



A Spiking Neural Network Architecture for Object Tracking

Yihao Luo¹, Quanzheng Yi¹, Tianjiang Wang^{1(✉)}, Ling Lin¹, Yan Xu¹,
Jing Zhou^{1,2}, Caihong Yuan^{1,3}, Jingjuan Guo^{1,4}, Ping Feng^{1,5}, and Qi Feng¹

¹ School of Computer Science and Technology,
Huazhong University of Science and Technology, Wuhan 430074, China
tjwang@hust.edu.cn

² School of Mathematics and Computer Science, Jiangnan University,
Wuhan 430056, Hubei, China

³ School of Computer and Information Engineering, Henan University,
Kaifeng 475004, China

⁴ School of Information Science and Technology, Jiujiang University,
Jiujiang 332005, China

⁵ School of Information, Guizhou University of Finance and Economics,
Guiyang 550025, Guizhou, China

Abstract. Spiking neural network (SNN) has the advantages of high computational efficiency, low energy consumption, low memory resource consumption, and easy hardware implementation. But its training algorithm is immature and inefficiency which limits the applications of SNN. In this paper, we propose a SNN architecture named SiamSNN for object tracking to avoid the training problems. Specifically, we propose a more comprehensive parameter conversion scheme with the processes of standardization, retraining, parameter transfer, and weight normalization, in order to convert a trained CNN to a similar SNN. Then we propose an encoder named Attention with Average Rate Over Time (AAR) in order to encoding images to spiking sequences. By using IF model, the accuracy decreases by only 0.007 on MNIST compared to the original method. Our approach applies SNN to object tracking and achieves certain effects, which is a reference for SNN applications in other computer vision areas in the future.

Keywords: Spiking neural network · Object tracking · Conversion · Encoder

1 Introduction

Spiking Neural Network (SNN) is known as the “third-generation neural network”, which simulates the information processing mechanism of biological neurons and has a high degree of bionics. It has become the focus of research in pattern recognition such as image classification. It belongs to the frontier technology research topic in the field of artificial intelligence, and has the advantages

of high computational efficiency, low energy consumption, low resource consumption, easy hardware implementation, etc. It is an ideal choice for researching brain-like calculation and coding strategies. Through the theoretical and applied research on SNN, it is of great significance to promote the development of artificial neural networks. It can also promote the research of edge devices such as new artificial intelligence chips that are not Von Neumann architecture.

At present, there are some preliminary results for SNN research, but its application is still in its infancy, mainly used for handwritten digit recognition, image segmentation, etc., and it is difficult to be applied to complex visual scenes. The key to this problem is that the neuron function in SNN cannot be differentiated and its hard to be trained using traditional backpropagation. However, the training algorithm with too low efficiency cannot overcome the training problem of complex SNN model, which brings a bottleneck to the popularization and application of SNN.

On the other hand, object tracking is an important research in the field of computer vision. It has specific applications in many fields such as autonomous driving, security, behavior recognition and human-computer interaction. In recent years, deep learning models based on Convolutional Neural Network (CNN) [1] and automatic encoder (AE) [2] have made a lot of progress in tracking technology. This is because that the depth model has significant feature extraction capabilities. However, due to its large amount of computation, large resources, and the need to rely on top-level GPU acceleration, these models cannot be applied to edge devices. However, if it can take advantages of computationally efficient and easy hardware implementation in SNN, it is possible to apply the target tracking algorithm to the edge devices. However, SNN has not been applied to object tracking.

Therefore, this paper combines deep neural network with spiking neural network, and constructs an object tracking model based on SNN. We avoid the difficulties of SNN training through a conversion scheme, which can advance the progress of SNN related theoretical research and provide reference for SNN to apply to more computer vision problems in the future. On the other hand, the tracking model based on SNN can reduce the computational resource occupation, reduce the power consumption generated by the calculation and the degree of dependence on hardware when a certain tracking effects is achieved. It provides a new method for the application of complex deep learning techniques to edge devices such as tracking.

2 Related Work

Although deep neural networks are historically brain-inspired, there are fundamental differences in their structure, neural computations, and learning rule compared to the brain [3]. SNN has good biomimetic properties, delivering and processing information with precise spiking sequences. And it has theoretically been shown to have Turing-equivalent computing power [4]. The combination of SNN and current deep learning models includes how to build more complex deep

SNNs, overcome training problems and convert pre-trained model parameters to SNN.

