CDNA-SNN: A New Spiking Neural Network for Pattern Classification using Neuronal Assemblies

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems on December 1, 2023, DOI: 10.1109/TNNLS.2024.3353571. This is the author accepted manuscript (AAM) version of the article. The final published version may differ from this version.

CDNA-SNN: A New Spiking Neural Network for Pattern Classification using Neuronal Assemblies

Vahid Saranirad, Shirin Dora, Thomas Martin McGinnity, *Senior Member*, *IEEE*, and Damien Coyle, *Senior Member*, *IEEE*

***Abstract*— Spiking neural networks (SNNs) mimic their biological counterparts more closely than their predecessors and are considered the third generation of artificial neural networks. It has been proven that networks of spiking neurons have a higher computational capacity and lower power requirements than sigmoidal neural networks. This paper introduces a new type of spiking neural network that draws inspiration and incorporates concepts from neuronal assemblies in the human brain. The proposed network, termed as CDNA-SNN, assigns each neuron learnable values known as Class-Dependent Neuronal Activations (CDNAs) which indicate the neuron's average relative spiking activity in response to samples from different classes. A new learning algorithm that categorizes the neurons into different class assemblies based on their CDNAs is also presented. These neuronal assemblies are trained via a novel training method based on Spike-Timing Dependent Plasticity (STDP) to have high activity for their associated class and low firing rate for other classes. Also, using CDNAs, a new type of STDP that controls the amount of plasticity based on the assemblies of pre- and post-synaptic neurons is proposed. The performance of CDNA-SNN is evaluated on five datasets from the UCI machine learning repository, as well as MNIST and Fashion MNIST, using nested cross-validation for hyperparameter optimization. Our results show that CDNA-SNN significantly outperforms SWAT ($p<0.0005$) and SpikeProp ($p<0.05$) on 3/5 and SRESN ($p<0.05$) on 2/5 UCI datasets while using the significantly lower number of trainable parameters. Furthermore, compared to other supervised, fully connected SNNs, the proposed SNN reaches the best performance for Fashion MNIST and comparable performance for MNIST and N-MNIST, also utilizing much less (1-35%) parameters.**

***Index Terms*— class-dependent neuronal activation, CDNA, neuronal assembly, leaky integrate and fire, spiking neural network, spiking neurons.**

I. Introduction

Compared to traditional neural networks, Spiking Neural Networks (SNNs) more closely imitate the information processing mechanisms employed by biological neurons. SNNs comprise of neurons that transmit information using spikes which are discrete events at specific time instants. Consequently, SNNs consume lower power in hardware than traditional neural networks, which represent information using continuous real values [1]. It has been shown that an SNN has higher computational capacity in comparison to its predecessors [2]. However, the discontinuous nature of spikes makes a spiking neuron's output non-differentiable with respect to the weights. Therefore, popular gradient-based supervised training techniques for sigmoidal neural networks cannot be applied directly to SNNs.

To avoid computing gradients in SNNs, a number of studies have developed methods to convert trained sigmoidal neural networks into SNNs [3][4][5]. In these approaches, at first, a sigmoidal neural network is trained using the conventional backpropagation algorithm. Then the sigmoidal neurons in the trained network are replaced by spiking neurons in such a way that the responses of neurons in both networks are similar for the same inputs. The resulting SNNs usually yield lower classification accuracy compared to the trained sigmoidal neural networks. Several techniques are employed to minimize the loss of accuracy resulting from the conversion process, such as constraining the neurons' firing rate and thresholds [3] and synaptic weight normalization [6]. However, SNNs developed using conversion techniques are unable to match the performance of the network from which they were derived [3][4].

