Supplementary Materials for: Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks

Wei Fang[1,2], Zhaofei Yu[1,2,3*], Yanqi Chen[1,2], Timothée Masquelier[4], Tiejun Huang[1,2,3], Yonghong Tian[1,2*]
[1]Department of Computer Science and Technology, Peking University, China
[2]Peng Cheng Laboratory, China
[3]Institute for Artificial Intelligence, Peking University, China
[4]Centre de Recherche Cerveau et Cognition (CERCO), UMR5549 CNRS - Univ. Toulouse 3, France
{fwei, yuzf12, chyq}@pku.edu.cn, timothee.masquelier@cnrs.fr, {tjhuang, yhtian}@pku.edu.cn

1. Reproducibility

All experiments are implemented by SpikingJelly [4]. All of the source codes, training logs are available on GitHub. To maximize reproducibility, we use identical seeds in all codes.

2. Network Structure Details

Tab. S1 illustrates the details of the network structures for different datasets. *c128k3s1* represents the convolutional layer with $output\ channels = 128$, $kernel\ size = 3$ and $stride = 1$. *BN* is the batch normalization. *MPk2s2* is the max-pooling layer with $kernel\ size = 2$ and $stride = 2$. *PLIF* is the PLIF spiking neurons layer. *DP* represents the dropout layer [9]. *FC2048* represents the fully connected layer with $output\ features = 2048$. The symbol $\{\}*$ indicates the repeated structure. For example, $\{$*c128k3s1-BN-PLIF-MPk2s2*$\}$*2 means that there are two $\{$*c128k3s1-BN-PLIF-MPk2s2*$\}$ modules connected sequentially. The last layer *APk10s10* is the voting layer, which is implemented by an average-pooling layer with $kernel\ size = 10$ and $stride = 10$.

*Corresponding author

Dataset	Network Structure
*MNIST	{c128k3s1-BN-PLIF-MPk2s2}*2-DP-FC2048-PLIF-DP-FC100-PLIF-APk10s10
CIFAR-10	{{c256k3s1-BN-PLIF}*3-MPk2s2}*2-DP-FC2048-PLIF-DP-FC100-PLIF-APk10s10
CIFAR10-DVS	{c128k3s1-BN-PLIF-MPk2s2}*4-DP-FC512-PLIF-DP-FC100-PLIF-APk10s10
DVS128 Gesture	{c128k3s1-BN-PLIF-MPk2s2}*5-DP-FC512-PLIF-DP-FC110-PLIF-APk10s10

Table S1. Detailed network structures for different datasets. **MNIST* represents MNIST, Fashion-MNIST, and N-MNIST datasets.

3. Training Algorithm to Fit Target Output

After defining the derivative of the spike generative process, the parameters of SNNs can be trained by gradient descent algorithms as that in ANNs. Classification, which is the task in this paper, as well as other tasks for both ANNs and SNNs, can be seen as optimizing parameters of the network to fit a target output when given a specific input. The gradient descent algorithm for SNNs to fit a target output is derived in the main text (Eq. (16) and Eq. (17)), and is as follows:

Algorithm S1 Gradient Descent Algorithm for SNNs to Fit Target Output

Require: learning rate ϵ, network's parameter $\boldsymbol{\theta}$, total simulating time-steps T, input $\boldsymbol{X} = \{\boldsymbol{X}_0, \boldsymbol{X}_1, ..., \boldsymbol{X}_{T-1}\}$, target output $\boldsymbol{Y} = \{\boldsymbol{Y}_0, \boldsymbol{Y}_1, ..., \boldsymbol{Y}_{T-1}\}$, loss function $L = \mathcal{L}(\boldsymbol{O}, \boldsymbol{Y})$
initialize $\boldsymbol{\theta}$
create an empty list $\boldsymbol{S} = \{\}$
for $t \leftarrow 0, 1, ...T-1$
  input $\boldsymbol{X}_t$ to network, get output spikes $\boldsymbol{S}_t$
  append $\boldsymbol{S}_t$ to $\boldsymbol{S} = \{\boldsymbol{S}_0, \boldsymbol{S}_1, ..., \boldsymbol{S}_{t-1}\}$
calculate loss $L = \mathcal{L}(\boldsymbol{Y}, \boldsymbol{O})$
update parameter $\boldsymbol{\theta} = \boldsymbol{\theta} - \epsilon \cdot \nabla_{\boldsymbol{\theta}} L$

Here the loss function $L = \mathcal{L}(\boldsymbol{O}, \boldsymbol{Y})$ is a distance measurement between $\boldsymbol{Y}$ and $\boldsymbol{S}$, e.g., the mean squared error (MSE) in the main text.