On conversion research Neil et al. [5] studied to convert the model parameters of the trained fully connected neural network to the SNN model weights with similar structure through mapping, and achieved the similar accuracy on MNIST while reducing the power consumption and delay of the model. With the development of CNN, the effect on computer visual tasks is better than fully connected networks, which promotes the research of the combination of CNN and SNN. Xu et al. [6] proposed CSNN structure, it inputs the features extracted by CNN to the classification layer of SNN, achieving an accuracy of 88% on MINST, which verifies the possibility of end-to-end training CNN and SNN. Furthermore, some researchers have tried to convert the weighted CNN structure into a SNN structure, it can be used with less operation and consume less energy [7], which can apply CNN to edge devices. Diehl et al. [8] used weight normalization to improve the architecture for reducing performance loss. Rueckauer et al. [9] proposed several conversion criteria to support the biasing of the original CNN and maximal pooling layer converted to SNN, and they tried to identify targets that are more difficult than MNIST (e.g. CIFAR-10 [10] and ImageNet [11]). However, this conversion method is complicated and limited by the specific neuron model, and it is difficult to apply it in SNN of a different neuron model. [12, 13] studies how to convert deeper depth CNN structures such as ResNet [14] to SNN.

On object tracking algorithms based on deep feature similarity have recently made significant progress. They can use a large amount of training data for offline learning, and these models can achieve high accuracy [15]. SiamFC [16] uses full convolution and similarity learning to solve the tracking problem, and finally determines the location of the object through the response score map. In this paper we propose a tracking model with SNN based on SiamFC. SiamRPN [17] introduced region proposal network (RPN) [18] in the field of object detection, avoiding multi-scale testing through network regression, it improves the speed while directly obtaining a more accurate target position through the regression of RPN. DaSiamRPN [19] proposed a distractor-aware feature learning scheme based on SiamRPN which significantly improves the discriminative power of the network, it obtains state-of-the-art accuracy and speed.

Some experimental results show that SNN methods have similar effects as the traditional deep learning methods, but SNN usually requires less operation [9]. In addition, SNN is a structure that mimics the human brain and has a potential to perform better than traditional neural network in the future, it has important research value.

3 Conversion Scheme

3.1 Background

The spiking neuron model used for our work is the integrate-and-fire (IF) model. The membrane potential $V(t)$ of a spiking neuron in the SNN architecture is

updated at each time step by the following equation:

$$V(t) = V(t - 1) + L + \sum_{n=1}^N w_n p_n(t) \tag{1}$$

If $V(t) \geq \theta$, spike and reset $V(t) = 0$
 If $V(t) < Vmin$, reset $V(t) = Vmin$

We save the last membrane potential $V(t - 1)$, and then calculate the current $V(t)$ corresponding to the voltage by the current spike. $p_i(t)$ represents the upper layer of neurons output spike which is present at the current time, the value is 1 when it produces a spike, and the value is 0 when there is no spike. Where L is the constant parameter, $\sum_{n=1}^N w_n p_n(t)$ is the summed input at time t from all synapses connected into the neuron. Whenever $V(t)$ exceeds the voltage threshold θ , the neuron fires and produces a spike (output 1), and its membrane potential $V(t)$ is reset to zero. The membrane potential V is not allowed to go below its resting state $Vmin$ which is usually set to 0, but it can be also changed to allow V to go negative. The parameters used in our simulation can be found in Sect. 5.

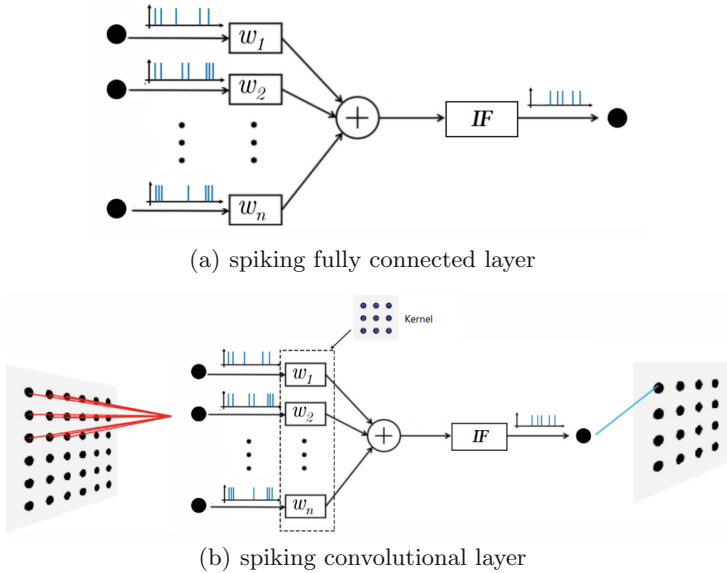


Fig. 1. How the spiking fully connected layer and spiking convolutional layer work.