Many researchers have proposed error backpropagation based methods for SNNs that employ approximate derivatives where a discontinuity exists [7][8][9][10][11][12][13][14][15]. SpikeProp [8] employs a mean square error based on the spike timing of desired and actual output spike trains, assuming that postsynaptic potential is a linear function of time in a period close to the firing time. SpikeProp and its variations [16][17] cannot handle the silent neuron problem, which refers to neurons that stop spiking because of updates in their weights. It is not possible to compute an error for such neurons, as a result, they stop contributing to the network's output. SLAYER[9] utilises the spike response model, in which the neuronal membrane potential depends on current and past values of inputs. This approach defines a temporal error between the output and desired spike patterns and computes gradients using the probability of switching between spiking and non-spiking states of a neuron. SLAYER trains both weights and axonal delays of SNN simultaneously and addresses the silent neurons problem. The Macro-Micro backpropagation method [14] decomposes the error backpropagation into three components:1) macro-level that backpropagate error over firing rates, 2) micro-level that uses spike trains to transfer error to the

V. Saranirad, T. M. McGinnity and D. Coyle are with the Intelligent Systems Research Centre, School of Computing, Engineering and Intelligent Systems, Ulster University, Londonderry, BT487JL, U.K. (e-mail: saranirad-v@ulster.ac.uk; tm.mcginnity@ulster.ac.uk; dh.coyle@ulster.ac.uk).

S. Dora is with the Department of Computer Science, Loughborough University, Loughborough, LE113TU, U.K. (e-mail: s.dora@lboro.ac.uk).

D. Coyle is also with The Bath Institute for the Augmented Human, University of Bath, Bath, BA2 7AY, U.K. (e-mail: dhc30@bath.ac.uk).

previous layers, and 3) backpropagation of error using interactions between the first two components. SLAYER and Macro-Micro backpropagation can train multilayer architectures of fully connected and convolutional SNNs. However, the backward pass in spike-based backpropagation integrates the gradients across all time steps, which increases memory demands and computational complexity [18].

Inspired by biological neurons, a group of training methods for SNNs have used spike-based formulations of Hebb's rule [19][20][21]. According to the conventional Hebbian rule, "neurons that fire together, wire together" [19]. Different variants of this rule have been proposed but the most popular one for SNNs is spike-timing-dependent plasticity (STDP), which modifies the strength of synaptic connections based on the spike timing of pre- and post-synaptic neurons [22]. According to STDP, the weight of a synapse between two neurons is increased, when the presynaptic neuron generates a spike before a spike emitted by the postsynaptic neuron. On the contrary, the weight of a synapse is reduced, if the presynaptic neuron fires after the postsynaptic neuron. The strengthening and weakening of synaptic connections are termed as long-term potentiation (LTP) and long-term depression (LTD), respectively [23].

Although STDP is inherently unsupervised, it serves as the foundation for supervised algorithms such as ReSuMe [24], SWAT [25], and SEFRON [26]. These methods use a single layer SNN and are inapplicable to multilayer SNNs because STDP lacks a mechanism for propagating output errors across layers in the network. To deal with this issue, some methods trained multilayer SNNs employing STDP in a layer-wise manner [20][21]. After training a given layer, they freeze its synaptic weights, and the output of this layer is used to train the next layer. Lee et al. [21] trained hidden layers of a multilayer convolutional SNN using layer-wise unsupervised STDP. However, for the last fully connected layer, they employed a supervised variation of STDP. During the training process of the last layer, each sample is only used to train the neuron corresponding to the sample class. Thiele et al. [27][28] introduced a biologically implausible scheme for end-to-end training of a deep convolutional SNN using dual accumulator neurons and STDP. Each neuron in their network has two separate integrators with different thresholds; the integrator with the higher threshold adjusts the synaptic weight of the neuron with STDP, and the other integrator generates enough spikes (information) to train the subsequent layers. All the above STDP-based approaches train the network's layers independently and only use the class information for the last layer. Unsupervised learning in the hidden layers may result in lower classification performance.