4. Introduction of the Datasets

MNIST The MNIST dataset of handwritten digits comprises 28×28 gray-scale images which are labeled from 0 to 9. The MNIST dataset includes 60,000 training images and 10,000 test images.

Fashion-MNIST Similar to the MNIST dataset, the Fashion-MNIST dataset consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example in the Fashion-MNIST dataset is a 28×28 gray-scale image with a label from 0 to 9.

CIFAR-10 The CIFAR-10 dataset consists of 60,000 natural images in 10 classes, with 6,000 images per class. The number of the training images is 50,000, and that of the test images is 10,000.

N-MNIST The Neuromorphic-MNIST (N-MNIST) dataset is a spiking version of the MNIST dataset recorded by the neuromorphic sensor. It was converted from MNIST by mounting the ATIS sensor on a motorized pan-tilt unit and moving the sensor while recording MNIST examples on an LCD monitor. It consists of 60,000 training examples and 10,000 test examples.

CIFAR10-DVS The CIFAR10-DVS dataset is the neuromorphic version of the CIFAR-10 dataset. It is composed of 10,000 examples in 10 classes, with 1000 examples in each class. As the CIFAR10-DVS dataset does not separate data into training and testing sets, in each class, we choose the first 9000 samples for training and the rest 1000 samples for testing, which is similar to [16].

DVS128 Gesture The DVS128 Gesture dataset is recorded by a DVS128 camera, which contains 11 kinds of hand gestures from 29 subjects under 3 kinds of illumination conditions.

5. Preprocessing

Static Datasets. We apply data normalization on all static datasets to ensure that input images have zero mean and unit variance. Besides, random horizontal flipping and cropping on MNIST and CIFAR-10 are conducted to avoid overfitting. We do not use these augmentations on Fashion-MNIST as images in this dataset are tidy.

Neuromorphic Datasets. The data in neuromorphic datasets usually take the form of address event representation (AER) $E(x_i, y_i, t_i, p_i)$ $(i = 0, 1, ..., N-1)$ to represent the event location in the asynchronous stream, the timestamp, and the polarity. As the number of events is large, e.g. more than one million in CIFAR10-DVS, we split the events into T slices with nearly the same number of events in each slice and integrate events to frames. The new representation $F(j, p, x, y)$ ($0 \leq j \leq T-1$) is the summation of event data in the j-th slice:

$$F(j, p, x, y) = \sum_{i=j_l}^{j_r-1} \mathcal{I}_{p,x,y}(p_i, x_i, y_i), \tag{S1}$$

where $\mathcal{I}_{p,x,y}(p_i, x_i, y_i)$ is an indicator function and it equals 1 only when $(p, x, y) = (p_i, x_i, y_i)$. j_l and j_r are the minimal and the maximal timestamp indexes in the j-th slice. $j_l = \lfloor \frac{N}{T} \rfloor \cdot j$, $j_r = \lfloor \frac{N}{T} \rfloor \cdot (j+1)$ if $j < T-1$ and N if $j = T-1$. Here $\lfloor \cdot \rfloor$ is the floor operation. Note that T is also the number of time-steps in our experiments.

Similar event-to-frame integrating methods for preprocessing neuromorphic datasets are widely used in both ANNs [6, 13, 12] and SNNs [16, 17, 1, 19, 6, 7, 10]. T of [16] in Tab. 3 of the main text for N-MNIST and CIFAR10-DVS are calculated manually according to their paper. Specifically, they illustrate that the time resolution is reduced by accumulating the spike train within every 5 ms and the time range (us) of N-MNIST and CIFAR10-DVS are [290901, 315348] and [1149758, 1459301], respectively.