As shown in Fig. 1, spiking fully connected layer has a very similar form to the fully connected layer in the traditional neural network. The latter directly inputs the result of the operation into the activation function to generate the output value of the neuron, and the former needs to accumulate the IF neuron

membrane voltage continuously during the simulation time. The IF neurons are equivalent to the activation function in the traditional neural network, which increase the ability of neurons by nonlinear processing.

In the convolution calculation, the same weight matrix is used for the different receptive field regions in an image. Spiking convolution layer also uses this strategy to achieve spiking calculation. On each input image, the local spiking feature map of each receptive field region are calculated convolution operator. Therefore, the calculation method is equivalent to fully connected layer. Membrane voltage of the IF neuron is cumulatively calculated, and then it obtains a spiking feature map.

3.2 Challenges in Conversion

In the specific calculation, the SNN is determined to be different because the input is a spiking sequence. But the parameters can be the same as CNN. In theory, the similarity between structure and parameters can support the conversion from CNN to SNN. But directly using parameter transfer have many problems and challenges, which will cause excessive loss of accuracy [7]:

- (1) Negative output values:
 - a. The activation function $\tanh()$ has output values between -1.0 and 1.0
 - b. Weights and biases can be negative may causing the output value to be negative
 - c. preprocessing may produce negative values.
- (2) Representation problems:
 - a. The biases in each convolution layer can be positive or negative, which cannot be represented easily in SNN.
 - b. Max-pooling requires two layers of spiking networks. This approach requires more neurons and can cause accuracy loss due to the added complexity.
 - c. Softmax layer, batch normalization layer (BN), local response normalization layer (LRN) cannot be represented directly.

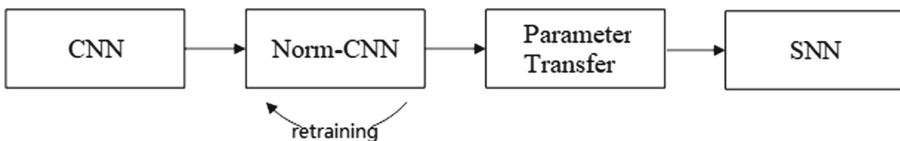


Fig. 2. Flow-diagram of converting a CNN into SNN architecture.

To solve these problems from converting CNN to SNN, we need two steps as Fig. 2:

- (1) Model normalization: Modify the structure of the CNN to become the specific structure of the Norm-CNN, and then retrain it.
- (2) Parameter Transfer: Transfer the parameters of Norm-CNN to SNN with weight normalization.

3.3 Model Normalization

We need process CNN into a similar form (Norm-CNN) by some operations for converting to SNN easily. These operations include:

- (1) Remove biases from all convolution and the fully connected layers, the kernel size and initialization settings are unchanged.
- (2) Where the original activation function is used, it is replaced with the ReLU function, in order to avoiding the negative numbers and reducing the loss of precision after conversion. If the original structure is not activated after the convolutional layer or fully connected layer, the ReLU layer needs to be added later.
- (3) If the network uses a single-spike output neuron, the pooling layer maintains the original Max-Pooling layer or the Average-Pooling layer. If a multi-spike output neuron is used, the pooling layer needs to modify the Max-Pooling layer to the Average-Pooling layer.
- (4) Expect the output layer, we use L2 regularization during training in order to accelerating the convergence of weights to a smaller range and avoiding model overfitting.
- (5) Remove LRN, BN, etc. layer that cannot be directly represented in SNN. Meanwhile, to avoid the model doesnt converge during training, the input image needs to be normalized in a positive range.
- (6) Model compression can be performed by converting weights to 16-bit floats.

Studies have shown that in a convolutional neural network, the use of a 16-bit floating-point type can achieve the same effect as a 32-bit floating-point type [20]. The simulation of the network computing process on the GPU, with the underlying support of CUDA, is twice as fast as the 32-bit floating point type. In addition, the model is quantized and compressed to reduce the bit width, which is more conducive to hardware implementation, breaking the limitation of hardware on type accuracy and size.