Mozafari et al. [29] introduced a reward-modulated STDP to train a convolutional SNN. In this approach a reward/punishment signal is transmitted to neurons in the last hidden layer as a modulator [30] for STDP. A reward signal updates the weights with normal STDP, but the punishment modulator adjusts the weights using anti-STDP in which LTP and LTD are swapped and the weights in other layers of the network are trained using unsupervised STDP.

It is hypothesized that in the human brain an assembly or a group of neurons are associated to certain sensory information or cognitive data, such as a memory, a concept, or a phrase [31]. As one of the first studies at the level of neuronal groups, Hebb [19] indicated that if particular receptors are stimulated repeatedly, an area of cells will be associated with that stimulus, called an assembly of neurons. This assembly acts as a unified learning system or perception to create the simplest representation of an idea or an image [19]. Gerstein et al. [32] examined the characteristics of neuronal assemblies. They concluded that the assembly of neurons is formed by learning and strengthened by frequent use. Also, neuron assignments to different assemblies are based on the correlation of spiking between neurons, and neuronal assemblies corresponding to different subjects or memories overlap. Overlapping means that a neuron can be a part of multiple assemblies and contribute to each of their actions. However, existing SNN training methods do not utilize the idea of neuronal assemblies.

This paper introduces Class-Dependent Neuronal Activation based Spiking Neural Network (CDNA-SNN), which draws inspiration from neuronal assemblies. The learning algorithm for CDNA-SNN estimates how strongly a neuron fires for samples from a given class, using values referred to as Class-Dependent Neuronal Activation (CDNA). The proposed layer-wise learning method categorises the neurons into class assemblies based on their CDNAs. These assemblies enable the network to train the synaptic weights using supervised STDP for all layers. In addition, the proposed training algorithm can identify neurons with low relative spiking activity in response to the training data. These neurons, known as the hypoactive assembly, are deleted from the network after training.

The performance of CDNA-SNN has been evaluated using five numerical classification datasets from the UCI machine learning repository, as well as MNIST and Fashion MNIST image datasets. A statistical analysis has also been carried out, including a one-way ANOVA followed by a pairwise comparison employing Fisher's least significant difference (LSD) approach [33]. The results indicate that the proposed network can achieve higher or comparable performance using considerably fewer network parameters than other SNNs in comparison.

The remainder of the paper is organized as follows. The architecture and the learning algorithm for CDNA-SNN is presented in section II. Next, in Section III, performance evaluation of CDNA-SNN is described. The results of CDNA-SNN on different benchmark datasets is presented in Section IV. Section V discusses the results and highlights the advantages and drawbacks of the proposed training algorithm. Finally, Section VI presents conclusions.