6. Hyper-Parameters

We use the Adam [8] optimizer with the learning rate 0.001 and the cosine annealing learning rate schedule [11] with $T_{schedule} = 64$. The *batch size* is set to 16 to reduce memory consumption. The drop probability p for dropout layers is 0.5. The clamp function for PLIF neurons is $k(a) = \frac{1}{1+e^{-a}}$ and the surrogate gradient function is $\sigma(x) = \frac{1}{\pi} \arctan(\pi x) + \frac{1}{2}$, thus $\sigma'(x) = \frac{1}{1+(\pi x)^2}$. We set $V_{reset} = 0$ and $V_{th} = 1$ for all neurons. We notice that some previous works, e.g., [15], [16], fine tuned V_{th} for different tasks, which is unnecessary. To be specific, as $\Theta(V - V_{th}) = \Theta(V_{th}(\frac{V}{V_{th}} - 1)) = \Theta(\frac{V}{V_{th}} - 1)$ and V is directed influenced by trainable weights, setting $V_{th} = 1$ implements an implicit normalization for weights, which can mitigate the exploding and vanishing gradient problem. As discovered by Zenke and Vogels [23], ignoring the neuronal reset when computing gradients by detaching them from the computational graph can improve performance, we also detach S_t in the neuronal reset.

7. Accuracy with a Validation Set

The performance comparison in Tab. 2 of the main text is obtained by training on the training set, testing on the test set alternately, and recording the maximum test accuracy. Both this paper and the state-of-the-art methods use this way to report performance. However, this kind of accuracy is overestimated. Here we also report the accuracy with

Dataset	Without Validation	15% Validation
MNIST	99.72%	99.63%
Fashion-MNIST	94.38%	93.85%
CIFAR-10	93.50%	92.58%
N-MNIST	99.61%	99.57%
CIFAR10-DVS	74.80%	69.00%
DVS128 Gesture	97.57%	96.53%

Table S2. Accuracy of the proposed method with/without the validation set on different datasets.

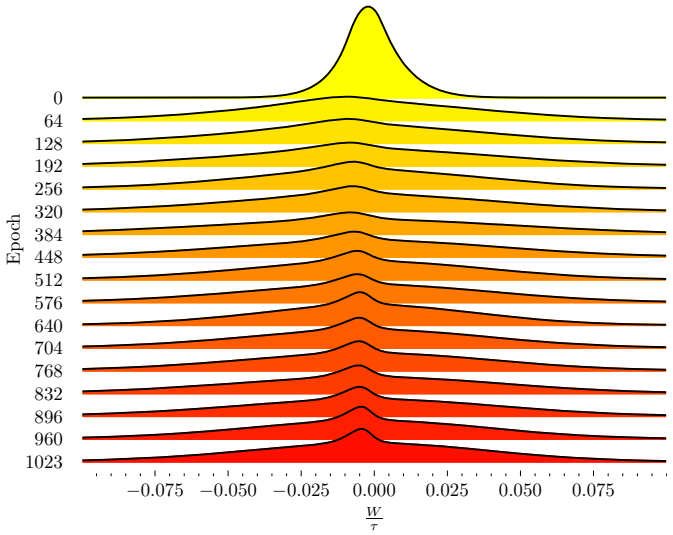


Figure S1. The distribution of $\frac{W_{fc}}{\tau}$ during training the SNN on CIFAR10-DVS.

validation, which is obtained by splitting the origin training set into a new training set and validation set, training on the new training set, testing on the validation set alternately, and recording the test accuracy on the test set only once with the model that achieved the maximum validation accuracy. We utilize 85% samples of each class in the origin training set as the new training set and set the rest 15% as the validation set. The accuracy with and without the validation set of proposed methods is shown at Tab. S2. The experiment results in Tab. 2 of the main text and Tab. S2 show that the proposed method outperforms the state-of-the-art accuracy on nearly all datasets.

8. Distribution of the First $\frac{W_{fc}}{\tau}$

In Sec. 4.3 of the main text, we find that PLIF neurons after the first FC layer are learning to become the Non-Leaky-Integrate-and-Fire neurons as $\frac{1}{\tau} \to 0$ and $\frac{W_{fc}}{\tau}$ converges. To illustrate the convergence, we show the distribution of $\frac{W_{fc}}{\tau}$ during training the SNN on CIFAR-10DVS in Fig. S1. The distribution of $\frac{W_{fc}}{\tau}$ on other datasets converges in the same way.

