Action Recognition Using Supervised Spiking Neural Networks

Aref Moqadam Mehr[1], Saeed Reza Kheradpisheh [1,*], and Hadi Farahani[1]

[1] *Department of Computer and Data Sciences, Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran*

Abstract

Biological neurons use spikes to process and learn temporally dynamic inputs in an energy and computationally efficient way. However, applying the state-of-the-art gradient-based supervised algorithms to spiking neural networks (SNN) is a challenge due to the non-differentiability of the activation function of spiking neurons. Employing surrogate gradients is one of the main solutions to overcome this challenge. Although SNNs naturally work in the temporal domain, recent studies have focused on developing SNNs to solve static image categorization tasks. In this paper, we employ a surrogate gradient descent learning algorithm to recognize twelve human hand gestures recorded by dynamic vision sensor (DVS) cameras. The proposed SNN could reach 97.2% recognition accuracy on test data.

Keywords: *Spiking Neural Networks, Dynamic Vision Sensor, hand Gesture Recognition, Surrogate Gradient, Supervised Learning.*

1 Introduction

In recent years, the fruitful coupling of artificial neural networks (ANN) and deep learning has led to the birth of powerful deep neural networks (DNN) achieving impressive results on a variety of different tasks. The backbone of DNN is a neural net with differentiable activation and cost functions to allow the employment of gradient-based learning algorithms such as gradient descent and error backpropagation. Although DNNs are believed to be largely brain-inspired and they can well predict human neural and behavioral data, there are substantial differences in how they process and learn input data.

Biological neurons use spikes to transmit and process information through the network which largely reduces the energy consumed by the brain. The aim of the spiking neural networks (SNN) is to mimic the spike-based neural processing in the brain [1]. SNNs have two important advantages over ANNs [2]. First, the lower computational complexity and energy consumption that makes them suitable for hardware implementation. Second, due to their temporal nature, SNNs can better handle sequential data like videos and sounds.

In an SNN, each neuron integrates incoming spikes from its presynaptic neurons and emits a spike whenever it reaches its threshold. Due to the non-differentiability of the thresholding activation function in spiking neurons, it is not easy to apply gradient descent and backpropagation to SNNs [3], and hence, local unsupervised learning rules such STDP were more common in the past [4–6]. One of the main solutions to this problem is the use of surrogate gradients around the spike times [7]. This way, one can easily backpropagate errors to hidden layers and update the synaptic weights.

Here in this research, we use a three-layer (input, hidden, and output) fully connected SNN with surrogate-gradient-based backpropagation al-

*Corresponding author.
Email addresses:
a.moqadammehr@mail.sbu.ac.ir (AMM),
s_kheradpisheh@sbu.ac.ir (SRK),
h_farahani@sbu.ac.ir (HF)

gorithm to recognize twelve different human hand gestures recorded by dynamic vision sensor (DVS) cameras available in [8]. Each stimulus is a sequence of spike events (representing pixel-level brightness changes) recorded from a different subject and under a different light condition during a 1000 ms time window. Our network could reach 97.2% recognition accuracy for testing hand gestures only after 100 training epochs. Interestingly the network had only 128 spiking neurons in the hidden layer. Such an impressive result shows the merits of SNNs with supervised learning rules to learn and process temporal and sequential data.

2 Dataset

The dataset used in this research is taken from the IBM open research repository [8] available at `http://research.ibm.com/dvsgesture/`. This dataset contains several human hand gestures recorded by DVS cameras under different light conditions. DVS cameras are bio-inspired vision sensors that output pixel-level brightness changes instead of standard intensity frames and their output is a stream of events (i.e., spikes) at microsecond time resolution. This dataset consists of 151 DVS data files recorded from 29 human subjects performing different hand gestures. Each data file includes 11 actions (i. e., hand gestures) listed in Table 1. Each spike is labeled to one of these eleven actions or it is labeled as No-action (the class zero). The data is captured in 5 light conditions listed in Table 2. The light condition can affect the number of spikes produced in each action.

The data is stored in the spike trail format by which the spikes are sequentially serialized one by one. Each spike has four important parameters: two parameters for the cartesian coordinates of that spike in the camera frame, one parameter for the polarity of the spike; which will be ignored for simplicity in this research, and one for the timestamp of the spike occurrence.

These spikes are then tied together in spike packets. Each packet has a different number of spikes. For instance, in a hand-clapping gesture, each packet has around 250 spikes and its time frame is around 9 ms. In this research, we consider each packet as an image frame and project its spikes in a 64 by 64 image frame and process them together. In this way, the process efficiency can be improved. We have to note that the original frame size of the camera was 128 by 128 and we have reduced this size by half to reduce the complexity.

