imagenet-1K using the ODG method, which
is a method that simulates the basic mechanism of event cameras. The method includes a designed image movement, which results in the value changes in the HSV color space and provides spatial gradient information. This dataset solves the problem of lacking a suitable large-scale classification dataset in the SNNs’ research field.

The dataset includes 1.3M samples with 1000 categories, which is almost the same as ImageNet-1K (some samples are wiped out during conversion because of the bad event rates). Samples in this dataset are stored in quadruple \((x, y, t, p)\), where \(x\) and \(y\) are the coordinates of events, \(t\) denotes the timestamps, and \(p\) is the event polarity. This data mode is typical event data, which is similar to the sampling results of a DVS camera.

Because of the sparsity of data and the large scale, this dataset has become a challenging task for neural morphological algorithms. The highest recognition accuracy obtained on it is still lower than 50%, so we hope to use pretraining technology to make a breakthrough.

CIFAR10-DVS is another neuromorphic dataset, which is directly recorded using a DAVIS camera. This kind of camera provides a high time-resolution event data stream, where the time resolution in DVS-CIFAR10 is 1 \(\mu s\). This dataset is converted from the famous RGB-image classification dataset CIFAR10. Researchers design a fixed motion trajectory and move the images along this orbit on an LCD screen, then record the motion using a DAVIS camera. There are only 9000 samples in the training set and 1000 in the validation set; however, due to the low data quality and noise, the best classification accuracy of this dataset is still lower than 80%.

In the experiments on CIFAR10-DVS, one case is to resize the samples into 32 \(\times\) 32 spatial resolution and this operation causes a serious information loss. The curves of Fig. 4 in the main article indicate that without pretraining, the network will suffer from information loss, especially the low-level feature missing, and fail in the training process. More training details are provided in Table VI.

C. Experiments on ES-UCF101

We generate the ES-UCF101 dataset from UCF101 by simply making difference between every two adjacent items of the gray-scale image sequence in each video clip. To fix the length of generated event streams, 31 images are uniformly sampled for each video clip in UCF101, and after making difference, we could get 30 event frames. As a result, we could get one event stream tensor \(E\) per video clip, with the shape of \((C, T, H, W)\) = \((2, 30, 256, 342)\), where \(C\) denotes the channel number (positive and negative channel), \(T\) denotes the length of event frames, and the spatial resolution is \(H \times W\).

Download link: https://drive.google.com/drive/folders/1_dNYxXKOaFlE4DjdL7ERBvBh4Yld8EQh?usp=sharing

Spiking-LRNC and Hyperparameters: We train the encoder and decoder jointly, thus, ADAM is chosen as the optimizer. The length of each frame sequence is 15. To alleviate overfitting, we implement a simple CNN as encoder, which has four convolutional layers with output layer 8, 16, 32, and 64 channels, respectively, and two linear layers: 17920 \(\times\) 1023 and 1024 \(\times\) 512. A one-layer 512 \(\times\) 256 LSTM acts as a decoder, followed by one 256 \(\times\) 101 linear layer for classification. Total training epochs are set to 60, which is enough for convergence.

D. Codes

The codes for the experiments can be found at https://github.com/lyh983012/SNN-ANN-Pretrain.