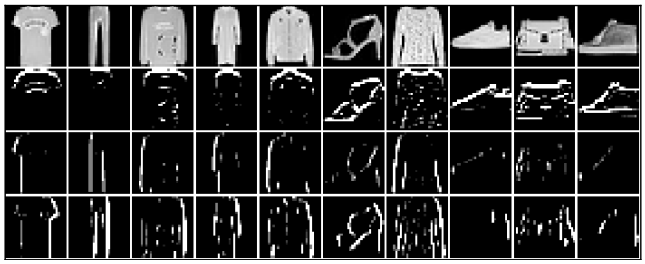


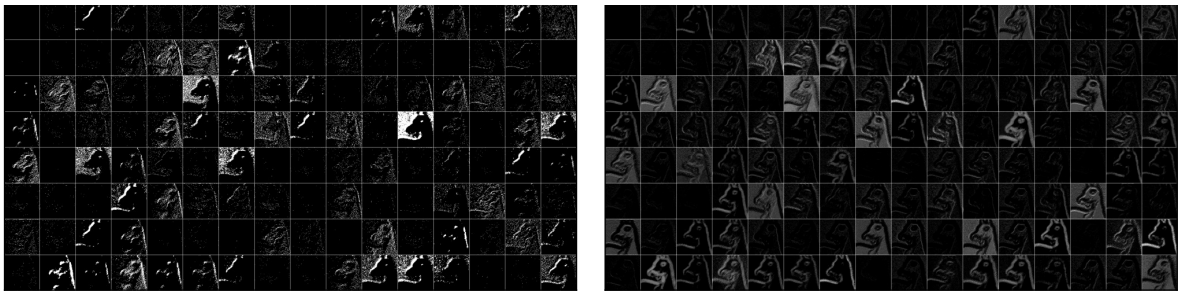
Figure S2. Ten samples from the Fashion-MNIST dataset and the corresponding firing rates $\boldsymbol{F}^2_{T_s=8}$ from channel 45,75 and 76 ($c = 45, 75, 76$) of the first PLIF neurons layer are shown in row 1-4. Each column represents a sample and corresponding firing rates.

9. Visualization of Spiking Encoder

To evaluate the learnable encoder, we give inputs $\boldsymbol{x}_t$ to the trained network and show the output spikes $\boldsymbol{S}^n_t(c)$ and the firing rates $\boldsymbol{F}^n_{T_s}(c) = \frac{1}{T_s}\sum_{t=0}^{T_s-1}\boldsymbol{S}^n_t(c)$ from channel c in the n-th layer, which is similar to [2]. Although the output spikes from deeper spiking neurons layers contain more semantic features, they are harder to read and understand. Thus we only show the spikes from the first spiking neurons layer, that is, $n = 2$.

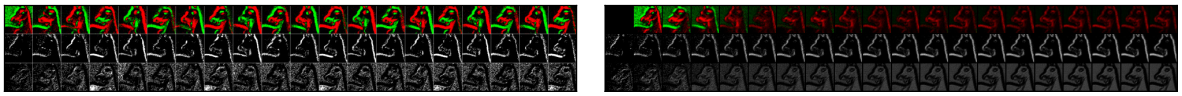
Fig. S2 illustrates 10 input images from static Fashion-MNIST dataset (row 1) and the firing rates $\boldsymbol{F}^2_{T_s=8}$ of three typical channel (45, 75 and 76) of the first PLIF neurons layer (row 2, 3 and 4). One can find that the firing rates from channel 45, 75 and 76 detect upper, left, right edges of the input images. Fig. S3(a) shows a 2-D grid flatten across channels from the 3-D tensor $\boldsymbol{S}^2_{t=0}(c = 0, 1, ..., 127)$ when given an input sample labeled *horse*, which illustrates the features extracted by the spiking encoder at $t = 0$. As the CIFAR10-DVS dataset is converted from the static CIFAR-10 dataset, the firing rates accumulated from spikes can reconstruct the images filtered by the convolutional layer. Fig. S3(b) illustrates the firing rates $\boldsymbol{F}^2_{T_s=19}$ of all 128 channels ($c = 0, 1, ..., 127$), which have clearer texture than binary output spikes in Fig. S3(a). Fig. S4(a) shows the input $\boldsymbol{x}_t$ (row 1) and the corresponding output spikes $\boldsymbol{S}^2_t$ of channel 40 and 103 (row 2 and 3) at $t = 0, 1, ..., 19$, and Fig. S4(b) shows the mean input $\boldsymbol{x}(T_s) = \frac{1}{T_s}\sum_{t=0}^{T_s-1}\boldsymbol{x}_t$ (row 1) and the corresponding firing rates $\boldsymbol{F}^2_{T_s}$ of channel 40 and 103 (row 2 and 3) at $T_s = 0, 1, ..., 19$. One can find that as T_s increases, the texture constructed by firing rates $\boldsymbol{F}^2_{T_s}$ becomes more distinct, which is similar to the use of the Poisson encoder.