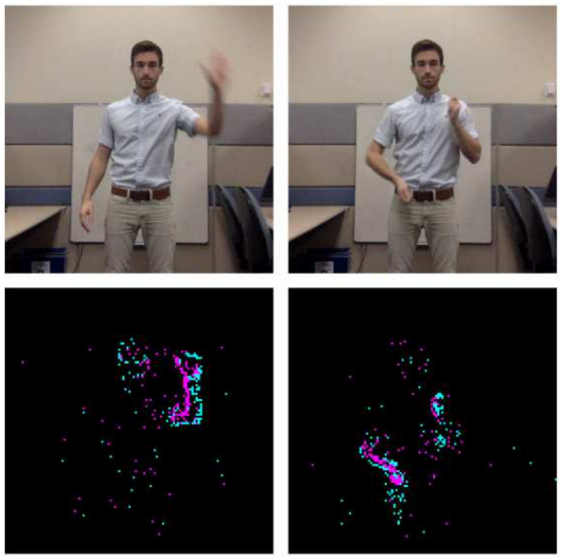

Figure 1: Sample images of the dataset taken from [8]. The top (bottom) row shows a subject performing left-hand-wave and air-drums in front of a normal (DVS) camera. The cyan and magenta dots in DVS plot represent the polarity of the DVS events.

Note that the actions in each file (corresponding to a particular subject and light condition) are continuously recorded one after another (where the order of actions is fixed among all the files) and each action happens only once in each file. So, only the beginning and the ending timestamps of each action are marked in a CSV file. To process this data easier, we have labeled each data frame regarding the mean timestamp of its spikes (by calculating where this timestamp falls regarding the CSV file).

3 Network Architecture

The network architecture used in this research is inspired by [7]. However, the mentioned research applied SNNs on the static images. To do so, it

Table 1: List of the hand gestures. There are eleven hand gestures labeled from 1 to 11. We also have a no-action class labeled as 0 which represents samples lying in between two consecutive actions when the subject is changing the action.

Hand Gestures	Labels
hand-clapping	1
right-hand-wave	2
left-hand-wave	3
right-arm-clockwise	4
right-arm-counter-clockwise	5
left-arm-clockwise	6
left-arm-counter-clockwise	7
arm-roll	8
air-drums	9
air-guitar	10
other-gestures	11

takes an image and each pixel spikes regarding its intensity multiplied by a time constant K_t - assuming that the intensity is between 0 and 1. Then the network calculates the errors over the K_t and tries to optimize its weight on this error.

The network used in our research has three layers (input, hidden, and output layer). The input layer propagates spike packets of each input stimuli one after one through the simulation time steps. The hidden layer consists of 128 Leaky-Integrate and Fire (LIF) neurons that receive spikes from the input layer and send outgoing spikes to the neurons in the output layer. Finally, the output layer which contains 12 threshold-free LIF neurons (one for each category) that receives spikes from the hidden neurons. The activation function of the output neuron is a log softmax performed over the maximum membrane potential of all the output neurons.

In order to reduce the computational complexity of the network, the forward spike propagation is done layer by layer. This means that, for any given batch of data, first, the input synaptic current and membrane potential of the hidden neurons in all the time steps are calculated. Then, the output spikes of this layer are calculated by applying the threshold function and upon that the input synaptic current and membrane potential of neurons in the output layer are computed.

Table 2: List of light conditions. Subjects perform all the hand gestures under each of these five light conditions. Different light source might have different frequency of emitting light rays, thus causing differences in DVS events pattern.

#	Light Condition
1	fluorescent
2	fluorescent_led
3	lab
4	led
5	natural

Last thing to note here is that, we have set the number of processing time steps to 100 frames. In other words, neurons perceive data from 100 consecutive frames (∼1000 ms) and then make decision about the category of the hand gesture. Note that all the 100 frames should belong to the same hand gesture, otherwise it is ignored. Also, due to the relatively small dataset size, we set the batch size to 16 to make sure enough diversity in our training set.

