Convolutional Spiking Neural Networks for Spatio-Temporal Feature Extraction

Ali Samadzadeh · Fatemeh Sadat Tabatabaei Far · Ali Javadi · Ahmad Nickabadi · Morteza Haghir Chehreghani

Accepted: 8 March 2023 / Published online: 4 May 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
Spiking neural networks (SNNs) can be used in low-power and embedded systems e.g., neuromorphic chips due to their event-based nature. They preserve conventional artificial neural networks (ANNs) properties with lower computation and memory costs. The temporal coding in layers of convolutional SNNs has not yet been studied. In this paper, we exploit the spatio-temporal feature extraction property of convolutional SNNs. Based on our analysis, we have shown that the shallow convolutional SNN outperforms spatio-temporal feature extractor methods such as C3D, ConvLSTM, and cascaded Conv and LSTM. Furthermore, we present a new deep spiking architecture to tackle real-world classification and activity recognition tasks. This model is trained with our proposed hybrid training method. The proposed architecture achieved superior performance compared to other SNN methods on NMNIST (99.6%), DVS-CIFAR10 (69.2%), and DVS-Gesture (96.7%). Also, it achieves comparable results compared to ANN methods on UCF-101 (42.1%) and HMDB-51 (21.5%) datasets.

Keywords SNN · CNN · Feature extractor · Spatio-temporal

Ahmad Nickabadi
nickabadi@aut.ac.ir
Ali Samadzadeh
a_samad@aut.ac.ir
Fatemeh Sadat Tabatabaei Far	tabatabaeifateme@aut.ac.ir
Ali Javadi
fleissig@aut.ac.ir
Morteza Haghir Chehreghani
torteza.chehreghani@chalmers.se

1 Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran
2 Department of Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden
1 Introduction

Since the commercial release of event cameras in 2008, applications for neural networks with low computation costs have emerged e.g. Spiking Neural Networks (SNN). Interestingly, SNNs can be utilized in real-time applications and harsh environments (i.e. noisy pixel values and adversarial attacks [1]). Real-time applications of SNNs consist of (1) visual simultaneous localization and mapping or visual odometry (also known as VSLAM or VO) [2–4], (2) pose tracking applications [5, 6] and etc. Furthermore, SNNs are useful in high-speed applications such as object recognition in self-driving cars [7]. Moreover, if event-camera’s price drops, applications of SNNs can be extended to other vision domains. Also, there is new research around the usage of high-speed spiking data to achieve impressive results such as high-speed object tracking via transformer SNNs [8], and high-speed video reconstruction and reducing blur in videos [9].

SNN encodes data in sequences of spike signals. Consequently, it can execute more complex cognitive tasks similar to the brain cortex processing pattern [10–12]. When a neuron’s membrane potential reaches a threshold, the neuron will trigger and transmit a spiking signal. To be precise, spikes are binary codes that decay in time similar to an electrical capacitor’s charge. This binary nature of SNNs makes them efficient in terms of memory consumption and computation cost which leads to lower power consumption [7].

The training of SNNs is a challenging task and an active area of research. There are some great attempts in the literature, namely Spiking Timing Dependent Plasticity (STDP), ANN-SNN conversion, and backpropagation through time (BPTT) [13–15]. STDP-based methods [16–19] may have some advantages like the ability to train unsupervised but the number of layers inside them is limited. Conversely, ANN-SNN conversion methods [20–26] have achieved outstanding results with deep architectures (more than 18 layers). However, these methods assume that SNNs are a sampling of ANNs in time [25]. This assumption ignores the temporal properties of SNNs. There are some remarkable temporal properties to the SNNs that cannot be achieved with ANN-SNN conversion-based training [27–29]. BPTT-based methods solve this problem by using training over time. As [30] explained, the main problem of direct SNN training is the non-differentiability of the spiking function. [14] overcame this problem by approximating the derivative of the threshold function. This approximation will be the source of gradient vanishing as demonstrated in Fig. 1. These vanishing gradients will cause some issues in training deep architectures (more than 16 layers). [15] tries to recreate batch normalization for SNNs to use its properties and build deeper networks. Nevertheless, the NeuNorm solution [15] does not have the same properties as batch-norm and is not much of a help in training deeper networks.