Because we modified the network structure, we should retrain the model to get the parameters.

3.4 Parameter Transfer

Because of the similarity between the Norm-CNN and SNN, the activation function in the original model is replaced with the IF neurons. Other parts of the SNN model can transfer the trained parameters.

After that, the SNN model can perform feature extraction or classification, but it may not achieve the desired result. This is because the original weight is matched to the discrete eigenvalues instead of the spiking value (e.g. 0 or 1) of the previous layer. In the network with voltage threshold obtained by converted, when the forward process calculation is performed according to Eq. 1, it is likely that the result of the accumulated membrane voltage will far exceed the threshold value, or cannot be reached at all, resulting in high spiking rate or very low spiking rate [8].

In order to perform spiking activation more effectively, we fixedly set the membrane voltage threshold of each neuron in the network to 1, and the resting potential to 0. The weights are normalized so that the membrane voltage threshold and weight can be adapted for producing spiking properly [7]. This process requires the participation of some training samples, but does not require labels.

If the weight of the l th layer in the network is W^l , we can get all the output values greater than 0 in this layer by inputting the samples into Norm-CNN, and sort them from small to large. Then we select the K th (often set to 99.9%) one as the scaling factor λ^l of this layer [9], and the new weight is calculated according to Eq. 2. This completes the normalized calculation process and sets the normalized parameters as final parameters.

$$W^l \rightarrow W^l \cdot \frac{\lambda^{l-1}}{\lambda^l} \quad (2)$$

Weight normalization is an optional operation, and there are different ways of dealing with complex networks or different network model. In addition, K can also be adjusted according to the specific application effect.

Before applying the converted SNN model to a specific problem such as classification, the following operations are required:

- (1) Encoding: Input image should be encoded to become a spiking sequence before forward calculation.
- (2) Output: The output spiking sequence should be processed. Since the Softmax layer is removed, it is necessary to count the output result obtained at each moment of the last layer, calculate the total number of spiking in each neuron. Then output the classification category according to the category of the neuron which produces the maximum spiking numbers.

Now the entire CNN to SNN conversion process is complete. SNN model can be used to process visual tasks through the conversion method which indirectly overcome the problem of end-to-end training inefficiently. Its possible to promote SNN to more complex fields.

4 SiamSNN: A SNN Architecture for Tracking

4.1 AAR: A Spiking Encoder

SNN is different from CNN, the image needs to be encoded into a spiking sequence before input. There are two main types of spiking coding methods in terms of temporal coding and rate coding [21]:

- (1) Temporal coding
The pixel value is encoded with the precise firing time of the information. The specific time at which pixel should be spiking is determined according to the size of the pixel value and the total encoding time. The encoding process in this way is relatively fast, but only produces a single spiking sequence.

The spiking sequence is too sparse, so that membrane voltage may not be accumulated to reach the threshold when calculating, which has a great influence on the result. However, the sparse spiking sequence can generate facilitates the training algorithm to determine the order of the precise firing timing of the spiking, which makes weight adjustment easier. Therefore, the temporal coding mechanism is suitable for the case of end-to-end training SNN.

(2) Rate coding

The pixel value is encoded with the average firing rate of the neurons. This scheme presupposes that the information content is hidden at the rate of spiking. It is necessary to count the number of spiking generated in a certain period to determine the rate of the spiking sequence for obtaining the result. The coding efficiency of this method is relatively low, and each pixel point is spiking at the first moment by default, so the firing time is not accurate. However, the densely distributed spiking sequence is suitable for the converted SNN, which is beneficial to accumulating membrane voltage to reach the threshold in time and realizing the forward transmission of information.

Because we use the converted SNN, this paper proposes a new method based on the rate coding scheme named Attention with Average Rate Over Time (AAR) in order to improve the effect of the converted SNN.