3.1 Neuron Dynamics

Each neuron has an input current, I_t, which is calculated by decaying the input current at the previous time point and adding the sum of the received spikes multiplied by their corresponding synaptic weights. Hence, the dynamic of the total input synaptic current is computed as

$$\frac{dI_i}{dt} = -\frac{I_i(t)}{\tau_{syn}} + \sum_j W_{ij} S_j(t), \tag{1}$$

in which, $I_i(t)$ is the total input current to the i^{th} neuron at time t. Note that the total input current decays in time with a time constant, τ_{syn}. The

term $S_j(t)$ is 1 if the presynaptic neuron j has fired at time t and 0 otherwise. The term W_{ij} is the synaptic weight connecting neuron j to neuron i.

The membrane potential of each neuron, U_i, has a decaying factor which turns it back to the resting potential, U_{rest}, with a time constant denoted as τ_{mem}. Also, at each time point, the input current, I_i, is added to the potential after being multiplied by a resistant factor, R. Finally, if the neuron reaches its threshold, θ, at time t, it will emit a spike (i.e., $S_i(t) = 1$) and send it to neurons in the next layer. After each spike, the neuron's membrane potential is reset to the resting potential. Therefore, the dynamics of the neuron's membrane potential, U_i is calculated as

$$\frac{dU_i}{dt} = -\frac{(U_i - U_{rest})}{\tau_{mem}} + RI_i + S_i(t)(U_{rest} - \theta). \quad (2)$$

By discretizing the above model, we can achieve a simpler model formulated as Eq. 3 and Eq. 4

$$I_i[n+1] = \alpha I_i[n] + \sum_j W_{ij} S_j[n], \quad (3)$$

$$U_i[n+1] = \beta U_i[n] + I_i[n] - S_i[n], \quad (4)$$

where α and β are the decaying factors of the total input current and the membrane potential, respectively. U_{rest} and θ are set to 0 and 1, respectively. We set the other parameters as $\alpha = 0.9$ and $\beta = 0.9$. All the variables I_i, U_i, S_i, and W_{ij} are in the range [0,1] and the time variable is discretized to time steps denoted as n.

3.2 Training

To train the network, first, we need to calculate the loss function and then try to adjust the weights of the network in order to minimize this function. But, before describing this process, let's briefly explain how we calculate the output of the network. As mentioned earlier, neurons in the output layer have no threshold and, hence, they do not fire at all. This means that each output neuron receives spikes from the hidden layer and update its potential via Eq. 3 and Eq. 4. Since the process is time-boxed, we can denote the network's decision by computing the maximum potential of the output neurons during the simulation time steps. Therefore, to compute the final activity of each output neuron, we apply a softmax function over theses maximum potentials. The softmax operation is denoted in Eq. 5,

$$P_i = \frac{e^{y_i}}{\sum_j e^{y_j}}, \quad (5)$$

where, y_i is the maximum potential of the i^{th} output neuron during the simulation. In other words, we calculate the probability of each class by applying a softmax on the maximum potential of the output neurons over all processing time steps. Then, to get the loss value of the current instance, a Negative Log-Likelihood function, Eq. 6, is applied,

$$L = -\log P_i, \quad (6)$$

where, i is the index of the correct class of the current instance. Now, the Backpropagation Through Time (BPTT) algorithm [7] is applied to update the synaptic weights in order to reduce the loss value (note that our network does not have recurrent connections). To do so, the derivative of the loss function concerning each synaptic weight is required to update the weights,

$$\frac{\partial L}{\partial W_{ij}^l} = \sum_n \delta_i^l[n] S_j^{l-1}[n], \quad (7)$$

where, $\delta_i^l[n]$ is the derivative of the loss function with respect to the membrane potential of the i^{th} neuron in the l^{th} layer at time step n. As the output neurons are threshold-free, there is no challenge in computing $\delta_i^o[n]$ of the i^{th} neuron in the output layer.

To calculate the δ_i^h of the i^{th} hidden neuron we have

$$\delta_i^h[n] = \sigma'(U_i^h[n]) \sum_k \delta_k^o[n] W_{ik}^l, \quad (8)$$

where, k iterates over the neurons at the output layer, o, and σ is the neuron's activation function (in our case it is the thresholding function).

However, $\delta_i^h[n]$, can not be calculated due to the non-differentiability of σ, the thresholding activation function of LIF spiking neurons. Thus, calculating the error portion of hidden neurons is not possible in traditional ways. Hence, we have to use

a surrogate as the gradient of the threshold function. This approach is known as the Surrogate Gradient Descent and is described in details in [7]. To do so, instead of using the derivative of the thresholding activation function (which is undefined at the spike time and 0 elsewhere), we have used the following smooth surrogate gradient function (the derivative of the negative half of fast sigmoid),

$$\sigma^{'}(U_i^h[n]) = (\gamma * |I_i| + 1.0)^{-2}, \tag{9}$$

where, γ is a scaling factor. More details about this function can be found in [9].