Time Inheritance Training (TIT) is introduced in [32]. This method reduces the training time by initializing the SNN with a smaller simulation length. With the help of TIT, the performance of SNNs has improved on both static and neuromorphic datasets. However, This method does not help with the gradient vanishing of the deep neural networks. Moreover, [33] addresses the problem of undesired Membrane Potential Distribution (MPD) shifts in direct training of SNNs in static datasets. Based on their analysis, a novel distribution loss (MPD-Loss) that explicitly penalizes those undesired shifts can result in rectified MPD and more stable training. This work is valuable for training SNNs on spatial-only feature extraction and does not include temporal training into account. Some recent works explore more energy-efficient training procedures for SNNs [34].

To solve the aforementioned challenges, we propose a solution by pre-initializing the weights with initial non-spiking training and using skip connections inspired by the ResNet
architecture (more explanation in Sect. 3.2). Furthermore, to demonstrate the spatio-temporal power and efficiency of SNNs (as [27] mentioned), we design a series of test cases (synthetic dataset). Also, This synthetic dataset helps to explore the spatio-temporal property of SNNs compared to ANNs (details in Sect. 2.2). Then, we discuss the spatio-temporal properties of SNNs that distinguishes them from conventional recurrent ANNs (See Fig. 2). Finally, we propose a deep architecture and test it on NMNIST [35], IBM DVS-Gesture [36], DVS-CIFAR10 [37], UCF-101 [38], HMDB-51 [39], and N-imagenet [40] datasets.

To summarize, our main contributions are:

- Providing insight into a new capability of convolutional SNNs,
- Introducing a stable training method for spatio-temporal deep SNNs,
- Introducing a custom-designed deep Spatio-temporal Spiking ResNet to overcome complex tasks.

2 Spatio-Temporal Property of SNNs

The Leaky Integrate and Fire (LIF) model [41] is one of the simple and successful modelings of SNN. We have used this model in this work. A brief overview of LIF is provided in Sect. 2.1. Also, a detailed explanation of the proposed synthetic dataset is provided in Sect. 2.2.

2.1 Leaky Integrate and Fire (LIF)

The LIF model neuron implementation is defined as follows:

\[ U^{(t,n)} = \alpha(J - O^{(t-1,n)})U^{(t-1,n)} + g(O^{(t,n-1)}) \]  
\[ O^{(t,n)} = f(u^{(t,n)}) \]  

Fig. 1 Top row: the gradient vanishing problem in using the approximate method for the derivating of the step activation function. Bottom row: the advantage of high gradient flow by using the Leaky-ReLU activation function.
Fig. 2 Comparison between Grad-Cam [31] analysis between using two-layer CNN and two-layer spiking CNN. The outputs are based on the correct classification of number and rotation angle. The input is Seq. 2 of the synthetic dataset described in Sect. 2.2.

In Eq. (1), $J$ is a matrix of ones and $g(.)$ is the layer-wise operation. For linear layers, $g(.)$ will be defined as:

$$g(O^{(t, n-1)}) = W_1^{n} O^{T(t, n-1)}$$

(3)

The term $(J - O^{(t-1, n)})$ in Eq. (1) represents the neuron rest and enforces sparsity in the LIF neurons. $f(.)$ is the activation function which can be interpreted as a threshold function. This function for each node is defined as:

$$f(U_i^{(t, n)}) = \begin{cases} 1 & \text{if } U_i^{(t, n)} \geq T \\ 0 & \text{if } U_i^{(t, n)} < T \end{cases}$$

(4)

In the above equations, $n$ is the layer number, and $t$ is the time-stamp. In Eq. (1) $\alpha$ is the decay factor. Note that the decay factor needs careful tuning and all of the equations (except the activation function) are in the matrix form.

The following operation needs to be performed on the output of the last layer to obtain the output of the SNN (assuming rate encoding over an arbitrary time window):

$$O^{(t, N)} = \frac{1}{T} \sum_{k=t-T}^{t} O^{(k, N)}$$

(5)

Fig. 3 summarizes the description of LIF model. In the next section, this synthetic dataset is explained in detail.

### 2.2 Synthetic Dataset

A synthetic dataset is designed to demonstrate the spatio-temporal feature extraction property of SNNs (see Fig. 4). This dataset is important for the experiments since it is a simple benchmark to evaluate the ability of spatio-temporal feature extraction in very shallow (2 layer) neural networks. The results in Sect. 4.1 and Fig. 7 show that no shallow neural network other than SCNN is able to reach higher than 92% accuracy in total. This synthetic dataset
is designed using MNIST numbers and consists of 5 cases, each for testing one aspect of spatio-temporal feature extraction.

