AUT Journal of Electrical Engineering

AUT J. Electr. Eng., 55(2) (2023) 179-190
DOI: 10.22060/eej.2023.21991.5502

DS4NN: Direct training of deep spiking neural networks with single spike-based temporal coding

Maryam Mirsadeghi[1], Majid Shalchian[1]*, Saeed Reza Kheradpisheh[3]

[1] Department of Electrical Engineering, Amirkabir University of Technology, Tehran, Iran
[2] Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran

ABSTRACT: Backpropagation is the foremost prevalent and common algorithm for training conventional neural networks with deep construction. Here we propose DS4NN, temporal backpropagation for deep spiking neural networks with one spike per neuron. We consider a convolutional spiking neural network consisting of simple non-leaky integrate-and-fire (IF) neurons, and a form of coding named time-to-first-spike temporal coding in which, neurons are allowed to fire at most once in a specific time interval, which corresponds to simulation duration here. These features together improve the cost and the speed of network computation. We use a surrogate gradient at firing times to solve the non-differentiability of spike times concerning the membrane potential of spiking neurons, and to prevent the emergence of dead neurons in deep layers, we propose a relative encoding scheme for determining desired firing times. Evaluations on two classification tasks of MNIST and Fashion-MNIST datasets confirm the capability of DS4NN on the deep structure of SNNs. It achieves the accuracy of 99.3% (99.8%) and 91.6% (95.3%) on testing samples (training samples) of respectively MNIST and Fashion-MNIST datasets with the mean required number of 1126 and 1863 spikes in the whole network. This shows that the proposed approach can make fast decisions with low-cost computation and high accuracy.

Review History:

Received: Dec. 13, 2022
Revised: Feb. 05, 2023
Accepted: Feb. 27, 2023
Available Online: Oct. 10, 2023

Keywords:

Deep spiking neural network
Temporal backpropagation
Single spike-based coding
Supervised learning
Integrate-and-fire neuron model

1- Introduction

Spiking neural networks (SNNs) have received much attention as the third generation of artificial neural networks (ANN) in scientific circles. They have a temporal and event-driven processing nature which makes them more efficient and powerful than the traditional ANNs. However, their computational power cannot be fully used due to the limited number of supervised learning methods. The most prominent supervised learning method for artificial neural networks is the backpropagation algorithm (BP) which is based on the gradient descent method to minimize the output error [1]. Due to the temporal nature of SNNs and, the use of dynamic units and discrete activation functions, the backward flow of the error in the BP algorithm cannot be directly applied to spiking neural networks. Therefore, supervised training of SNNs has remained an unsolved problem so far. Numerous solutions and methods have been proposed for training SNNs by using BP, which can be divided into four categories: 1- Conversion approach, 2- tandem learning, 3- proxy learning, and 4- direct training methods and, we focus on the last category.

In the conversion methods, firstly, backpropagation (BP) is employed to train ANN and then convert it to an equivalent SNN with the same structure [2-7]. In tandem learning, SNN and ANN are coupled layer by layer with shared weights. In the forward path, the output of each SNN layer converts to spike counts which is fed to the ANN layer as input. In the backward pass, each ANN layer receives the spike counts and updates the shared weights [8, 9]. In proxy learning, an SNN is trained via a proxy ANN [10]. Two networks are independent in the forward path. After computing the final output of SNN, the SNN error is calculated and replaced in the equivalent ANN to update the shared weights.

Although the approaches mentioned above could be applied to the deep structure of SNNs with state-of-the-art accuracies, they are based on rate-coding or multi-spike schemes and they do not consider temporal coding in which each neuron emits at most one spike during a specific time interval. In temporal coding schemes based on a single spike [11-22], information is encoded in the spike times where neurons are allowed to spike once. such coding can significantly reduce the computational cost and energy demand of SNNs compared to rate-based coding paradigms.

In direct training methods, the aim is to directly apply backpropagation to SNNs with temporal [11-14, 23-25] or rate coding [26-30]. Here, we focus on single-spike temporal coding. Bohte, et al. [23] introduced Spikeprop, which directly trains a single-spike-coded multilayer SNN using a temporal version of BP. They used an exponential spike-response model (SRM) for neurons and updated the synaptic weights to minimize the temporal error of the network. Mostafa [24] defined the neuron firing time as a differentiable function of the firing times of its afferents and employed

*Corresponding author's email: shalchian@aut.ac.ir

gradient descent to minimize the temporal error. He used integrate-and-fire neuron models with exponential decaying functions and reached the state-of-the-art performance for a multi-layer single-spike-based SNN. Comsa et al. [25] used the same method and employed neuron models with alpha synaptic function which is computationally expensive. In [11], Zhang et al. proposed to use of Rectified Linear Postsynaptic Potential function (ReL-PSP) for spiking neurons to overcome the discontinuity of the spike function and employed temporal BP to train a multi-layer SNN with time-to-first-spike coding. Gardner et al. [31] applied the BP learning algorithm to a network with first-to-spike decoding. They defined the cost function of the network over the first spike arrival times in the output layer, while other neurons in the hidden layer are based on rate-coding schemes. Kherapisheh et al. in [12] proposed S4NN which applies the temporal version of traditional BP to multilayer SNNs with one spike per neuron. To do so, the temporal error is calculated as the difference between the target and actual firing times of output neurons, and the gradient descent is applied to minimize the error. They employed simple non-leaky IF neurons with time-to-first-spike temporal coding to reduce the computational cost. Mirsadeghi et al. [13] introduced the STiDi-BP learning algorithm in which, the backward recursive gradient computation is eliminated. They employed a linear SRM spiking neuron model with one spike per neuron, and in each layer, local Gradient descent (GD) is applied to minimize the local temporal error.