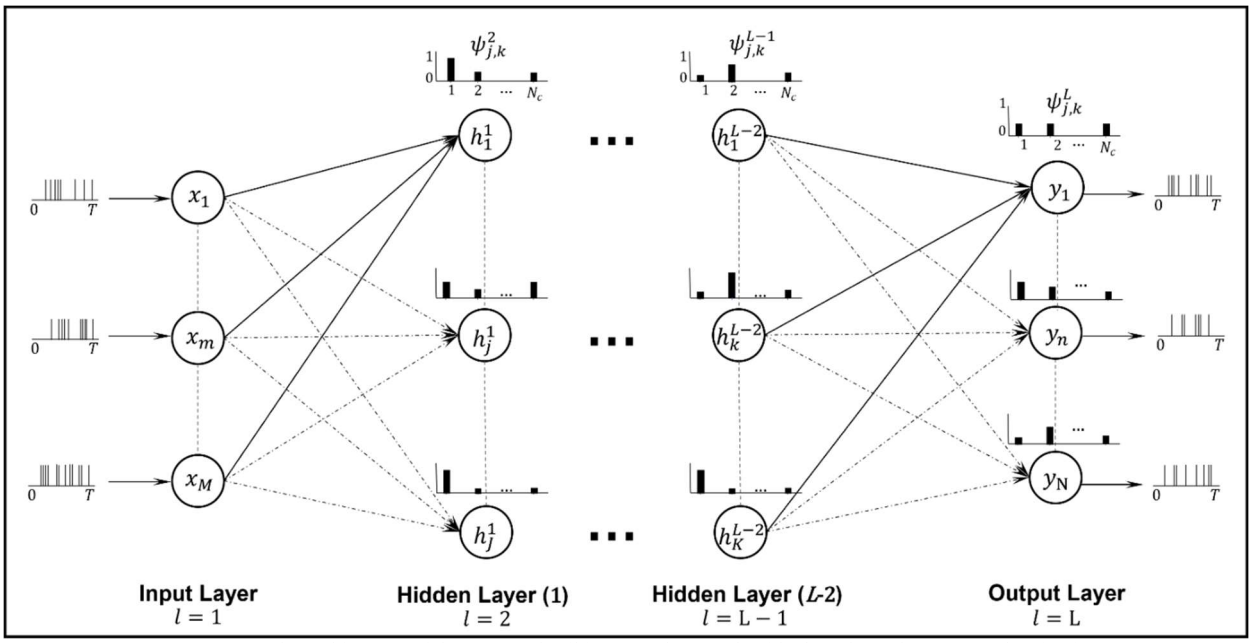

Figure 1. The architecture of the proposed spiking neural network with class-dependent neuronal activation

II. Spiking Neural Network with Class-Dependent Neuronal Activations

In this section, the architecture and learning algorithm for the Class-Dependent Neuronal Activation based Spiking Neural Network (CDNA-SNN) are presented.

A. Network Architecture

Figure 1 shows the architecture of CDNA-SNN with L layers. The first layer is the input layer which is used to present rate-coded spike patterns $X_n = [x_1, \dots, x_M]$ to the network. The true class label for X_n is denoted by $c_n \in [1, \dots, N_C]$ where N_C is the total number of classes. The input layer is followed by $L-2$ hidden layers and an output layer. All neurons in l^{th} layer of the network are fully connected to neurons in the $(l+1)^{th}$ layer, and there are no connections between the neurons from the same layer. The weight of the connection between the i^{th} neuron in the l^{th} layer and j^{th} neuron in the $(l+1)^{th}$ layer is denoted by w_{ij}^{l}. All spiking neurons in the network are modelled using the Leaky Integrate and Fire (LIF) neuron model. Based on the LIF neuron model, the membrane potential $(v_j^l(t))$ of the j^{th} neuron in the l^{th} layer at time t is determined using the differential equation given below [21]:

$$\tau \frac{dv_j^l(t)}{dt} = -v_j^l(t) + I_j^l(t), \quad \forall j, \forall l \in [2, \dots, L] \tag{1}$$

$$I_j^l(t) = \sum_{i,k} w_{ij}^l \delta\left(t - t_k^i\right) \tag{2}$$

where τ denotes the time constant of the membrane, $I_j^l(t)$ is the input current received by the j^{th} neuron in the l^{th} layer due to spikes generated by neurons in the $(l-1)^{th}$ layer. $\delta(t)$ denotes the Dirac delta function which has unit magnitude at $t = 0$ and zero elsewhere, and $\delta\left(t - t_k^i\right)$ is the k^{th} spike from neuron i. When $v_j^l(t)$ reaches a threshold V_{th}, the neuron generates a spike, and the membrane voltage is reset to zero for a refractory period τ_{ref}.

The class-dependent neuronal activation of a given neuron j in l^{th} layer for class k is denoted by $\psi_{j,k}^l$. CDNAs are real numbers in the range [0,1] that are indicative of the average firing rate of a neuron for spike patterns from a given class relative to its firing rate for other classes.