In detail, seq. 1 is obtained by doing either zoom-in or zoom-out in a time window of 10 (All of the frames have the same zoom percentage compared to the previous frame). Each image in the time window is 10% larger (or smaller based on zoom-in or zoom-out class) than the previous one (see Fig. 4). In this sequence, the network should classify the number from the ten characters of the MNIST dataset and the act (zoom-in or zoom-out); therefore, there are 20 classes to be specified.

Seq. 2 is a regular rotation from 0° to 360° within the time window of ten. Each frame is obtained by rotating the previous frame 36°. In Fig. 4, a 7 character from MNIST is rotated both clockwise and counterclockwise rows 3 and 4, respectively.

Seq. 3 is very similar to sequence 1 with the difference in zoom percentage. In this sequence, the zoom-in (and zoom-out) is performed from 0 to 50%. Therefore, this sequence is less challenging than sequence 1.

Seq. 4 is designed to evaluate the noise and occlusion robustness of models. In this sequence, the characters of MNIST are masked by a square with dimensions of 14x14 pixels (half the dimensions of the original character). This mask is moved randomly over the characters inside the time window of 10. Using this masking, it is almost impossible for the network to determine the class of characters and reach high accuracy without looking into other frames. An example of character 4 from this sequence is depicted in Fig. 4.

Lastly, seq. 5 is the most challenging sequence in terms of the human eye (see Fig. 4). In this sequence, each character is rotated in the same direction (counterclockwise or clockwise) with a random degree between 0° and 360°. This random selection is based on the uniform distribution (therefore, the mean rotation would be 180°). The rotations are also performed in a time window of 10. Similar to sequence 2, this sequence also consists of 20 classes (counterclockwise or clockwise rotated frames of MNIST characters).
In summary, this synthetic dataset can verify most aspects of the spatio-temporal feature extraction in a simple way.

3 Methodology

In this section, we propose a new method to train deep SNN architectures. Also, we have proposed a new architecture to prove the advantage of our training method, which is shown in Fig. 5. This architecture is inspired by ResNet architecture and the work of [25].
Fig. 6  Training accuracy per epoch. This plot shows the robustness of performance using our learning method over multiple tries on DVS-CIFAR10. We trained the model three times randomly, using each learning method. In our training method, the first five epochs were pre-training and the rest is almost equal to STBP.

3.1 Training Method

As explained in Sect. 1, ANN-SNN conversion methods achieve very high accuracy over very complex datasets like Imagenet. However, they lack the temporal property of SNNs and don’t have much noise robustness as BPTT methods. The proposed training method is a hybrid of ANN base backpropagation and BPTT proposed in [14]. During the first \( n \) epochs, the SNN model is trained with normal backpropagation with shifted leaky-ReLU activations for each layer (except the output layer). Then, the SNN model is trained with BPTT. Moreover, the problem of gradient vanishing is resolved within the first steps and the convolutional layers will have good initial weights. Having this weight initialization helps to reach optimal results easier. Also, there is no resting mechanism embedded. Hence, basically the Eqs. 1 and 4 change into the Eqs. 6 and 7. In the shifted leaky-ReLU function demonstrated in Eq. 7, the \( \beta \) hyper-parameter is a small constant near zero.

\[
U^{(t,n)} = \alpha U^{(t-1,n)} + g(O^{(t,n-1)})
\]

\[
f(U_i^{(t,n)}) = \begin{cases} 
U_i^{(t,n)} & \text{if } U_i^{(t,n)} \geq T \\
-\beta U_i^{(t,n)} & \text{if } U_i^{(t,n)} < T
\end{cases}
\]

After \( n \) epochs, the activation function is switched into the step function and LIF neuron model is used. The outputs of each layer are binary and the training procedure after \( n \) epochs is similar to BPTT proposed in [14, 15]. The proposed initial step will make the training much more robust (as depicted in Fig. 6) and will enable the deep SNNs to reach much higher accuracy; therefore, almost resolving the problem of gradient vanishing as demonstrated in Fig. 1. In the following section, a deep SNN architecture is proposed to showcase this training method’s efficiency and state-of-the-art performance.
3.2 The STS-ResNet Architecture