For the input image, the average rate coding scheme is calculated according to Eqs. 3 and 4. $P_{i,j}$ is the value of each pixel in the image, the maximum pixel value in the whole image is $Pmax$, the minimum pixel value is $Pmin$, the total spiking time is T . And the maximum number of spiking is S , which is produced by the pixel with $Pmax$ (The maximum spiking rate equals S/T). The number of spiking for each pixel is $s_{i,j}$, and the corresponding rate is $f_{i,j}$. The result spiking sequence for each pixel is averaged over the total spiking time T by rate $f_{i,j}$.

$$s_{i,j} = 1 + (S - 1) \cdot \frac{P_{i,j} - Pmin}{Pmax - Pmin} \quad (3)$$

$$f_{i,j} = \frac{T}{s_{i,j}} \quad (4)$$

The ability of SNN to extract features is weak relatively, and the original image may have some noise, which affects the effect of the model in applications. Therefore, we convolve the original image to extract the edge features, which can extract better spiking convolution features and suppress the noise. Meanwhile, since the maximum value of the pixel is too large, the proportion of pixels that can reach the maximum number of spiking is small, resulting in information is easily lost. So AAR does the following operations:

- (1) Pre-convolution: Use $w = \begin{bmatrix} -2 & 1 & -2 \\ 1 & 5 & 1 \\ -2 & 1 & -2 \end{bmatrix}$ 3×3 receptive field filter to convolute the original image to obtain a feature map. In a specific application, the size and value of the filter can be adjusted according to the effect.
- (2) Attention processing: Set the maximum eigenvalues for the top 20% of the eigenvalues in all the eigenvalues of the feature map. This will ensure enough maximum spiking rate.
- (3) The encoding of each pixel is performed according to Eqs. 3 and 4, then the final spiking map is obtained (Fig. 3).

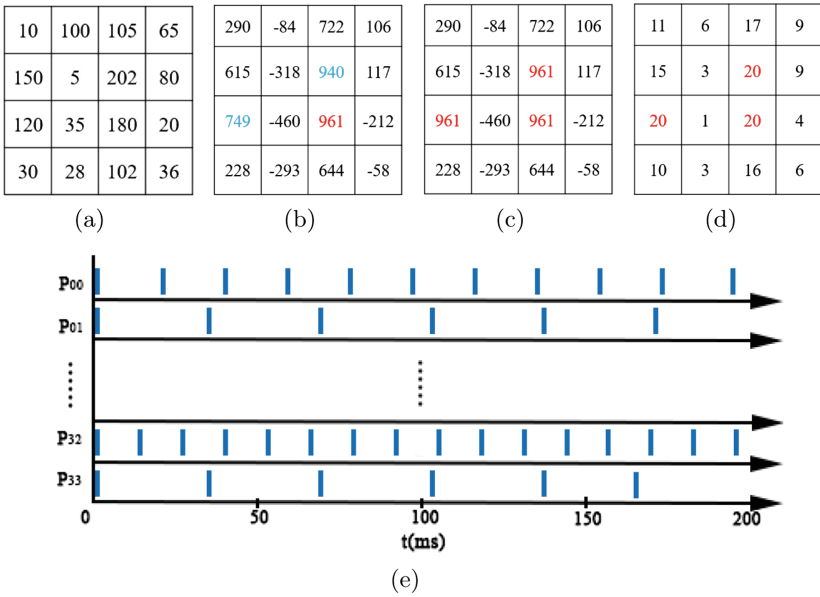


Fig. 3. AAR Calculation schematic diagram. (a) is original image, assume that the maximum number of spiking $S=20$, the total spiking time $T=200$. After the first AAR convolution, the result is shown in (b). After the attention processing, the result is shown in (c). The number of spiking for each pixel is shown as (d). The final spiking coding result is shown in (e).

4.2 SiamSNN Construction

According to the above research, we propose an object tracking model SiamSNN. It is based on SiamFC [16].

As shown in Fig. 4 the architecture is fully-convolutional with respect to the input image. The output is a scalar-valued score map whose dimension depends

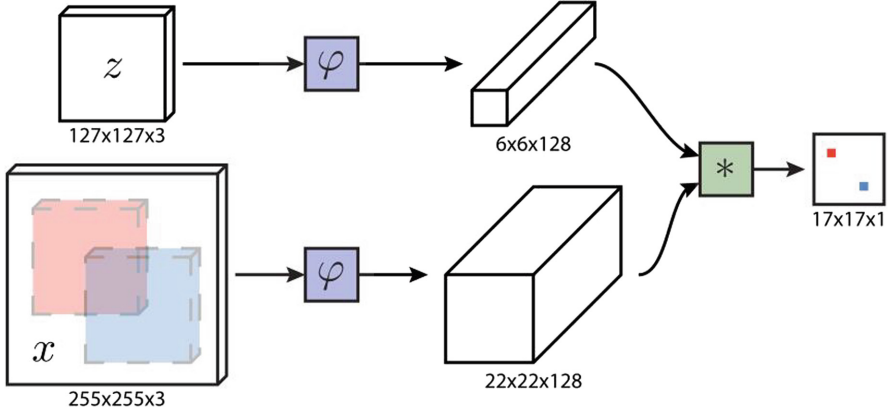


Fig. 4. SiamSNN architecture.