However, there are several challenges to directly applying BP on deep SNNs with temporal coding (one spike per neuron) including the emergence of dead neurons in the middle layers and the gradient explosion that unstabilized learning in the in-entry layers. There are few works that succeeded to overcome these challenges. Zhang et al. in [11], used their proposed learning algorithm in a deep convolutional spiking neural network consisting of two convolutional and two hidden layers and achieved good results. In [14], the authors extended STiDi-BP for training deep structures of SNNs and reached the plausible performance in a deep convolutional spiking neural network (CSNN) on the fashion product images.

In this paper, we extend the S4NN learning algorithm [12] to overcome the aforementioned challenges and apply them to deep convolutional SNNs. We use a dynamic encoding scheme to define target firing times and use a surrogate gradient to calculate the derivative of the firing times with regard to membrane voltage and solve the non-differentiability of the spiking function. Using IF neurons as the simplest model of a spiking neuron, a sparse temporal coding in which each neuron fires at most once, and, a temporal version of BP to directly train the network make it to decide as accurately and fast as possible.

The simulation results for the categorization task on two datasets of MNIST and Fashion-MNIST, as two popular benchmarks, confirm the feasibility of the proposed algorithm on deep CSNNs.

2- DS4NN Learning Approach

Here we propose a novel supervised learning algorithm, DS4NN, which is the modified version of the S4NN learning approach [12] to make it applicable in deep convolutional spiking neural networks. DS4NN employs temporal backpropagation for the deep single spike-based temporal SNNs in which all neurons fire only once per stimulus and their exact firing times carry information.

2- 1- Forward path

In forward propagation, sparse spikes representing input patterns are presented to the network for estimating the network outputs. The proposed spiking neural network has a convolutional structure that includes an input layer, a set of convolutional layers and pooling layers that are used to extract features and placed one after the other, and a stack of fully connected layers for classification. At the entry layer, efficient temporal coding (time-to-first-spike coding) is used to convert the pixel values of the input images to individual spikes in a sparse manner. Neurons that correspond to higher-value pixels emit spikes earlier and each neuron produces only one spike per pixel. We consider that the pixel intensity values of each image are in the range [0, $I_{\max}$]. Therefore, the following equation is used to convert a pixel value to a single spike time:

$$t_i = [\frac{I_{\max} - I_i}{I_{\max}} t_{\max}], \tag{1}$$

where, $t_{\max}$ is the maximum simulation time and t_i is the firing time of i^{th} the input neuron, which is obtained by encoding the value of the i^{th} pixel (I_i).

After encoding each input image and injecting it into the network, neurons in the subsequent convolutional layers integrate the voltage sum of all the presynaptic spikes received from their receptive field to update their membrane voltage. Each convolutional layer consists of integrate-and-fire (IF) neurons organized in several feature maps. Each feature map corresponds to a convolutional filter. The parameters of filters should be learned. The membrane potential of the j^{th} neuron in the l^{th} layer at time step t, $V_j^l(t)$, is calculated as

$$V_j^l(t) = V_j^l(t-1) + \sum_{i \in I} w_{ji}^l S_i^{l-1}(t), \tag{2}$$

where S_i^{l-1} is the input spike from the presynaptic neuron i and w_{ji}^l is the input synaptic weight between the i^{th} presynaptic neuron in the previous layer and the postsynaptic neuron j. i iterates over the presynaptic neurons. Whenever the membrane potential of the IF convolutional neuron reaches the threshold v_{th} and the neuron has not fired at any previous time step, it emits a spike and remains silent until the end of the simulation. This feature is described by:

$$S_i^l(t)=\begin{cases}1 & if\ V_i^l \geq v_{th} \quad \& \quad S_i^l(<t)\neq 1\\ 0 & Otherwise\end{cases} \tag{3}$$

The spike time of each convolutional neuron determines the saliency of the extracted feature. Weight sharing happens across the receptive field of the neurons (filters) in a particular layer. It means that there are fixed weights for each filter across the entire input and neurons in a specific map detect the same feature at different locations.

After one or more convolutional layers, a pooling layer is incorporated. The pooling layers are implemented by IF neurons with the input synaptic weights and the threshold of one. Therefore, the first input spike from the neighboring afferent neurons activate the pooling neuron and make it to fire a spike immediately. Note that the learning process is not performed for the pooling neurons. Each pooling neuron fires only once and emits the earliest input spike time as the output, therefore, according to the time-to-first-spike coding, it does a nonlinear max pooling operation on a set of neighboring neurons. This reduces the size of feature maps and the number of parameters and computations by down-sampling the representation.

The output of the last convolutional or spike-pooling layer is converted to a one-dimensional vector (it is no longer represented by a matrix but by a vector) and, fed to the classification layer. It includes one or more fully-connected hidden layers and one output layer. The extracted features are processed in the hidden layers where each IF neuron updates its membrane voltage according to Eq. (2) and fires a spike once its membrane voltage crosses the threshold. The same process has been performed in the following fully-connected layers. In the output layer, the number of neurons and classes are equal. Each output neuron represents a class. The output neuron with the smallest firing time is the winner neuron and specifies the input image category.

2- 2- Backward path

Following forward propagation of the input image, each output neuron fires once at a specific time step. The index of the output neuron with the earliest firing time defines the category of the input image. Here, the modified S4NN learning rule [12] is applied to perform categorization tasks in the deep convolutional SNNs. First, we compute the temporal error by calculating the difference between the actual firing times of the output neurons and the target firing times. Target firing time determines the class the input belongs. Then, we use the Adam optimizer and backpropagation algorithm, to update the network parameters and minimize the output error.

In the following, we show how to compute the target spike firing times and, express the details of the proposed learning algorithm.