As mentioned in the previous sections, the principal difficulty of training deep SNNs is gradient vanishing. ResNet architectures, solve the obstacle of gradient vanishing by utilizing the skip connections. The skip connections increase performance at a drastic rate as well. Inspired by [25], an architecture based on ResNet18 is proposed (see Fig. 5). The main difference between our proposed network and the one in [25] is synapse places, dropouts, additional skip connections, and the fact that it can be trained directly. Hence, this architecture is called spatio-temporal Spiking ResNet (STS-ResNet). Details of each block are shown in Fig. 5. Similar to ResNet18 architecture, there are two sub-blocks in each block. In order to force binary outputs after each layer, the thresholding activation function, namely synapse, is applied to the output of each layer. Inside subblocks, there is a dropout layer to ensure generality and force sparsity. Dropouts are almost playing the role of batch normalization for generalization. Average pooling layers do not conflict with the nature of binary outputs in each layer. They are part of the next layer operation; therefore, for example, the last average pooling layer is part of the FC layer defined as a new layer (a fact never mentioned before in previous works). Another major difference between this architecture and ResNet18 is the block skip connections. They are added from blocks three and four to the input of the average pooling layer. In order to increase performance, instead of conventional summation operation, concatenation operation is used afterward. This concatenation does not happen in ResNet skip architecture.

The STS-ResNet architecture consists of 18 layers including 16 layers in the blocks, a convolutional layer, and a fully connected layer at the end of the architecture. To allow all kinds of inputs (binary, grayscale, color image), a binarizing synapse is applied at the input. It is the first time an 18-layer SNN is trained in the spatio-temporal domain to classify spatio-temporal tasks.

4 Experiments

4.1 Spatio-Temporal Feature Extraction Experiments

In this section, the performance of top-quality ANN spatio-temporal feature extractors (namely CNN, C3D, ConvLSTM, conv+LSTM) are evaluated and compared with SNNs. Convolutional Neural Networks (CNN) are known for their usage in computer vision tasks [42]. Moreover, CNN+LSTM [43] and C3D network [44] are appropriate for modeling spatio-temporal information. ConvLSTM proposed in [45] is also suitable for spatio-temporal feature learning. As mentioned in [46], there are some spatio-temporal datasets such as moving MNIST and CIFAR10-DVS to evaluate these methods. However, these datasets are not simple or intuitive enough to show every aspect of spatio-temporal feature extraction. Therefore, the synthetic dataset (explained in Sect. 2.2) is designed to demonstrate the critical factors of these architectures. For this matter, a shallow network (< 5 layers) of each model is used for comparison. Furthermore, ConvSNN can achieve the same or better results with fewer neurons than its competitors. Having fewer parameters results in less computational power (as demonstrated in Fig. 7). In the following paragraph, we will explain how we feed the synthetic dataset into the Neural Networks (NNs).

The synthetic dataset is fed into CNN via frame concatenation, and CNN gets a stack of images as input. This type of dataset feeding holds for the C3D too, as we concatenate
| Method          | Config        | MNIST (%) | Seq1 (%) | Seq2 (%) | Seq3 (%) | Seq4 (%) | Seq5 (%) | Total (%) |
|-----------------|---------------|-----------|----------|----------|----------|----------|----------|-----------|
| CNN             | 256-256       | 99        | 98.2     | 98.8     | 98.27    | 20.1     | 82.8     |
| CNN+LSTM       | 256-256 + 256(LSTM) | 96.84   | 94.88    | 91.2     | 73.23    | 91.5     | 64.0     | 92.7      |
| ConvLSTM       | 256-256       | 99        | 99.11    | 98.96    | 97.43    | 99.17    | 98.8     | 99.36     |
| C3D            | 48-48         | 99.4      | 99.49    | 99.32    | 99.73    | 99.43    | 99.1     | 99.43     |
| STS-ResNet     | 18 layers (see Sec 3.2) | 99.7   | 99.26    | 99.43    | 99.1     | 99.43    | 97.9     | 97.9      |
the frames in 3rd dimension and give it to the C3D network. Time-domain is considered for feeding data into other architectures (ConvLSTM, Conv+LSTM, and ConvSNN) and frames are fed into the networks in each time step and then backpropagation through time is performed at the end. The output for these architectures is determined by a voting mechanism which indicates that a neuron with the most amount of firing in the time window will win (also known as the winner-takes-all or fire-rate-based output). In the following paragraph, the essence of memory for NNs is explained.

Due to the structure of the synthetic dataset, this frame-by-frame feeding through time requires the networks to have some sort of memory. To explain this memory need, take the example of seq2. If the NN does not have memory, it cannot recognize clockwise and counterclockwise movement. Since it has received the same frames with different orders at the end of the time window. Fire-rate-based output mechanism will not make an acceptable prediction for the two mentioned frame sequences. In the following, we will discuss training methods and evaluation criteria.

The training method used for ConvSNN is the simple spatio-temporal backpropagation presented in [14]. A training algorithm for other methods was conventional Backpropagation and BPTT. The training is performed for at least 100 epochs and the maximum performance on the test set is recorded for each architecture. For evaluation, each NN is tested 10 times and the average top-1 accuracy is reported in Table 1.