B. Learning Algorithm

In this section the learning algorithm for CDNA-SNN which is used to estimate the synaptic weights and CDNAs of all neurons in the network is presented. CDNA-SNN employs a layer-wise training approach i.e., while training the l^{th} layer in the network, the weights and CDNAs of neurons in previous layers are not changed.

Below, we first present the mechanism for estimating the CDNAs and then the training algorithm for CDNA-SNN is described.

1) Class-Dependent Neuronal Activations

Prior to training the l^{th} layer in the network, the CDNAs of all neurons in layer l with N_l neurons are initialised to equal values as

$$\psi_{j,k}^l = \frac{1}{N_c} \quad \forall j \in [1, \dots, N_l], k \in [1, \dots, N_c] \tag{3}$$

When an input spike pattern X_n with class label c_n is presented to CDNA-SNN, the most active neuron j^* in l^{th} layer is determined as:

$$j^* = \text{argmax}_j\left(r_j^l\right) \quad (4)$$

where r_j^l is the firing rate of the j^{th} neuron in the l^{th} layer of the network and is the total number of spikes that a neuron fires in response to the current input divided by the simulation duration. When more than one neuron spikes at the maximum rate, one of them is chosen randomly as j^*. The CDNAs for the neuron j^* are updated such that

$$\sum_{k=1}^{N_c} \psi_{j^*,k}^l = 1 \quad (5)$$

which implies that the sum of class-dependent neuronal activations for a given neuron is always equal to one. The change in CDNA ($\Delta\psi_{j^*,c_n}^l$) of neuron j^* corresponding to the sample class c_n with normalized firing rate $g_{j^*}^l$ is increased by:

$$\Delta\psi_{j^*,c_n}^l = \frac{1-\psi_{j^*,c_n}^l}{1+\exp\left(-g_{j^*}^l\right)} \quad (6)$$

where $g_{j^*}^l$ is the normalized firing rate of the neurons and is given by

$$g_{j^*}^l = \frac{r_{j^*}^l}{\sum_{j=1}^{N_l} r_j^l} \quad (7)$$

and the change in CDNAs for neuron j^* for other classes are as follows.

$$\Delta\psi_{j^*,k}^l = -\frac{1-\Delta\psi_{j^*,c_n}^l}{N_c-1} \quad \forall k|k \neq c_n \quad (8)$$

According to equation (6), the higher a neuron's normalised firing rate, the larger the amount of change in its CDNA. Updating CDNAs using equations (6) and (8) ensures that the sum of all CDNA for a given neuron are always equal to 1 (see Equation (5)). At the end of training, the CDNAs of a neuron represent its relative activity for each class. CDNAs separate the neurons in every layer and form assemblies of neurons for each class. Assembly $\Omega_{c_n}^l$ of neurons corresponding to class c_n in layer l can be defined as

$$\Omega_{c_n}^l = \forall j|\text{argmax}_k\left(\psi_{j,k}^l\right) = c_n \quad (9)$$

According to equation (9), neurons are associated to the class assembly for which they have the highest value of CDNA. However, some neurons might have equal CDNAs for all classes. These neurons are termed hypoactive because they exhibit lower relative firing rates for samples from different classes compared to other neurons in the same layer. The assembly of hypoactive neurons $\Omega_{\emptyset}^l$ in layer l can be formulated as

$$\Omega_{\emptyset}^l = \forall j|\,\psi_{j,k}^l = \frac{1}{N_c}, \forall k \in [1,\dots,N_c] \quad (10)$$

CDNAs estimated using Equations (6) and (8) enable the CDNA-SNN to perform supervised learning using STDP in all layers.