2- 2- 1- Encoding target firing times

The appropriate definition of desired firing times has a great effect on preventing the occurrence of dead neurons, which is a big challenge in deep SNNs. Here we use a relative encoding scheme [13] in which, the target firing times of output neurons for each input image are determined dynamically concerning the output actual firing times. We assume a classification task with C classes, and the input image belongs to the j^{th} category, $\tau_{\min}$ and $\tau_{\max}$ are the minimum and maximum firing times in the output layer. The target firing time of output neuron i is calculated as

$$T_i^l=\begin{cases}\tau_{\min}-\lambda & if\ i=j\\ \tau_{\max}+\lambda & if\ i\neq j,\end{cases} \tag{4}$$

where, λ is a consistent parameter used to separate the correct neuron and help it to fire earlier. Here, the upper and lower limits are equal to 0 and $T_{\max}$ (the maximum simulation time). We consider T_i^l equal to $T_{\max}$, when, $\tau_{\max}+\lambda$ becomes larger than $T_{\max}$ and, equal to 0, when, $\tau_{\min}-\lambda$ becomes smaller than 0. With this encoding, the output neuron corresponding to the category of the input image is forced to fire earlier and others are forced to fire at later times.

2- 2- 2- Temporal backpropagation learning rule

In the training phase, the network parameters should be adjusted so that the output neuron corresponding to the input category emits a spike at earlier times. To do so, first, we calculate the output temporal error and then, the gradient descent method is applied to update the weights. Here we use the Adam optimizer [32] to adjust the parameters and the learning rate.

For each output neuron k, the temporal error is measured as a difference between its target firing time T_k^{out} and its actual firing time t_k^{out} predicted by the network, and the cost function is the least mean squares of these temporal errors (as expressed in Eq. (6))

$$e_k^{out}=t_k^{out}-T_k^{out}, \tag{5}$$

$$E^{out}=\sum_k E_k^{out}=\sum_k \frac{1}{2}(e_k^{out})^2. \tag{6}$$

Then, the gradient of the cost function δ^l is estimated at the output layer, and it is backward propagated to the hidden layers using the chain rule to update the parameters and reduce the output error, as shown in Eq. (8):

$$\delta^l=\frac{\partial E^{out}}{\partial t^l}, \tag{7}$$

Table 1. Model parameters for the MNIST dataset

layer	η	λ	initial weights
1^{st}	0.1	10	[0, 80]
2^{nd}	0.1	10	[0, 50]

$$\frac{\partial E^{out}}{\partial w^l} = \frac{\partial E^{out}}{\partial t^l}\frac{\partial t^l}{\partial V^l}\frac{\partial V^l}{\partial w^l}. \quad (8)$$

The first term of Eq. (8) (the gradient to the error function) for the output neuron k is expressed as follows

$$\delta^l = \frac{\partial E^{out}}{\partial t_k^{out}} = e_k^{out}, \quad (9)$$

and to propagate the error to the deeper layers (middle layers l), δ_j^l is computed based on the following equation:

$$\delta_j^l = \sum_i \frac{\partial E^{out}}{\partial t_i^{l+1}}\frac{\partial t_i^{l+1}}{\partial V_i^{l+1}}\frac{\partial V_i^{l+1}}{\partial t_j^l} = \sum_i \frac{\partial E^{out}}{\partial t_i^{l+1}} w_i^{l+1}[t_j^l \le t_i^{l+1}], \quad (10)$$

here, i iterates on the postsynaptic neurons in layer $l+1$, t_j^l is the firing time of presynaptic neuron j in layer *l,* and t_i^{l+1} is the spike latency of postsynaptic neuron i in layer $l+1$. If the presynaptic neuron j fires after the postsynaptic neuron i, it has no contribution to the above computation.

At each layer l, the backpropagated gradients δ^l should be normalized before updating the synaptic weights to avoid exploding gradients.

$$\delta_j^l \leftarrow \frac{\delta_j^l}{\sum_i \delta_i^l}. \quad (11)$$

The second term of (8), the derivative of the postsynaptic firing time concerning its membrane voltage, is a challenging term due to the temporal and discrete nature of SNNs. Here we employ a surrogate gradient to approximate this derivative [12, 33-35]. As a prior knowledge, there is a linear relationship between the input and output of the artificial neuron with Rectified Linear Units (ReLU) [36] activation function so that a larger net input causes a greater output value. On the other hand, according to the time-to-first-spike temporal coding in SNNs, larger values correspond to earlier spikes. If an IF neuron receives these early spikes through strong synaptic weights, it will also fire earlier. Hence, an IF neuron can approximate the functionality of a ReLU neuron. It remains silent if the output of ReLU is zero, and it will fire at earlier times for larger values of ReLU outputs. Therefore, we can surrogate the second term of Eq. (8) with the negative of the output derivative of the ReLU neuron with respect to its input ($\frac{\partial t_i^l}{\partial V_i^l} \to -1$).

Finally, the third term of (8) can be calculated using Eq. (2):

$$\frac{\partial V_i^l(t)}{\partial w_{ji}} = S_j^{l-1}(t). \quad (12)$$

Where, $S_j^{l-1}(t)$ is 1 if the presynaptic neuron j emits a spike at time t, else, it has the value of zero. It confirms that the backward computation is valid only at the firing times, not all the time steps which are due to the use of first-time-to-spike-based temporal coding.

3- Experiment results

3- 1- MNIST dataset

In this section, we evaluate the DS4NN algorithm on the MNIST dataset of handwritten digits which contains 60000 (10000) 28×28 training (testing) images. Although this problem is largely solved using traditional deep convolutional neural networks, MNIST still poses a challenge for the solutions based on single-spike temporal SNNs. Our proposed CSNN consists of two convolutional layers, both are followed by pooling layers. The first and second convolutional layers consist of 64 and 128 neuronal maps with conv-window sizes of 3×3 and $3\times3\times64$ respectively and the threshold voltage for firing of neurons is 100 mV. In each pooling layer, there is a pooling window of size 2×2 and the stride value is 2. The output layer has 10 IF neurons. The maximum simulation is 100 ($T_{\max}=100$) and other parameters are given in Table 1.