As discussed in Sect. 2.2, the general property of spatio-temporal feature extraction is examined with seq2 and seq3. All architectures show promising results on those two sequences (Table 1). The long-term preservation of data in memory is investigated using seq1 as explained in Sect. 2.2. The typical LSTM layer (lacking cut connections) does not have this property and as a result, the accuracy will drop as shown in Table 1. Moreover, the robustness of architectures to noise and their ability to extract spatial features through time is examined with seq4. The results prove SNN’s superiority in this aspect over all other networks. The performance of CNN+LSTM is a little lower than others. The output of CNN is given to the LSTM layer and it does not contain all properties of spatial features which makes it strenuous for the LSTM layer to determine the occluded character. Finally, seq5 evaluates the random (in time) spatio-temporal feature extraction. This sequence evaluates the non-repeatable temporal feature extraction ability of the mentioned NNs. Results in Table 1 show that only CNN+LSTM can outperform SNN in this sequence with little margin, other NNs (especially the CNN architecture) drastically fail to achieve comparable accuracy. However, the conv+LSTM network has too many layers compared to SNN. The memory complexity is shown in Fig. 7.
To further evaluate the amazing achievement of ConvSNN, the confusion matrices are illustrated for seq1 and seq15 in Figs. 9 and 10. Figure 9 illustrates the imperfection of convSNNs in extracting random temporal properties (the problem of distinguishing between clockwise and counterclockwise rotation in seq5). Figure 10 shows long-term memory preservation issues in conventional LSTM layers.

Moreover, SNN architecture tries to extract both spatial and temporal properties inside the feature map, while ANN only extracts the spatial property of MNIST characters. An experiment is conducted to show the differences in the areas that are important to SCNN compared to CNN (see Fig. 2). In this figure, the Grad-CAM [31] output of the last convolutional layer of spiking and non-spiking NN is shown. The Grad-CAM output is proven to be a good measure for determining what deep CNNs are looking into inside the input data. As can be seen in Fig. 2, the spiking convolutional layers have very sparse attention regions, while non-spiking CNN is giving attention to the character shape.

The mentioned results prove the claim that SNNs are suitable for spatio-temporal feature extraction especially when features aren’t in a regular time or space pattern or they are noisy. In order to make the SNNs more suitable for complex conditions, we proposed STS-ResNet architecture as mentioned in Sect. 3.2. The results of this architecture are reported in Table 1 with other methods. The performance of this architecture over more challenging datasets is explained in Sect. 4.3.

4.2 Experiments on Memory and Compute Complexity of SNN

In this section, we experiment on memory consumption and time (compute) complexity of SCNN compared to other aforementioned NNs. The goal of this experiment is to prove the claim of low compute and memory usage of SNNs compared to ANNs. For this experiment, the explained synthetic dataset is used. ConvSNN and STS-ResNet are compared to CNN, CNN+LSTM, ConvLSTM, and C3D with the same configuration as the previous experiment in Sect. 4.1. The results are depicted in Fig. 7 and Table 2. The results in Fig. 7 show the great performance of the SNN-based architectures with low memory (num of parameters) consumption compared to competitors. The reason behind the low memory consumption of
SNN-based NN is that SNNs can use binary and low kernel size to their advantage. Also, Table 2 shows that even 18-layer STS-ResNet is far less memory and compute-intensive than two-layer ConvLSTM and C3D models. However, the time complexity (depicted as MFLOPS in Table 2) per number of parameters of spiking-based NNs is much higher than the other networks. This high time to memory complexity ratio is because of the timestep window and several passes of calculation for the Spiking models. The ConvLSTM model is similar to SNN-based networks regarding this property.

4.3 Experiments with Deep Residual SNN

First, we have conducted an experiment on seq2 of the synthetic dataset to investigate the impact of introducing ResNet blocks to SNNs. The results are depicted in Fig. 8. In this figure, the mean of all gradient norms of all layers is shown for all epochs. Figure 8 shows the gradient vanishing and becoming zero for the Spiking CNNs with more than seven layers. While the ResNet-based SNN gradients remain enough for training and achieving high accuracy.

Second, to showcase the advantages of our learning method and designed architecture, over the prior SNNs and ANNs, we chose NMNIST and DVS-Gesture, and N-imagenet [40] datasets. We compare STS-ResNet with other state-of-the-art SNN methods. Table 3 depicts the performance comparison. Furthermore, we chose CIFAR10-DVS dataset to depict the performance of this architecture in complex neuromorphic tasks compared to the state-of-the-art SNN methods. Finally, we have evaluated the ResNet18 (traditional ANN) and STS-ResNet architecture on complex activity recognition of UCF-101 and HMDB-51 datasets to compare the STS-ResNet architecture with its ANN counterpart.