on the size of the search image. It uses similarity learning to solve the tracking problem. The model producing a higher score if the objects are similar, and the target position is predicted by the position of the maximum value in the score graph. During training, SiamFC preprocesses the training set, the images are resized to 255×255 and the center 127×127 area is the ground truth. In the same video, two images in a certain interval are selected to input the network for training. The positive and negative samples are calibrated by the distance from the center point in the response map. In the prediction, the template frame is processed in the same way, and the template frame branch is calculated only once. Then select three region proposals of different scales in the position of the previous frame to obtain three response maps, and select the maximum response value to get the final result.

We construct SiamSNN by the following steps:

- (1) Convert CNN part of SiamFC into Norm-CNN, and the modified structure is shown in Table 1. The type of all parameters are float16, and we use ReLU activation function after each conv layer.
- (2) Re-train Norm-CNN by using the original training method and data set in SiamFC, and perform weight normalization. Add IF neurons on the Norm-CNN to construct the SNN and transfer the trained parameters.
- (3) Add AAR encoder after input layer. We use the same convolution operation to evaluate the similarity of two spiking features.

The input and output sizes of the SiamFC and SiamSNN models have no difference, so we use the same strategy in preprocessing, scale selection, and response map processing. In order to guarantee the calculation speed, we continue to use the convolution operation to measure the similarity between the pulse characteristics. But it is not accurate enough, which is one of the main reasons for the decline in accuracy. We will next study the similarity evaluation method for spiking features to replace the convolution operation (Fig. 5).

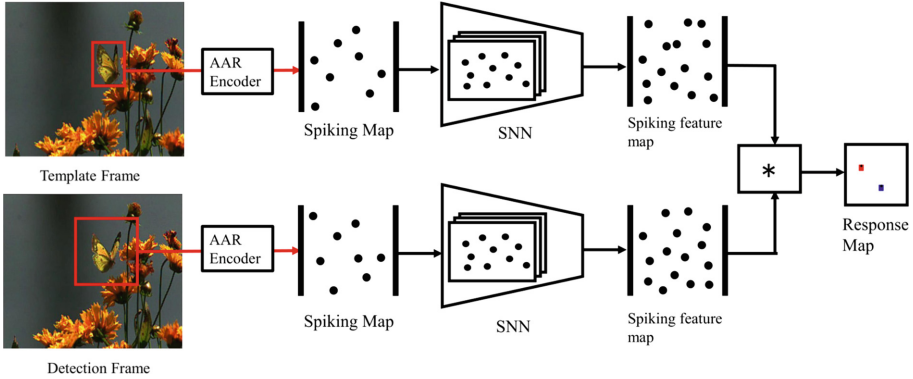


Fig. 5. SiamSNN architecture.

Table 1. Architecture of convolutional embedding function in Norm-CNN. In-channels are the kernel channels matched to input, Out-channels are the numbers of kernels.

Layer	Kernel-size	In-channels	Out-channels	Strides	Detection-size	Template-size
Input	—	3	—	—	127×127	255×255
Conv1	11×11	3	96	2	59×59	123×123
Avg-pooling1	3×3	—	—	2	29×29	61×61
Conv2	5×5	96	256	1	25×25	57×57
Avg-pooling2	3×3	—	—	2	12×12	28×28
Conv3	3×3	256	384	1	10×10	26×26
Conv4	3×3	384	384	1	8×8	24×24
Conv5	3×3	384	256	1	6×6	22×22

Our SiamSNN architecture theoretically takes advantage of high computational efficiency, low energy consumption, low resource consumption, easy hardware implementation in SNN, which makes it possible to apply the model to edge devices.

5 Experimental Results

5.1 Converted SNN on MNIST

We modify LeNet [22] to Norm-LeNet based on the method in Sect. 3.

In the process of modifying CNN to Norm-CNN, although the biases were removed and the Max-pooling layer was changed, the accuracy was only reduced by 0.003 after retraining in Table 2. In this adjustment process, the loss is not a lot. And the parameters are all the float16 type, which is significantly lower than the original model’s occupation of space resources.