In Table 2, we present the classification accuracy of the proposed DS4NN along with some recent works which are based on direct training of single spike-based temporal SNNs. Most of the works [24,25,12,13] are based on fully-connected implementation with shallow structure. For example, S4NN

Table 2. Comparison of the classification accuracies between some recent works that employed supervised algorithms to directly train temporal SNNs on the MNIST dataset. We represent the convolution and pooling layers by C and P, respectively, and separate layers by -

Model	Structure	Accuracy(%)
Mostafa [24]	784-800-10	97.2
Comsa et al. [25]	784-340-10	97.9
Kheradpisheh et al. [12]	784-400-10	97.4
Mirsadeghi et al. [13]	784-350-10	97.4
Zhang et al. [11]	784-800-10	98.5
Mirsadeghi et al. [14]	40C5-P2-1000-10	99.2
Zhang et al. [11]	16C5-P2-32C5-P2-800-128-10	99.4
DS4NN (This work)	64C3-P2-128C3-P2-10	99.3

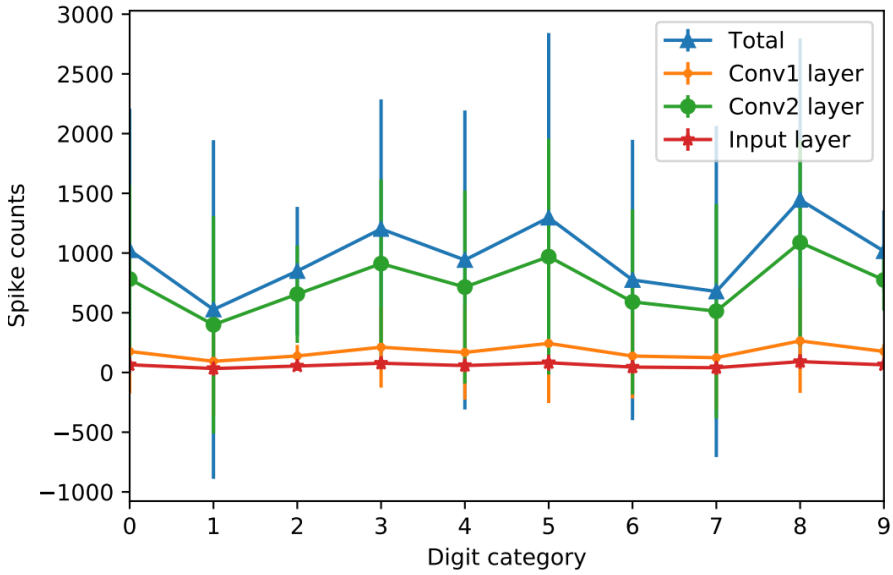

Fig. 1. The mean required number of spikes in each layer of the proposed CSNN for MNIST dataset

algorithm [12] is only applied to a shallow fully-connected SNN with one hidden layer and 400 neurons and achieves the accuracy of 97.4%. There are limited works on deep SNNs due to some challenges. One important challenge is the non-propagation of errors to the deeper layers caused by the dead neurons in the middle layers. Zhang et al. in [11] applied temporal BP to a CSNN consisting of two convolutional layers with 16 and 32 neural maps and two hidden layers with 800 and 128 neurons (16C5-P2-32C5-P2-800-128-2). They used a simple spiking neuron model with rectified linear Post-Synaptic-Potential and reached the accuracy of 99.4%.

In [14], Mirsadeghi et al. modified the STiDi-BP algorithm to make it practical for deep SNN and achieved a performance of 99.2% for the MNIST dataset classification. Here we introduce DS4NN, as the modified version of S4NN. We reach the state-of-the-art accuracy of 99.3% on the testing samples for a deep SNN with two convolutional layers (64C3-P2-128C3-P2-10) and the accuracy of 99.8% has been reached on the training set.

Low cost and rapid computation are the most important advantages of single spike-based SNNs over rate-based SNNs and traditional ANNs. This fact is illustrated in Fig.1 and Fig.2. In Fig.1, the mean number of spikes that are emitted in all layers is depicted. Each input image is recognized by firing a limited number of neurons and by producing a limited number of spikes in the whole network. For each output neuron, the average firing time over the input images of all classes is shown in Fig.2. The correct output

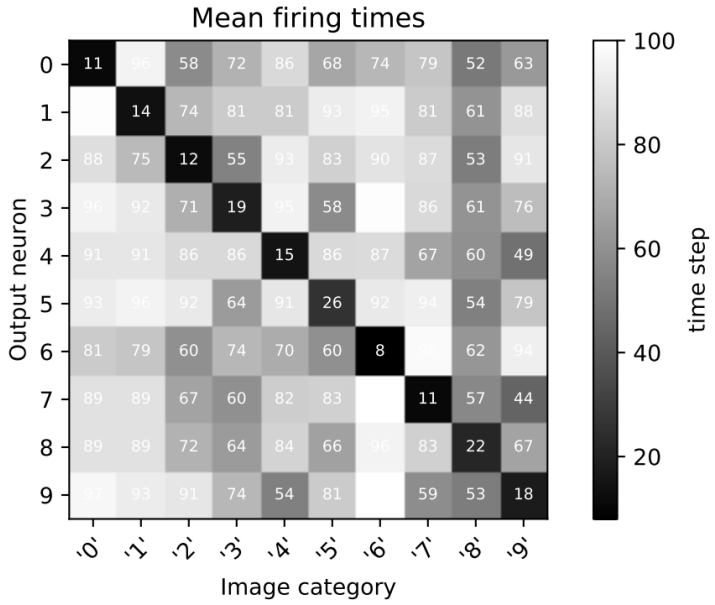

Fig. 2. The average firing time of each output neuron over the images of different digit categories of the MNIST dataset in the proposed CSNN