Fig. S5 visualizes three input samples $\boldsymbol{x}_t$ and output spikes $\boldsymbol{S}^2_t(c = 59)$ in the DVS128 Gesture dataset. Three samples labeled *random other gestures*, *right hand clockwise*, *drums* at $t = 0, 1, ..., 19$ from the DVS128 Gesture dataset are shown in row 1, 3, 5 of Fig. S5. For comparison, the corresponding output spikes from channel 59 of the PLIF neurons in the first conventional layer are shown



(a) $\boldsymbol{S}^2_{t=0}(c = 0, 1, ..., 127)$ (b) $\boldsymbol{F}^2_{T_s=19}(c = 0, 1, ..., 127)$

Figure S3. Given a sample labeled *horse* from CIFAR10-DVS, (a) shows spikes from all 128 channels of the first spiking neurons layer at $t = 0$, and (b) shows firing rates of these neurons at $T_s = 19$.



(a) $\boldsymbol{x}_t$ and $\boldsymbol{S}^2_t(c = 40, 103)$ at $t = 0, 1, ..., 19$ (b) $\boldsymbol{x}(T_s)$ and $\boldsymbol{F}^2_{T_s}(c = 40, 103)$ at $T_s = 0, 1, ..., 19$

Figure S4. Given the sample sample as Fig. S3, the input data and output spikes of channel 40 and 103 at each time-step are showed in (a) at row 1, 2, 3, respectively. The mean input data and firing rates of channel 40 and 103 at each time-step are showed in (b).

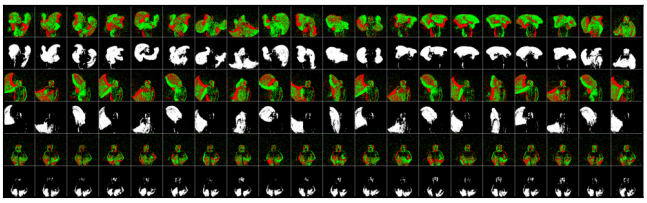


Figure S5. Three samples from the DVS128 Gesture dataset labeled *random other gestures*, *right hand clockwise*, *drums* are shown in row 1, 3, 5. The corresponding output spikes from channel 59 of the first PLIF neurons layer are shown in row 2, 4, 6.

in rows 2, 4, 6. One crucial difference is that the output almost only includes the gesture's response spikes, indicating that the spiking neurons implement efficient and accurate filtering on both spatial-variant and temporal-variant input data, reserving the gesture but discarding the player.

10. Relations between different Encoders

The Poisson encoder is one of the rate encoding methods and widely used in SNNs [3, 9, 14, 24, 1, 5] to encode images into spikes. Given a image pixel $p \in [0, 1]$, the encoded spike S_t at time-step t is fired with the probability p. Thus, the expectation of the number of spikes during the whole time-steps T is $E_{Poisson}(\Sigma_{t=0}^{T-1} S_t) = pT$. In our poposed SNNs, the input is directly fed to the network without being first converted to spikes and the image-spike encoding is done by the first $\{$*Conv2d-Spiking Neurons*$\}$ module (BN is omitted), which can be seen as a learnable encoder. Here we denote this encoder as ENC_l. If we set $Conv2d$ non-learnable with $channels = kernel\ size = 1$, the kernel weight as the constant $w > 0$, and $Spiking\ Neurons$ as Non-Leaky-Integrate-and-Fire neurons with threshold potential V_{th} and $V_{reset} = 0$, then the expectation of spikes number of this module is $E(\Sigma_{t=0}^{T-1} S_t) = \lfloor \frac{T}{\lceil \frac{V_{th}}{wp} \rceil} \rfloor$, where $\lceil\rceil$ denotes the ceiling operation. We can find that $E(\Sigma_{t=0}^{T-1} S_t) \approx E_{Poisson}(\Sigma_{t=0}^{T-1} S_t)$ when $V_{th} = w = 1$, which indicates that ENC_l can approximate the function of the Possion encoder in rate encoding.