The weights are updated through Surrogate Gradient Descent and backpropagation algorithms to minimize the loss value. To this end we have used Adam Optimizer. However, there are few more hyperparameters including the number of frames in each data and the length of each frame that can affect the learning process. For the latter, we used 9.9 ms frame length, as equal as the camera packet length and for the former, we used 100 frames. This means the network tries to predict the hand gesture based on the past 100 frames.

4 Results

As for the results, the original dataset has 122 data files for training and 29 data files for testing. The test files were completely opted out from the training process and used only for determining the accuracy and reliability of the developed algorithm. Note that the hand gestures of each subject are recorded under different light conditions and we have used the data from all light conditions to train and test the network.

The trajectory of the loss value of the proposed network over the training and testing sets during the training epochs are plotted in Fig. 2. As seen, the loss value over both training (the blue curve) and testing (the dashed red curve) samples decrease through the epochs and finally converge to very small values. Hence, the proposed SNN would be able to well generalize what it has learned from the training samples to the unseen testing samples.

In the final result, we achieved 97.2% recognition accuracy on the test set and 96.5% on the train set. Such an impressive result indicates the power of SNNs to handle temporally dynamic inputs like videos and sounds. Indeed, each neuron is acting like a spatiotemporal filter as it receives incoming spikes through time and reports the presence of its learned pattern by sending spikes to its upstream neurons. Note that the employed backpropagation learning algorithm with surrogate gradients is playing a key role here as it updates the weights in the direction of minimizing the overall loss function. Hence, it can be concluded that the surrogate gradient we used is providing a suitable approximation to the gradient of the thresholding activation function of spiking neurons.

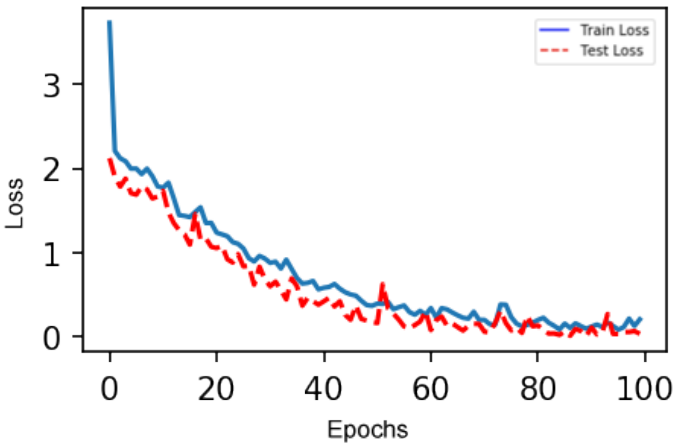

Figure 2: The trajectory of the loss value over the training (blue curve) and testing (red dashed curve) samples across the training epochs.

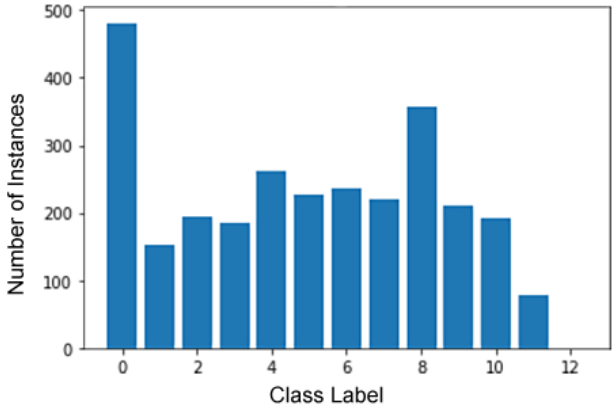

Figure 3: The number of testing samples belonging to each of the twelve hand gesture categories.

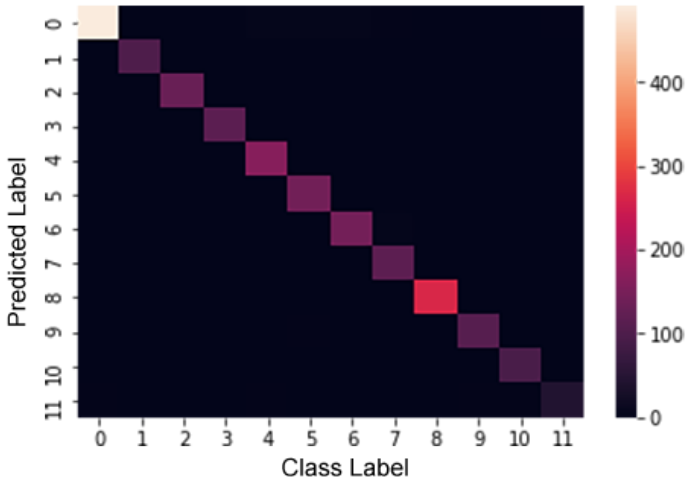

Figure 4: The confusion matrix of the proposed SNN over the testing samples. Rows (columns) represent the actual (predicted) hand gesture categories.