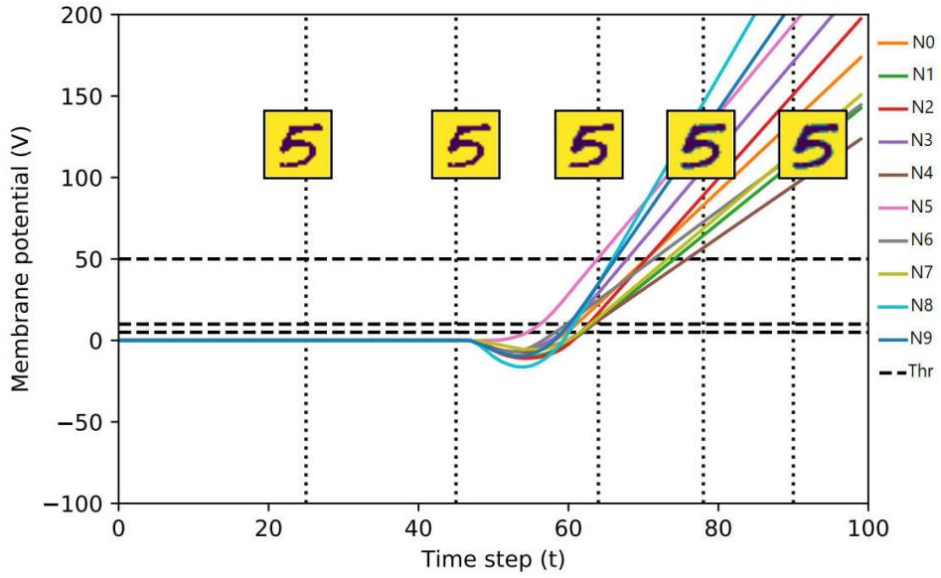

Fig. 3. The membrane potential of all output neurons for '5' pattern in the test image. The network can determine the class of input image by emitting only input spikes up to the time step 63 and other spikes are ignored

neuron corresponds to the category of the input image that fires earliest. This result confirms that the network can make decisions rapidly by firing a restricted number of neurons at each layer. Fig.3 shows that it is not required to emit all spikes presented in the input image to complete the classification task. According to Fig.3, the membrane potential of the correct output neuron (5^{th} output neuron corresponds to '5' pattern in test image) crosses the threshold at time step 63. Therefore, the propagation of a few spikes up to the time step 63 is enough for the network to complete the classification task.

The confusion matrix of the proposed learning algorithm is presented in Fig.4 showing that the images related to different classes are greatly contrasted with each other and this causes the network to categorize the images with high accuracy.

On average, the proposed learning algorithm makes decisions with an accuracy of 99.3% in 15.6 time-step by

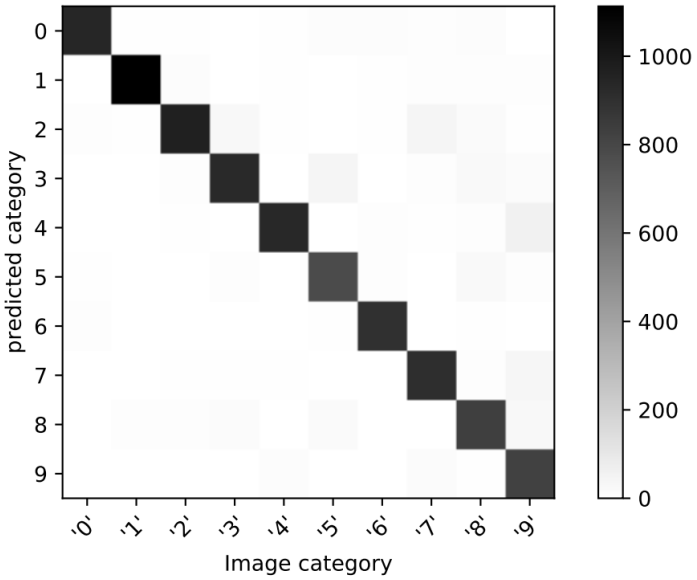

Fig. 4. The confusion matrix of the proposed CSNN on MNIST dataset

Table 3. Comparing the number of dead neurons presented in different methods of determining target firing time

Method	Num. of dead neurons
Proposed method	61%
Kheradpisheh et al [12]	95%
Mostafa [24]	95.4%
Comsa et al. [25]	95.4%

producing only 1126 spikes in the whole network. It confirms that the proposed network operates in a sparsely manner with fast and accurate computation. Also, it overcomes the occurrence of dead neurons that occur in S4NN due to the use of a proper method for determining the target firing time of output neurons, which is shown in Table 3.

Table 3 compares the number of dead neurons in the proposed method with the number of dead neurons reported in [12, 24, 25]. In the proposed network, 61% of the neurons in the whole network do not emit any spikes, which are called dead neurons. While, if we employ the method of [12] to determine the target firing times, 95% of neurons in the whole network become dead neurons and, the method used in [24] and [25] causes the death of 95.4% of neurons. This causes a sharp drop in network performance.

3- 2- Fashion-MNIST dataset

Here we employ Fashion-MNIST [37], a more challenging dataset than MNIST, to better evaluate the proposed learning algorithm. The Fashion-MNIST dataset contains 28×28 grey-scale images of clothing items with 10 classes. The proposed network has the structure of 128C3-P2-128C3-P2-128C3-10 that consists of three convolutional layers with 128 neuronal maps and conv-window sizes of 3×3, $3\times3\times128$ and $3\times3\times128$ respectively. Each layer has a voltage threshold of 100. First, the learning rate is set to 10^{-1} and decays through the learning epochs by an exponential function. The initial value of synaptic weights for each convolutional layer are respectively in ranges [0, 80], [0, 60], [0, 50], and the value of λ is 0.1. The pooling layers have a pooling window size of 2×2 with a stride of 2.