The latency encoder used in [18, 20, 21, 22] is a representative temporal encoding method. The latency encoder encodes the image pixel p into a spike at time-step t_p. Thus, the information of input is encoded in the precise firing time of the spike. t_p is usually inversely proportional to the input intensity p, e.g., $t_p = \lfloor (T_{max} - 1)(1 - p) \rfloor$ and T_{max} is the encoding period. We can also find that the first firing time of ENC_l for the given input p is $\lceil \frac{V_{th}}{wp} \rceil$, which satisfies that the larger input intensity p causes the faster spike. In fact, the latency encoder is an extremely simplified learnable encoder with directly inputted images. In this paper, the proposed learnable encoders have learnable weights and more channels, which is able to encode images into complex spikes pattern with more semantic information, e.g., reserving the gesture but discarding the player of samples from DVS128 Gesture dataset (see Fig. S5).

References

[1] Xiang Cheng, Yunzhe Hao, Jiaming Xu, and Bo Xu. LISNN: Improving Spiking Neural Networks with Lateral Interac-

tions for Robust Object Recognition. In *IJCAI*, pages 1519–1525. International Joint Conferences on Artificial Intelligence Organization, 7 2020. 2, 4

[2] Lei Deng, Yujie Wu, Xing Hu, Ling Liang, Yufei Ding, Guoqi Li, Guangshe Zhao, Peng Li, and Yuan Xie. Rethinking the performance comparison between SNNS and ANNS. *Neural Networks*, 121:294–307, 2020. 3

[3] Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9:99, 2015. 4

[4] Wei Fang, Yanqi Chen, Jianhao Ding, Ding Chen, Zhaofei Yu, Huihui Zhou, Yonghong Tian, and other contributors. Spikingjelly. https://github.com/fangwei123456/spikingjelly, 2020. 1

[5] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. RMP-SNN: Residual Membrane Potential Neuron for Enabling Deeper High-Accuracy and Low-Latency Spiking Neural Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 13558–13567, 2020. 4

[6] Weihua He, YuJie Wu, Lei Deng, Guoqi Li, Haoyu Wang, Yang Tian, Wei Ding, Wenhui Wang, and Yuan Xie. Comparing SNNs and RNNs on Neuromorphic Vision Datasets: Similarities and Differences. *arXiv preprint arXiv:2005.02183*, 2020. 2

[7] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic Plasticity Dynamics for Deep Continuous Local Learning (DECOLLE). *Frontiers in Neuroscience*, 14:424, 2020. 2

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[9] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 14, 2020. 1, 4

[10] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10:508, 2016. 2

[11] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 2

[12] Daniel Neil and Shih-Chii Liu. Effective sensor fusion with event-based sensors and deep network architectures. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2282–2285. IEEE, 2016. 2

[13] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences, 2016. 2

[14] Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in Neural Information Processing Systems*, 31:1412–1421, 2018. 4

[15] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12:331, 2018. 2

[16] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1311–1318, 2019. 2

[17] Yujie Wu, Rong Zhao, Jun Zhu, Feng Chen, Mingkun Xu, Guoqi Li, Sen Song, Lei Deng, Guanrui Wang, Hao Zheng, et al. Brain-inspired global-local hybrid learning towards human-like intelligence. *arXiv preprint arXiv:2006.03226*, 2020. 2

[18] Simei Gomes Wysoski, Lubica Benuskova, and Nikola Kasabov. Fast and adaptive network of spiking neurons for multi-view visual pattern recognition. *Neurocomputing*, 71(13):2563–2575, 2008. Artificial Neural Networks (ICANN 2006) / Engineering of Intelligent Systems (ICEIS 2006). 4

[19] Yannan Xing, Gaetano Di Caterina, and John Soraghan. A new spiking convolutional recurrent neural network (scrnn) with applications to event-based hand gesture recognition. *Frontiers in Neuroscience*, 14:1143, 2020. 2

[20] Qiang Yu, Kay Chen Tan, and Huajin Tang. Pattern recognition computation in a spiking neural network with temporal encoding and learning. *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2012. 4

[21] Qiang Yu, Huajin Tang, Kay Chen Tan, and Haizhou Li. Rapid feedforward computation by temporal encoding and learning with spiking neurons. *IEEE Transactions on Neural Networks and Learning Systems*, 24(10):1539–1552, 2013. 4

[22] Qiang Yu, Huajin Tang, Kay Chen Tan, and Haoyong Yu. A brain-inspired spiking neural network model with temporal encoding and learning. *Neurocomputing*, 138:3–13, 2014. 4

[23] Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *BioRxiv*, 2020. 2

[24] Wenrui Zhang and Peng Li. Spike-train level backpropagation for training deep recurrent spiking neural networks. In *Advances in Neural Information Processing Systems*, pages 7802–7813, 2019. 4