Then we verify the effects of different total spiking time, normalization, and the maximum spiking rate. The voltage threshold was chosen to be 1 in the experiment. The first number in the model name indicates whether the SNN

Table 2. Comparative experiment on MNIST

Model	Accuracy	Space usage
LeNet	0.983	843 KB
Norm-LeNet	0.980	420 KB

model uses weight normalization, 1 indicates use, and 0 indicates no use. The second number represents the total spiking time. The maximum spiking rate S/T is set to 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9.

Table 3. Converted SNN on MNIST with different parameters

Model	The maximum spiking rate & accuracy						
	0.3	0.4	0.5	0.6	0.7	0.8	0.9
SNN-0-200	0.562	0.628	0.584	0.677	0.631	0.606	0.591
SNN-0-300	0.595	0.634	0.577	0.679	0.632	0.606	0.595
SNN-1-200	0.972	0.972	0.972	0.975	0.974	0.972	0.972
SNN-1-300	0.971	0.971	0.972	0.974	0.973	0.973	0.971

As shown in Table 3, when weight normalization is performed, the total spiking time is selected to be 200 ms and the maximum spiking rate is 0.6, which can achieve good conversion effect. Finally, we verify AAR encoder. It can be seen from the Table 4 that the AAR coding scheme can achieve better accuracy than the original rate coding scheme, and the accuracy decreases by only 0.007 on MNIST compared to the original method.

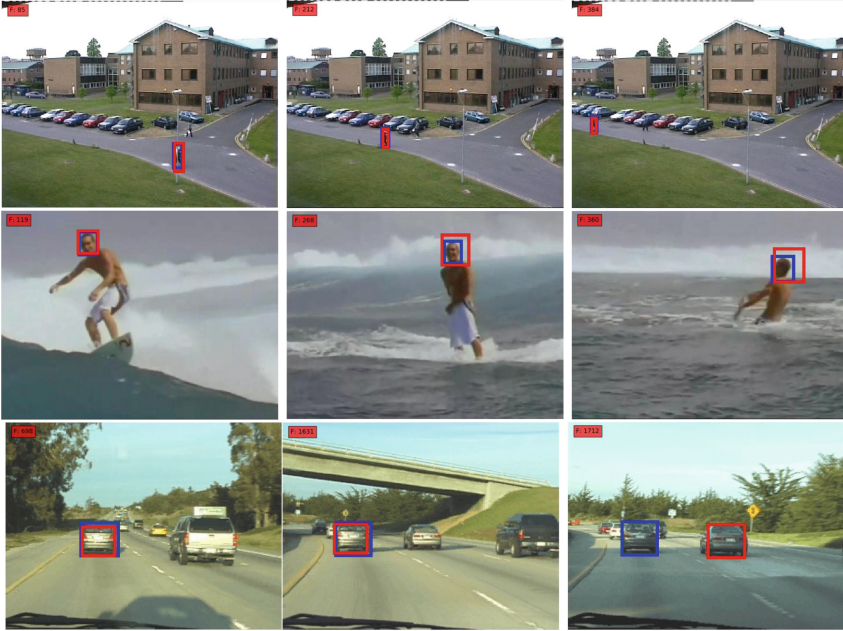
5.2 SiamSNN for Tracking

In SiamSNN, we set the total spiking time to be 200 ms, the maximum spiking rate is 0.6, use weight normalization, the constant parameter L in IF model is 0, and the voltage threshold is 1.

As shown in the Fig. 6, the red boxes are the results predicted by SiamSNN, the blue boxes are the ground truth. It can be seen from the image sequence of the first video that the SiamSNN model can achieve the desired tracking effect when the target has no background interference and there is no such problem as blur or severe deformation. In the second video, the model can basically track correctly, but the prediction result and the ground truth are too different. Starting from frame 268, the prediction result is larger than the ground truth. Although the tracking can be completed, it is not very accurate. In the third video, after 1713 frames, the tracking object was lost because of similar interference, and there was no retargeting in the subsequent sequences.

Table 4. Comparative experiment between AAR and original rate coding scheme

Model	Accuracy
LeNet	0.983
Rate coding SNN	0.975
AAR-SNN	0.976

**Fig. 6.** The first row is a video for tracking a walking man. The second row is a video for tracking a surfer’s head. The third row is a video for tracking a car. (Color figure online)

It is found that SiamSNN can successfully track the target in most of the previous frames. However, in the case where the disturbance is large and the similarity is too large, the effect of the tracking cannot be ensured. How to improve the robustness of the SNN model after conversion to improve the accuracy of tracking, is the place to be studied in the follow-up research.