The output of the networks is calculated based on the fire-rate-based method and the inputs are given to networks in raw form (neuromorphic datasets are fed into NNs with event stacking through time). The evaluation of the networks is based on top-1 accuracy calculated over 10 runs of training for at least 100 epochs.
Table 3 elucidates the significant improvement of STS-ResNet over the previous outstanding methods. The only downside of STS-ResNet is being inferior to ResNet18 over the UCF-101 dataset. The reason for this mediocre performance is the extreme importance of spatial features in UCF-101. It is evident from this observation that STS-ResNet, despite being trained by our method, is still inferior to the ANN counterpart. On the other hand, the results on HMDB-51 are promising, showing great performance over this challenging dataset. The potential for overfitting is high in this dataset. The HMDB-51 dataset has more temporal features embedded than the UCF-101 dataset. The temporal features make the dataset more suitable for SNNs since they have neuron memories to accommodate for temporal feature extraction. This is the first time that an SNN architecture is evaluated on complex datasets like UCF-10 and HMDB-51. Also, in N-imagenet dataset [40] the STS-ResNet is inferior to the DiST model with a small margin (developed by the same authors of the N-imagenet dataset). The authors of DiST used a ResNet30 as a backbone structure and hence our 18-layer STS-ResNet is superior to it because it has almost half the memory consumption and comparable results.

Furthermore, our architecture needs much fewer kernels than [15] and is much more memory efficient. To be precise, our model concentrates on making the network deep, while [15] tries to increase the number of kernels per layer. In order to see the exact parameters, refer to the implementation details in the next section.
4.4 Implementation Details

In this subsection, we provide the experiment details on the networks’ architectures and parameters. Basic implementation details for the test cases are as follows: Frame window size for all networks is set to 10. The learning rate for all network architectures is set to $1e^{-3}$ (except ConvSNN which is set to $5e^{-4}$). All architectures are trained enough to reach maximum accuracy (more than 100 epochs). These experiments are performed at least 10 times and the mean value for them is reported in the tables. For training networks, Adam optimizer and least mean square error are used (except for the ConvLSTM network where binary cross-entropy is used). Batch sizes are 100 except for ConvSNN which is 20.

As for the ConvSNN specific parameters, the threshold value is set to 0.5. This value is extremely important and a slight change in it will result in better or worse results. $\alpha$ or decay factor is set to 0.5. Increasing this value will result in better preservation of memory and vulnerability to noise. Also, the resting mechanism is disabled. The resting mechanism
maintains more sparsity in spike patterns but decreases the accuracy. Derivative of Dirac function is approximated with Gaussian function $\mathcal{N}(\text{Threshold}, \frac{1}{\sigma})$ (Rect function is a better approximation in terms of performance but it will learn harder and the mean accuracy for several runs will drop dramatically). The initial pre-training is performed for almost 50 epochs and then the BPTT presented in [14] is used. The comparison of our learning method and the previous BPTT-based method is depicted in Fig. 6.

Parameters of the proposed STS-ResNet architecture to tackle CIFAR10-DVS are as follows: frame window length and the learning rate is set to 10 and 5e-4, respectively and training is performed for more than 50 epochs and more than five runs as before. The optimizer is SGD with a momentum of 0.9. The loss function is binary cross entropy. The batch size is 10 and 1000 events are concatenated per frame. Other SNN-specific parameters (threshold, resting mechanism, derivative approximate function) are as before, except for the decay factor which is set to 0.8 (in the CIFAR10-DVS dataset memory is more important than noise robustness).

All of the experiments are performed on a system with Intel Core i5-6500, NVIDIA GTX 1080, and NVIDIA GTX 1080ti with 32 GB RAM and SSD storage (Figs. 9, 10).

5 Conclusion

This paper demonstrated the potential of SNNs in terms of spatio-temporal feature extraction. Particularly, their capacity to extract randomly distributed features in the time and space domains. This claim was supported by experiments with a special type of synthetic dataset designed for the matter. Furthermore, a modified training method is proposed to enable the training of deeper networks. Next, a new deep SNN architecture was proposed to showcase its performance gain in multiple known datasets. The introduced SNN architecture was evaluated on challenging datasets including CIFAR10-DVS, NMNIST, and DVS-Gesture to depict its performance gain compared to previous architectures.