In Table 4 we compare the classification accuracy and the structure of different temporal coding-based SNN methods on the Fashion-MNIST dataset. [38], [12] and [11] reached an accuracy of 87.3%, 88.0%, and 88.1% when they only used a fully-connected SNN with one hidden layer consisting of 1000 hidden neurons. Up to now, [11] and [14] and this work are the only implementations of deep single spike-based SNNs. The proposed DS4NN reaches an accuracy of 95.3% on the training set and the state-of-the-art accuracy of 91.6%

Table 4. Comparison of the classification accuracy and network structure reported by several recent works that employed a supervised algorithm to directly train temporal SNNs on the Fashion-MNIST dataset. We represent the convolution and pooling layers by C and P, respectively, and separate layers by -

Model	Structure	Accuracy(%)
Kheradpisheh et al. [38]	784-1000-10	87.3
Kheradpisheh et al. [12]	784-1000-10	88.0
Zhang et al. [11]	784-1000-10	88.1
Zhang et al. [11]	16C5-P2-32C5-P2-800-128-10	90.1
Mirsadeghi et al. [14]	20C5-P2-40C5-P2-1000-10	92.8
DS4NN (This work)	128C3-P2-128C3-P2-128C3-10	91.6

Table 5. The average firing time (AFT) of the correct output neuron and the mean required number (MRN) of spikes emitted in all layers of the network over each class of Fashion-MNIST

Category	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
AFT	37	22	40	30	32	31	45	20	21	17
MRN	2709	1925	2663	1251	1885	1312	2596	1038	3439	1695

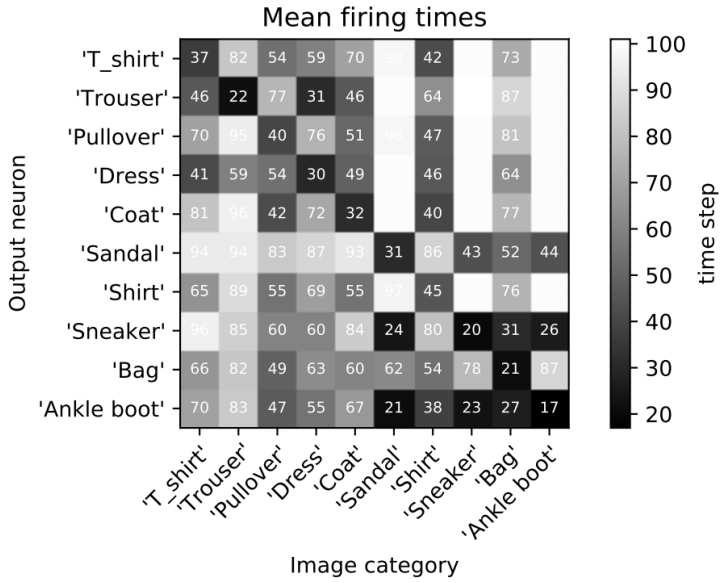

Fig. 5. The average firing times of the output neurons in the Fashion-MNIST classes in the proposed CSNN

on the testing samples.

The average spike time that output neurons fire and the mean number of spikes produced in the whole network are listed in Table 5. Similar to the result obtained for MNIST, to recognize the image class, it is not required to transmit all spikes of the input image, and, the classification task is accomplished by sending just a few spikes of the encoded input image to the network.

In Fig.5, we illustrate the average spike times that output neurons fire for the Fashion-MNIST dataset. For a few image categories including “Sandal” against “Sneaker” and “ankle boot”, the output neuron corresponding to the input image, does not fire much earlier than other output neurons due to the resemblance of those classes. This limits the accuracy of the D4SNN algorithm and is depicted in the confusion matrix of Fig.6.

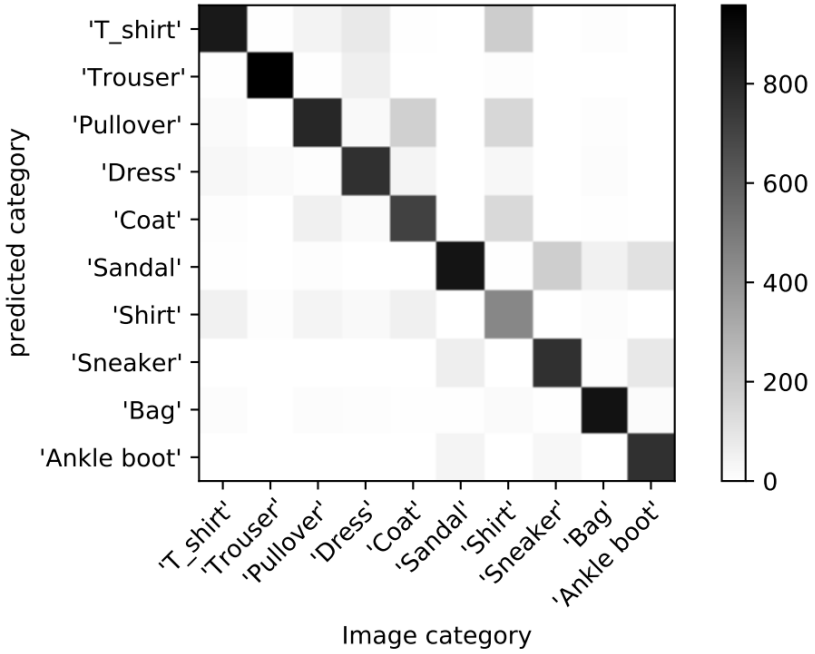

Fig. 6. The confusion matrix of the proposed CSNN on Fashion-MNIST

According to Fig.6, the network confuses ankle boots, sandals, and sneakers. The same goes for T-shirts, shirts, dresses, coats, and pullovers, where, the output neurons corresponding to these categories have similar average firing times.