6 Conclusion

In this paper, we propose a SNN architecture named SiamSNN for object tracking to avoid the training problems. Specifically, we propose a more comprehensive parameter conversion scheme with the processes of standardization, retraining, parameter transfer, and weight normalization, in order to convert a trained CNN

to a similar SNN. Then we propose an encoder named Attention with Average Rate Over Time (AAR) in order to encoding images to spiking sequences. We verify the effect of AAR encoder by experiments, and the converted SNN can reduce the resource consumption and reduce the complexity of the model in the case of classification or tracking.

Meanwhile, there is still a defect that the accuracy is reduced in the converted SNN model. This is mainly caused by the modification of pooling layer and the removal of the original BN layer. And the current similarity matching algorithm of spiking features is not accurate enough. In addition, the robustness of SNN is not as good as CNN. Later, we will work on these places to make SNN achieve better results in the applications.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 1097–1105. MIT Press, Lake Tahoe (2012)
2. Zhuang, B., Wang, L., Lu, H.: Visual tracking via shallow and deep collaborative model. *Neurocomputing* **218**(61), 71 (2016)
3. Tavanaei, A., Ghodrati, M., Kheradpisheh, S.R., et al.: Deep learning in spiking neural networks. *Neural Netw.* **111**(47), 63 (2019)
4. Maass, W.: Lower bounds for the computational power of networks of spiking neurons. *Neural Comput.* **8**(1), 1–40 (1996)
5. Neil, D., Pfeiffer, M., Liu, S.: Learning to be efficient: algorithms for training low-latency, low-compute deep spiking neural networks. In: *ACM*, pp. 293–298 (2016)
6. Xu, Q., Qi, Y., Yu, H., et al.: CSNN: an augmented spiking based framework with perceptron-inception. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1646–1652. Morgan Kaufmann, Stockholm (2018)
7. Cao, Y., Chen, Y., Khosla, D.: Spiking deep convolutional neural networks for energy-efficient object recognition. *Int. J. Comput. Vis. (IJCV)* **113**(1), 54–66 (2015)
8. Diehl, P.U., Neil, D., Binas, J., et al.: Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In: *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, Killarney (2015)
9. Rueckauer, B., Lungu, I.A., Hu, Y., et al.: Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* **11**, 682 (2017)
10. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report, University of Toronto (2009)
11. Deng, J., Dong, W., Socher, R., et al.: Imagenet: a large-scale hierarchical image database. In: *Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255. IEEE, Miami (2009)
12. Hu, Y., Tang, H., Wang, Y., et al.: Spiking deep residual network. arXiv preprint [arXiv:1805.01352](https://arxiv.org/abs/1805.01352) (2018)
13. Sengupta, A., Ye, Y., Wang, R., et al.: Going deeper in spiking neural networks: VGG and residual architectures. *Front. Neurosci.* **13**, 95 (2019)
14. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: *Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. IEEE, Las Vegas (2016)

15. Danelljan, M., Hager, G., Shahbaz, K.F., et al.: Convolutional features for correlation filter based visual tracking. In: Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV), pp. 58–66. IEEE, Santiago (2015)
16. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: Fully-convolutional siamese networks for object tracking. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9914, pp. 850–865. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48881-3_56
17. Li, B., Yan, J., Wu, W., et al.: High performance visual tracking with siamese region proposal network. In: Computer Vision and Pattern Recognition (CVPR), pp. 8971–8980. IEEE, Salt Lake City (2018)
18. Ma, J., Shao, W., Ye, H., et al.: Arbitrary-oriented scene text detection via rotation proposals. *IEEE Trans. Multimed.* **20**(11), 3111–3122 (2018)
19. Zhu, Z., Wang, Q., Li, B., Wu, W., Yan, J., Hu, W.: Distractor-aware siamese networks for visual object tracking. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11213, pp. 103–119. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01240-3_7
20. Farabet, C.: Towards real-time image understanding with convolutional networks. Master Degree thesis. University Paris-Est, Paris (2013)
21. Krause, T.U., Wrtz, P.D.D.R.: Rate coding and temporal coding in a neural network. Master Degree thesis. University of Bochum, Germany (2014)
22. LeCun, Y., Bottou, L., Bengio, Y., et al.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2324 (1998)