Regarding the results, a shallow SNN outperformed shallow ANNs over extreme conditions (synthetic dataset) and surpassed SNNs over the typical event-based dataset (CIFAR10-DVS). Moreover, SNNs have much lower memory consumption (with the assumption of binary connections) and computation cost which results in less overall hardware power consumption. Also, in some situations, SNNs with few numbers of neurons can achieve what oversized ANNs can barely achieve.

The remaining problem to be solved is the adaptation of batch normalization properties to SNNs. These properties are required to have very deep SNNs (e.g. 101 layers). Also, there should be a better solution for backpropagation other than approximating the derivative of the activation function. The approximate functions are the primary cause of gradient vanishing. Another step in the journey of analyzing these networks might be an analysis of other types of SNNs such as GANs.

To sum it all up, this work renders the advantages of SNNs transparent and proposes a new deep architecture with some solutions to train other deeper SNNs.

Funding Not available.

Availability of Data and Materials All of the data and materials are available under https://github.com/aa-samad/conv_snn.

Code Availability Code is available under https://github.com/aa-samad/conv_snn.
Declarations

Conflict of Interest Not available.

Consent to Participate Not applicable.

Consent for Publication Not applicable.

Ethics Approval Not applicable.

References

1. Sharmin S et al (2019) A comprehensive analysis on adversarial robustness of spiking neural networks. IEEE, pp 1–8
2. Kim H, Leutenegger S, Davison AJ (2016) Real-time 3d reconstruction and 6-dof tracking with an event camera. Springer, pp 349–364
3. Kueng B, Mueggler E, Gallego G, Scaramuzza D (2016) Low-latency visual odometry using event-based feature tracks. IEEE, pp 16–23
4. Rebecq H, Horstscherfer T, Gallego G, Scaramuzza D (2016) Evo: a geometric approach to event-based 6-dof parallel tracking and mapping in real time. IEEE Robot Autom Lett 2(2):593–600
5. Gallego G et al (2017) Event-based, 6-dof camera tracking from photometric depth maps. IEEE Trans Pattern Anal Mach Intell 40(10):2402–2412
6. Mueggler E, Huber B, Scaramuzza D (2014) Event-based, 6-dof pose tracking for high-speed maneuvers. IEEE, pp 2761–2768
7. Wang W et al (2020) Temporal pulses driven spiking neural network for fast object recognition in autonomous driving. arXiv preprint arXiv:2001.09220
8. Zhang J et al (2022) Spiking transformers for event-based single object tracking, pp 8801–8810
9. Zhu L et al (2022) Event-based video reconstruction via potential-assisted spiking neural network, pp 3594–3604
10. Allen JN, Abdel-Aty-Zohdy HS, Ewing RL (2009) Cognitive processing using spiking neural networks. IEEE, pp 56–64
11. Kasabov N, Capecci E (2015) Spiking neural network methodology for modelling, classification and understanding of eeg spatio-temporal data measuring cognitive processes. Inf Sci 294:565–575
12. Zhang X, Xu Z, Henriquez C, Ferrari S (2013) Spike-based indirect training of a spiking neural network-controlled virtual insect. IEEE, pp 6798–6805
13. Mostafa H (2017) Supervised learning based on temporal coding in spiking neural networks. IEEE Trans Neural Netw Learn Syst 29(7):3227–3235
14. Wu Y, Deng L, Li G, Zhu J, Shi L (2018) Spatio-temporal backpropagation for training high-performance spiking neural networks. Front Neurosci 12:331
15. Wu Y et al (2019) Direct training for spiking neural networks: faster, larger, better, vol 33, pp 1311–1318
16. Hao Y, Huang X, Dong M, Xu B (2020) A biologically plausible supervised learning method for spiking neural networks using the symmetric stdp rule. Neural Netw 121:387–395
17. Kheradpisheh SR, Ganjtabesh M, Thorpe SJ, Masquelier T (2018) Stdp-based spiking deep convolutional neural networks for object recognition. Neural Netw 99:56–67
18. Lee C, Panda P, Srinivasan G, Roy K (2018) Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning. Front Neurosci 12:435
19. Tavanaei A, Maida A (2019) Bp-stdp: approximating backpropagation using spike timing dependent plasticity. Neurocomputing 330:39–47
20. Cao Y, Chen Y, Khorasani D (2015) Spiking deep convolutional neural networks for energy-efficient object recognition. Int J Comput Vis 113(1):54–66