2) Synaptic Weights

Synaptic weights are initialized to random values in the interval [-1,1]. Without loss of generality, assume that the network has been trained using several spike patterns. At this stage neurons would be hypoactive or in one of the class assemblies depending on their CDNAs. Now assume the current spike pattern is X_n which is associated with actual class label c_n. Based on the spiking activity of neurons in response to X_n, three neurons in the l^{th} layer of the network can be determined:

- j^* : The most active neuron
- j' : The most active neuron in the assembly of c_n
- j'' : The most active neuron in the hypoactive assembly

Three possible scenarios can arise based on c_n and CDNAs of j^*. j^* might be associated with the assembly of 1) desired (actual) class, 2) undesired class, or 3) hypoactive neurons. Below, the learning rules for each of the scenarios are presented.

Desired Class: In this scenario the most active neuron j^* has the highest CDNA for the class c_n. Since this neuron which already belongs to the assembly $\Omega_{c_n}^l$ has a relatively high spiking activity for a spike pattern from the class c_n, this is a desired scenario. Therefore, STDP updates this neuron's weights to increase its activity for class c_n. The criterion for weight updating is given by

If $r_{j^*}^l < \alpha^l$ **Then** update the weights of j^*

This thresholding puts a limit on the maximum firing rate of neurons and prevents them from dominating the training process.

Undesired Class: In this scenario j^* has highest CDNA for a class other than c_n. As j^* was the most active neuron for a class incompatible with its maximum CDNA, anti-STDP is used to adjust its synaptic weights such that it fires weakly for samples from class c_n. Anti-STDP is the inverse of STDP, i.e., the weight of a synapse between two neurons is reduced when the presynaptic neuron fires before the postsynaptic neuron and vice versa. Additionally, it is essential to enhance the spiking activity of neurons in $\Omega_{c_n}^l$ in response to input patterns from c_n. Thereby, the most active neuron from $\Omega_{c_n}^l$ (j') gets an STDP update with the following criteria.

If $r_{j'}^l \geq \beta^l$ **Then** update the weights of j'

β^l prevents the weights of neurons with low firing rate from being updated, and therefore prevents loss of previously stored information in the network. If there is no neuron in $\Omega_{c_n}^l$, then the weights of the neuron with highest firing rate in the hypoactive assembly (j'') are updated using STDP. After several epochs of training, this neuron will start firing strongly for spike patterns from class c_n, improving its chance to join $\Omega_{k_0}^l$ during the rest of the training process.

Algorithm 1: Training process of l^{th} layer in the network

Input Dimension: M
Number of Neurons: N_l
Number of Classes: N_c
Initialise CDNAs: $\psi_{j,k}^{l} = \frac{1}{N_c} \quad \forall j \in [1, \dots, N_l], k \in [1, \dots, N_c]$
Initialise Synaptic Weights: $w_{M \times N_l} \leftarrow Normal_{Random(\mu,\sigma)}$
FOR numbers of epochs:
  FOR every *SpikePattern* X_n in *TrainingData with ClassLabel* c_n:
    Present X_n *to the SNN*
    Firing rate of j^{th} *neuron:* r_j^l
    Find the most active neuron: $j^* = \text{argmax}(r_j^l)$
    Normalised firing rate of neuron j^**:* $g_{j^*}^l = \frac{r_{j^*}^l}{\sum_{j=1}^{N_l} r_j^l}$
    Increase the CDNA of j^* *for the sample class:*

$$\Delta\psi_{j^*,c_n}^l = \frac{1-\psi_{j^*,c_n}^l}{1+exp(-g_{j^*})}$$

    Decrease the CDNAs of j^* *for other classes:*

$$\Delta\psi_{j^*,k}^l = -\frac{1-\Delta\psi_{j^*,c_n}^l}{N_c-1} \quad \forall k | k \neq c_n$$