4- Discussion

In this paper, we proposed a temporal version of backpropagation, DS4NN, for training deep single spike-based SNNs, directly. To this end, we extended S4NN introduced in [12] and employed it in an SNN with convolutional architecture (CSNN). In the forward path, the input image is encoded into a spike train using time-to-first-spike coding and fed into the network. Then, the convolutional and the max. pooling operations are applied to the input spike train and the extracted features are given to the subsequent hidden and output neurons in the classification layer. Finally, the network makes decisions by the first spike in the output layer. Here we used instantaneous synapses and non-leaky IF neurons as the simplest model of spiking neurons, which reduces the computational cost of the proposed CSNN. As soon as the neuron fires, it resets and forgets its state. Therefore, the corresponding memory can be used by other neurons, which makes the proposed method memory efficient. In the backward path, we applied the temporal version of the backpropagation algorithm. We calculate the temporal error by subtracting the actual and target firing times, and, the calculated gradients are backpropagated through the network to update the synaptic weights by using the Adam optimizer. To prevent exploding and vanishing gradients, we normalized the backpropagated gradients at each layer, and then, we updated the weights. Here we used a dynamic temporal encoding to define the desired firing times, which are dependent on the output spike latencies and the category label of the input image. The output neuron corresponding to the class label should be forced to fire earlier and the other output neurons are forced to fire later. Determining the proper target firing time for each input image has a great impact on reducing the number of dead neurons and increasing the classification accuracy of the network. Here to prevent extreme changes in the weights, the learning rate parameter η discounts by 10% every 5 epochs.

Spiking neural networks are more suitable for parallel processing compared to traditional neural networks, due to the fact that they are asynchronous. However, the temporal and discrete nature of spiking neurons makes the training of SNNs difficult. There are some approaches for supervised learning of temporal SNNs with at most one spike per neuron [23-25, 11-14].

In single spike-based temporal SNNs [23-25, 11-14], neurons transmit information by the timing of individual spikes rather than multi spikes, which makes the network decide before most neurons have fired. Also, gradients are backpropagated through the network only at the actual firing

times, which causes the space complexity of each layer to become O(N), while, in rate-based coding schemes, due to the backward computation in all steps of the simulation time the space complexity is O(NT).

Here, we employ the IF neuron model in all layers which has made the proposed algorithm more computationally efficient. While, other approaches [11, 13, 24, 25] have complex neural processing due to the use of complicated neuron models. In [24], a non-leaky integrate and fire neurons with an exponentially decaying function was used. [25] employed spike response neuron model (SRM) with alpha synaptic function, Zhang et. al in [11] introduced rectified linear PSP based spiking neurons, and Mirsadeghi et. al [14] employed piecewise linear neuron models in all layers.

Training of single spike-based SNNs, especially their deep structure, is a great challenge due to the occurrence of dead neurons in the middle layers that prevent the backpropagation of errors [23-25, 12, 13]. There are few works in this area that try to train deep temporal SNNs with single spike-based coding [11, 14]. Here we focus on this topic. We improved the target firing time determination method in S4NN to avoid the occurrence of dead neurons and reduced the learning rate by 10% every 5 epochs to avoid large changes in weights and network divergence. Also, we replaced Adam's optimizer with stochastic gradient descent (SGD). Therefore, we modified the S4NN algorithm for training deep SNNs.

Simulation results on two benchmark datasets of MNIST and Fashion-MNIST confirm that the proposed DS4NN algorithm is applicable in deep architectures of SNNs and it can make quick decisions with high accuracy and by producing only a few spikes. DS4NN outperforms [11] and [14] and achieves an accuracy of 99.3% on the MNIST dataset and an accuracy of 91.6% on the Fashion-MNIST dataset.

5- Conclusion

In this work, we introduced DS4NN for direct training of deep spiking neural networks. This method is faster, more energy efficient, and computationally cheaper than rate-based SNN and ANN due to the sparseness and use of simple IF neuron models. In single-spike-based temporal SNNs, each IF neuron is allowed to fire only once, which helps the network make fast decisions and perform backward computations in a sparse manner. Also, most of the energy consumed in neuromorphic hardware is caused by spikes and calculations in neurons. Therefore, using a simple neuron model and sparsely performing calculations can greatly reduce the energy consumption of the hardware.

References

[1] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[2] B. Rueckauer, S.C. Liu, Conversion of analog to spiking neural networks using sparse temporal coding, in: 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018, pp. 1-5.

[3] N. Rathi, G. Srinivasan, P. Panda, K. Roy, Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation, arXiv preprint arXiv:2005.01807, (2020).

[4] A. Sengupta, Y. Ye, R. Wang, C. Liu, K. Roy, Going Deeper in Spiking Neural Networks: VGG and Residual Architectures, Frontiers in Neuroscience, 13 (2019).

[5] C. Lee, S.S. Sarwar, P. Panda, G. Srinivasan, K. Roy, Enabling Spike-Based Backpropagation for Training Deep Neural Network Architectures, Frontiers in Neuroscience, 14 (2020).

[6] S. Deng, S. Gu, Optimal conversion of conventional artificial neural networks to spiking neural networks, arXiv preprint arXiv:2103.00476, (2021).

[7] J. Allred, K. Roy, L4-Norm Weight Adjustments for Converted Spiking Neural Networks, arXiv preprint arXiv:2111.09446, (2021).

[8] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, K.C. Tan, A Tandem Learning Rule for Effective Training and Rapid Inference of Deep Spiking Neural Networks, IEEE Transactions on Neural Networks and Learning Systems, (2021) 1-15.

[9] J. Wu, C. Xu, X. Han, D. Zhou, M. Zhang, H. Li, K.C. Tan, Progressive Tandem Learning for Pattern Recognition With Deep Spiking Neural Networks, IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(11) (2022) 7824-7840.