21. Diehl PU et al (2015) Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. IEEE, pp 1–8
22. Esser SK, Appuswamy R, Merolla P, Arthur JV, Modha DS (2015) Backpropagation for energy-efficient neuromorphic computing, pp 1117–1125
23. Han B, Srinivasan G, Roy K (2020) Rmp-snn: residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network, pp 13558–13567
24. Rueckauer B, Lungu I-A, Hu Y, Pfeiffer M, Liu S-C (2017) Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. Front Neurosci 11:682
25. Sengupta A, Ye Y, Wang R, Liu C, Roy K (2019) Going deeper in spiking neural networks: Vgg and residual architectures. Front Neurosci 13:95
26. Stromatias E, Soto M, Serrano-Gotarredona T, Linares-Barranco B (2017) An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data. Front Neurosci 11:350
27. Deng L et al (2020) Rethinking the performance comparison between ssn and anns. Neural Netw 121:294–307
28. He W et al (2020) Comparing ssn and rnns on neuromorphic vision datasets: similarities and differences. arXiv preprint arXiv:2005.02140
29. Wu J, Chua Y, Zhang M, Li H, Tan KC (2018) A spiking neural network framework for robust sound classification. Front Neurosci 12:836
30. Neftci EO, Mostafa H, Zenke F (2019) Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks. IEEE Signal Process Mag 36(6):51–63
31. Selvaraju RR et al (2017) Grad-cam: visual explanations from deep networks via gradient-based localization, pp 618–626
32. Deng S, Li Y, Zhang S, Gu S (2022) Temporal efficient training of spiking neural network via gradient re-weighting. arXiv preprint arXiv:2202.11946
33. Guo Y et al (2022) Recdis-snn: rectifying membrane potential distribution for directly training spiking neural networks, pp 326–335
34. Na B et al (2022) Autosnn: towards energy-efficient spiking neural networks. PMLR, pp 16253–16269
35. Orchard G, Jayawant A, Cohen GK, Thakor N (2015) Converting static image datasets to spiking neuro-morphic datasets using saccades. Front Neurosci 9:437
36. Amir A et al (2017) A low power, fully event-based gesture recognition system, pp 7243–7252
37. Li H, Liu H, Ji X, Li G, Shi L (2017) Cifar10-dvs: an event-stream dataset for object classification. Front Neurosci 11:309
38. Soomro K, Zamir AR, Shah M (2012) Ucf101: a dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1211.2490
39. Kuehne H, Huang H, Garrote E, Poggio T, Serre T (2011) Hmdb: a large video database for human motion recognition. IEEE, pp 2556–2563
40. Kim J, Bae J, Park G, Zhang D, Kim YM (2021) N-imagenet: towards robust, fine-grained object recognition with event cameras, pp 2146–2156
41. Fang W (2020) Leaky integrate-and-fire spiking neuron with learnable membrane time parameter. arXiv preprint arXiv:2007.05785
42. Tavanaei A, Ghodrati M, Kheradpisheh SR, Masquelier T, Maida A (2019) Deep learning in spiking neural networks. Neural Netw 111:47–63
43. Sainath TN, Vinyals O, Senior A, Sak H (2015) Convolutional, long short-term memory, fully connected deep neural networks. IEEE, pp 4580–4584
44. Tran D, Bourdev L, Fergus R, Torresani L, Paluri M (2015) Learning spatiotemporal features with 3d convolutional networks, pp 4489–4497
45. Xingjian S et al (2015) Convolutional lstm network: a machine learning approach for precipitation nowcasting, pp 802–810
46. Srivastava N, Mansimov E, Salakhudinov R (2015) Unsupervised learning of video representations using lstms, pp 843–852
47. Liu Q, Ruan H, Xing D, Tang H, Pan G (2020) Effective aer object classification using segmented probability-maximization learning in spiking neural networks, pp 1308–1315
48. Patino-Saucedo A, Rostro-Gonzalez H, Serrano-Gotarredona T, Linares-Barranco B (2020) Event-driven implementation of deep spiking convolutional neural networks for supervised classification using the spinnaker neuromorphic platform. Neural Netw 121:319–328
49. Wang Q, Zhang Y, Yuan J, Lu Y (2019) Space-time event clouds for gesture recognition: from rgb cameras to event cameras. IEEE, pp 1826–1835
50. Sironi A, Brambilla M, Bourdis N, Lagorce X, Benosman R (2018) Hats: histograms of averaged time surfaces for robust event-based object classification, pp 1731–1740
51. Hara K, Kataoka H, Satoh Y (2018) Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?, pp 6546–6555

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.