The number of testing samples for each of the twelve categories are provided in Fig. 3. As mentioned before, the zero label is assigned to the no-action category which contains those samples falling in between the two actions when the subject is changing his/her action. Fig. 4 presents the confusion matrix of the twelve hand gestures. The rows (columns) represent the actual (predicted) categories. The element on the i^{th} row and j^{th} column of this confusion matrix shows the number of samples from the i^{th} category being predicted to belong to the j^{th} category. As seen, the network is not biased to any category and the error is uniformly distributed over all the categories.

5 Discussions

These days, SNNs are getting more and more popular in the area of neural networks and are being used in different applications [10]. However, SNNs are still suffering from the lack of efficient supervised learning algorithms [3]. Recent efforts have succeeded to extend the gradient-based backpropagation algorithm to SNNs by using surrogate gradients for the non-differentiable thresholding activation function of spiking neurons [7]. Reported results have indicated the capacity of SNNs with surrogate-gradient-based backpropagation to solve static image categorization tasks [9, 11–13].

Here, we developed a three-layer fully connected SNN of LIF neurons with a surrogate-gradient-based backpropagation learning rule to solve human hand gestures recorded by DVS cameras. Our codes are available at https://github.com/ArefMq/action-recognition-via-snn.

The obtained results indicate that the proposed network can accurately distinguish twelve different hand gestures. It is a sign for the potentials of applying supervised SNNs in action recognition tasks. However, the current implementation needs the data to be split into K-frame episodes and it can be improved in future studies to process a real-time stream of spikes.

Another further research is to extend the proposed network into a convolutional architecture. In general, convolutional networks can better handle positional variations and are less sensitive to scale variations. Hence, if the subject moves its position or distance to the camera, it is expected that convolutional SNN would yet be able to keep its performance in recognizing the performed actions.

References

[1] A.Taherkhani, A. Belatreche, Y. Li, G. Cosma, L. P. Maguire, T. M. McGinnity. A review of learning in biologically plausible spiking neural networks. Neural Networks (2019) In Press, DOI: 10.1016/j.neunet.2019.09.036

[2] S. R. Kheradpisheh, T. Masquelier, S4NN: temporal backpropagation for spiking neural networks with one spike per neuron. arXiv (2019) 1910.09495.

[3] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, A. Maida, Deep learning in spiking neural networks, Neural Networks 111 (2019) 4763.

[4] T. Masquelier, S. J. Thorpe, Unsupervised learning of visual features through spike timing dependent plasticity, PLoS computational biology 3 (2) (2007) e31.

[5] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, T. Masquelier, Stdp-based spiking deep convolutional neural networks for object recognition, Neural Networks 99 (2018) 5667.

[6] R. Vaila, J. Chiasson, V. Saxena, Deep Convolutional Spiking Neural Networks for Image Classification. arXiv (2019) 1903.12272.

[7] E. O. Neftci, H. Mostafa, F. Zenke, Surrogate gradient learning in spiking neural networks, arXiv (2019) 1901.09948.

[8] A. Amir, et al. A low power, fully event-based gesture recognition system. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. pp. 7243-7252.

[9] F. Zenke,, and S. Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. Neural computation 30.6 (2018) 1514-1541.

[10] M. Pfeiffer, T. Pfeil, Deep learning with spiking neurons: opportunities and challenges, Frontiers in neuroscience 12 (2018) 774.

[11] S. M. Bohte, Error-backpropagation in networks of fractionally predictive spiking neurons, in: International Conference on Artificial Neural Networks, Springer, 2011, pp. 6068.

[12] E. O. Neftci, H. Mostafa, F. Zenke, Surrogate gradient learning in spiking neural networks, arXiv (2019) 1901.09948.

[13] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, W. Maass, Long short-term memory and learning-to-learn in networks of spiking neurons, in: Advances in Neural Information Processing Systems, 2018, pp. 787797.