REFERENCES

[1] Z. Wu, H. Zhang, Y. Lin, G. Li, M. Wang, and Y. Tang, “LIAF-Net: Leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 11, pp. 6249–6262, Nov. 2022.
[2] W. Severa, C. M. Vineyard, R. Dellana, S. J. Verzi, and J. B. Ainnone, “Training deep neural networks for binary communication with the whestone method,” Nature Mach. Intell., vol. 1, no. 2, pp. 86–94, 2019.
[3] W. Maass, “Networks of spiking neurons: The third generation of neural network models,” Neural Netw., vol. 10, no. 9, pp. 1659–1671, 1997.
[4] K. Roy, A. Jaishwal, and P. Panda, “Towards spike-based machine intelligence with neuromorphic computing,” Nature, vol. 575, no. 7784, pp. 607–617, Nov. 2019.
[5] J. Pei et al., “Towards bifurcational general intelligence with hybrid Tianic chip architecture,” Nature, vol. 572, no. 7767, pp. 106–111, 2019.
[6] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, “Direct training for spiking neural networks: Faster, larger, better,” in Proc. AAAI Conf. Artif. Intell., 2019, vol. 33, no. 1, pp. 1311–1318.
[7] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, “Going deeper with directly-trained larger spiking neural networks,” in Proc. AAAI Conf. Artif. Intell., May 2021, vol. 35, no. 12, pp. 11062–11070.
[8] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, “Deep residual learning in spiking neural networks,” in Proc. Adv. Neural Inf. Process. Syst., vol. 34, 2021, pp. 21056–21069.
[9] P. O’Connor, D. Neil, S. C. Liu, T. Delbruck, and M. Pfeiffer, “Real-time classification and sensor fusion with a spiking deep belief network,” Frontiers Neurosci., vol. 7, pp. 178, Oct. 2013.
[10] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, and K. C. Tan, “A tandem learning rule for effective training and rapid inference of deep spiking neural networks,” IEEE Trans. Neural Netw. Learn. Syst., early access, Jul. 21, 2021, doi: 10.1109/TNNLS.2021.3095724.
[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2009, pp. 248–255.
[12] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in Proc. 27th Int. Conf. Neural Inf. Process. Syst., vol. 2, Dec. 2014, pp. 3320–3328.
[13] Y. Ganin et al., “Domain-adversarial training of neural networks,” J. Mach. Learn. Res., vol. 17, no. 1, pp. 2030–2096, 2016.
[14] P. U. Diehl, D. Neil, J. Binns, M. Cook, S.-C. Liu, and M. Pfeiffer, “Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2015, pp. 1–8.
[15] S. Kim, S. Park, B. Na, and S. Yoon, “Spiking-YOLO: Spiking neural network for energy-efficient object detection,” in Proc. AAAI Conf. Artif. Intell., Apr. 2020, vol. 34, no. 7, pp. 11270–11277.
[16] P. J. Werbos, “Backpropagation through time: What it does and how to do it,” Proc. IEEE, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.
[17] L. Tian, Z. Wu, S. Wu, and L. Shi, “Hybrid neural state machine for neural network,” Sci. China Inf. Sci., vol. 64, no. 3, pp. 1–13, Mar. 2021.
[18] R. Kim, Y. Li, and T. I. Sejnowski, “Simple framework for constructing functional spiking recurrent neural networks,” Proc. Nat. Acad. Sci. USA, vol. 116, no. 45, pp. 22811–22820, 2019.
[19] N. Rathi, G. Srinivasan, P. Panda, and K. Roy, “Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation,” in Proc. Conf. Learn. Represent., 2020, pp. 1–14.
[20] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.
[21] Y. Lin, W. Ding, S. Qiang, L. Deng, and G. Li, “ES-ImageNet: A million event-stream classification dataset for spiking neural networks,” Frontiers Neurosci., vol. 15, p. 1546, Nov. 2021.
[22] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, “CIFAR10-DVS: An event-stream dataset for object classification,” Frontiers Neurosci., vol. 11, p. 309, May 2017.
[23] O. Russakovsky et al., “ImageNet large scale visual recognition challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[24] Y. Hu, Y. Wu, L. Deng, and G. Li, “Advancing residual learning towards powerful deep spiking neural networks,” 2021, arXiv:2112.08954.
[25] S. Deng, Y. Li, S. Zhang, and S. Gu, “Temporal efficient training of spiking neural network via gradient re-weighting,” in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–17.
[26] S. Deng and S. Gu, “Optimal conversion of conventional artificial neural networks to spiking neural networks,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–14.
[27] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, “Going deeper in spiking neural networks: VGG and residual architectures,” *Frontiers Neural Netw.*, vol. 13, p. 95, Mar. 2019.
[28] Z. Wu et al., “Modeling learnable electrical synapse for high precision spatio-temporal recognition,” *Neural Netw.*, vol. 149, pp. 184–194, Jan. 2022.
[29] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, “A 240 × 180 130 db 3 μs latency global shutter spatiotemporal vision sensor,” *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct. 2014.
[30] Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatze, and Y. Andreopoulos, “Graph-based object classification for neuromorphic vision sensing,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 491–501.
[31] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, “Incorporating learnable membrane time constant to enhance learning of spiking neural networks,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2661–2671.
[32] K. Soomro, A. R. Zamir, and M. Shah, “UCF101: A dataset of 101 human actions classes from videos in the wild,” 2012, arXiv:1212.0402.
[33] Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatze, and Y. Andreopoulos, “Graph-based spatio-temporal feature learning for neuromorphic vision sensing,” *IEEE Trans. Image Process.*, vol. 29, pp. 9084–9098, 2020.
[34] J. Donahue et al., “Long-term recurrent convolutional networks for visual recognition and description,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 2625–2634.

**Yihan Lin** received the B.E. degree from Tsinghua University, Beijing, China, in 2020, where he is currently pursuing the Ph.D. degree with the Department of Precision Instruments.
His research interests include brain-inspired artificial intelligence, neuromorphic computing, and neuromorphic image sensors.

**Yifan Hu** received the B.S. degree from Tsinghua University, Beijing, China, in 2019, where he is currently pursuing the Ph.D. degree with the Center for Brain-Inspired Computing Research, Department of Precision Instruments.
His current research interests include deep learning and neuromorphic computing.

**Shijie Ma** received the B.E. degree from Tsinghua University, Beijing, China, in 2021. He is currently pursuing the Ph.D. degree in pattern recognition and intelligent systems with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing.
His current research interests include machine learning, deep learning, and pattern recognition.

**Dongjie Yu** received the B.S. degree in automotive engineering from Tsinghua University, Beijing, China, in 2020, where he is currently pursuing the M.S. degree in mechanical engineering.
His current research interests include safe reinforcement learning and its applications in the decision-making and control of robotics and autonomous driving.

**Guoqi Li** (Member, IEEE) received the Ph.D. degree from Nanyang Technological University, Singapore, in 2011.
From 2011 to 2014, he was a Scientist with A*STAR, Singapore. From 2014 to 2022, he was an Assistant Professor and an Associate Professor at Tsinghua University, Beijing, China. Since 2022, he has been with the Chinese Academy of Sciences, Beijing, where he is currently a Full Professor.
His current research interests include brain-inspired intelligence and neuromorphic computing.

Dr. Li was a recipient of the 2018 First Class Prize in Science and Technology of the Chinese Institute of Command and Control, the Top Ten Scientific Advances Award in China selected by the Ministry of Science and Technology, China, as the Backbone of the Team Member, and the 2020 Second Prize of Fujian Provincial Science and Technology Progress Award. He received the Outstanding Young Talent Award of the Beijing Natural Science Foundation in 2021. He has been actively involved in professional services such as serving as a tutorial chair, a publication chair, a track chair, and workshop chair for several international conferences. He is an Editorial Board Member of *Control and Decision* and served as an Associate Editor for *Journal of Control and Decision and Frontiers in Neuroscience*. 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.