[10] S.R. Kheradpisheh, M. Mirsadeghi, T. Masquelier, Spiking Neural Networks Trained via Proxy, IEEE Access, 10 (2022) 70769-70778.

[11] M. Zhang, J. Wang, J. Wu, A. Belatreche, B. Amornpaisannon, Z. Zhang, V.P.K. Miriyala, H. Qu, Y. Chua, T.E. Carlson, H. Li, Rectified Linear Postsynaptic Potential Function for Backpropagation in Deep Spiking Neural Networks, IEEE Transactions on Neural Networks and Learning Systems, 33(5) (2022) 1947-1958.

[12] S.R. Kheradpisheh, T. Masquelier, Temporal backpropagation for spiking neural networks with one spike per neuron, International Journal of Neural Systems, 30(06) (2020) 2050027.

[13] M. Mirsadeghi, M. Shalchian, S.R. Kheradpisheh, T. Masquelier, STiDi-BP: Spike time displacement based error backpropagation in multilayer spiking neural networks, Neurocomputing, 427 (2021) 131-140.

[14] M. Mirsadeghi, M. Shalchian, S.R. Kheradpisheh, T. Masquelier, Spike time displacement based error

backpropagation in convolutional spiking neural networks, arXiv preprint arXiv:2108.13621, (2021).

[15] S.R. Kheradpisheh, M. Ganjtabesh, S.J. Thorpe, T. Masquelier, STDP-based spiking deep convolutional neural networks for object recognition, Neural Networks, 99 (2018) 56-67.

[16] M. Mozafari, S.R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, M. Ganjtabesh, First-Spike-Based Visual Categorization Using Reward-Modulated STDP, IEEE Transactions on Neural Networks and Learning Systems, 29(12) (2018) 6178-6190.

[17] T. Masquelier, S.J. Thorpe, Unsupervised Learning of Visual Features through Spike Timing Dependent Plasticity, PLOS Computational Biology, 3(2) (2007) e31.

[18] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S.J. Thorpe, T. Masquelier, Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks, Pattern Recognition, 94 (2019) 87-95.

[19] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, T. Masquelier, SpykeTorch: Efficient Simulation of Convolutional Spiking Neural Networks With at Most One Spike per Neuron, Frontiers in Neuroscience, 13 (2019).

[20] S.R. Kheradpisheh, M. Ganjtabesh, T. Masquelier, Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition, Neurocomputing, 205 (2016) 382-392.

[21] J. Göltz, L. Kriener, A. Baumbach, S. Billaudelle, O. Breitwieser, B. Cramer, D. Dold, A.F. Kungl, W. Senn, J. Schemmel, Fast and energy-efficient neuromorphic deep learning with first-spike times, Nature machine intelligence, 3(9) (2021) 823-835.

[22] R. Vaila, J. Chiasson, V. Saxena, Feature extraction using spiking convolutional neural networks, in: Proceedings of the International Conference on Neuromorphic Systems, 2019, pp. 1-8.

[23] S.M. Bohte, J.N. Kok, H. La Poutré, Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing, 48(1) (2002) 17-37.

[24] H. Mostafa, Supervised Learning Based on Temporal Coding in Spiking Neural Networks, IEEE Transactions on Neural Networks and Learning Systems, 29(7) (2018) 3227-3235.

[25] I.M. Comsa, K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo, J. Alakuijala, Temporal Coding in Spiking Neural Networks with Alpha Synaptic Function, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 8529-8533.

[26] S.B.a.O. Shrestha, Garrick, SLAYER: Spike Layer Error Reassignment in Time, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2018.

[27] Y. Wu, L. Deng, G. Li, J. Zhu, L. Shi, Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks, Frontiers in Neuroscience, 12 (2018).

[28] D.a.S. Huh, Terrence J, Gradient Descent for Spiking Neural Networks, in: Garnett (Ed.) Advances in Neural Information Processing Systems, Curran Associates, Inc., 2018.

[29] L. Zuo, F. Xu, C. Zhang, T. Xiahou, Y. Liu, A multi-layer spiking neural network-based approach to bearing fault diagnosis, Reliability Engineering & System Safety, 225 (2022) 108561.

[30] Q. Meng, M. Xiao, S. Yan, Y. Wang, Z. Lin, Z.-Q. Luo, Training High-Performance Low-Latency Spiking Neural Networks by Differentiation on Spike Representation, in, 2022, pp. arXiv:2205.00459.

[31] B. Gardner, A. Grüning, Supervised Learning With First-to-Spike Decoding in Multilayer Spiking Neural Networks, Frontiers in Computational Neuroscience, 15 (2021).

[32] D.P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, in, 2014, pp. arXiv:1412.6980.

[33] E.O. Neftci, H. Mostafa, F. Zenke, Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks, IEEE Signal Processing Magazine, 36(6) (2019) 51-63.

[34] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, W. Maass, Long short-term memory and learning-to-learn in networks of spiking neurons, Advances in neural information processing systems, 31 (2018).

[35] F. Zenke, S. Ganguli, SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks, Neural Computation, 30(6) (2018) 1514-1541.

[36] J. Brownlee, A gentle introduction to object recognition with deep learning, Machine Learning Mastery, 5 (2019).

[37] H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, in, 2017, pp. arXiv:1708.07747.

[38] S.R. Kheradpisheh, M. Mirsadeghi, T. Masquelier, BS4NN: Binarized Spiking Neural Networks with Temporal Coding and Learning, Neural Processing Letters, 54(2) (2022) 1255-1273.

HOW TO CITE THIS ARTICLE

M. Mirsadeghi, M. Shalchian, S. R. Kheradpisheh, DS4NN: Direct training of deep spiking neural networks with single spike-based temporal coding, AUT J Electr Eng, 55(2) (2023) 179-190.

DOI: 10.22060/eej.2023.21991.5502