    IF ($j^* \epsilon \Omega_{\emptyset}^l$) **or** ($j^* \epsilon \Omega_{c_n}^l$)**:** // Hypoactive or Desired Class
      IF ($r_{j^*} < \alpha^l$)**:**
        Update the weights of j^* *by STDP*
      ENDIF
    ELSEIF ($j^* \epsilon \Omega_k^l \ \forall k | k \neq c_n$)**:** // Undesired Class
      Update the weights of j^* *by ANTI-STDP*
      IF ($\Omega_{c_n}^l \neq \emptyset$)**:** // If there is at least one neuron in $\Omega_{c_n}^l$
        Find the most active neuron from $\Omega_{c_n}^l$*:* j'
        IF ($r_{j'}^l \geq \beta^l$)**:**
          Update the weights of j' *by STDP*
      ELSEIF ($\Omega_{c_n}^l = \emptyset$)**:** // There is no neuron in $\Omega_{c_n}^l$
        Find the hypoactive neuron with highest firing rate: j''
        Update the weights of j'' *by STDP*
      ENDIF
    ENDIF
  ENDFOR
ENDFOR
Remove all hypoactive neurons

Algorithm 2: Layer-wise Training Algorithm for CDNA-SNN

Number of Layers: L
Number of neurons in each layer: $N_l \ \forall l \in [2, \dots, L]$
Thresholds for the neurons in each layer: $\alpha^l, \beta^l \ \forall l \in [2, \dots, L]$
FOR layer l in the interval 2 to L:
  Initialize layer l with N_l neurons and parameters α^l, β^l
  IF ($l == 2$)**:** //First Hidden Layer
    Train the layer l *using Algorithm 1 and Conventional STDP*
  ELSE: //Subsequent Layers
    Train the layer l *using Algorithm 1 and CDNA-Weighted STDP*
  ENDIF
  $N_l = N_l - number\ of\ hypoactive\ neurons$
  Freeze all the weights and CDNAs in the layer l
ENDFOR

Hypoactive Neurons: In this scenario j^* has CDNAs equal to the initial values, and therefore it is in the hypoactive assembly. Since this neuron has the highest firing rate for c_n, and it is not associated with any of the classes, STDP updates its weights to increase its activity for class c_n provided that its firing rate is less than α^l.

The proposed algorithm for estimating CDNAs and training the synaptic weights of each layer of CDNA-SNN is summarized in Algorithm 1. After training every layer, all the hypoactive neurons are removed, and the weights and CDNAs for the neurons in this layer are frozen.

The new STDP-based learning rule introduced in this paper exploits the CDNAs of neurons in the $(l-1)^{th}$ layer, while learning synaptic weights of the neurons in the l^{th} layer of the network. Since the input layer neurons do not have CDNAs, the learning algorithm uses conventional STDP for learning the synaptic weights of the connections between the input layer and the first hidden layer. The following will explain each of these learning rules.

Conventional STDP: The change in the synaptic weight $(\Delta w_{i,j^*})$ of the connection between i^{th} input neuron and the neuron j^* are estimated using an exponential variant of STDP as

$$\Delta w_{ij^*} = \begin{cases} A.e^{\frac{-|t_i - t_{j^*}|}{\tau}} & t_{j^*} \geq t_i \\ B.e^{\frac{-|t_i - t_{j^*}|}{\tau}} & t_{j^*} < t_i \end{cases} \tag{11}$$

where τ is the time constant for STDP, and spike times of the pre- and post-synaptic neurons are represented by the variables t_i and t_{j^*}, respectively. A and B denote whether the learning rule is STDP or Anti-STDP as below

$$A = \begin{cases} +1 & \text{STDP} \\ -1 & \text{Anti} - \text{STDP} \end{cases}, \quad B = -A \tag{12}$$

CDNA-Weighted STDP: Conventional STDP only depends on the spiking activity of pre and post synaptic neurons. CDNA-Weighted STDP, as implied by its name, controls the amount of plasticity with respect to the CDNAs of the pre-synaptic neuron for the sample class c_n.

$$\Delta w_{ij^*} = \begin{cases} A.\Psi.e^{\frac{-|t_i - t_{j^*}|}{\tau}} & t_{j^*} \geq t_i \\ B.\Psi.e^{\frac{-|t_i - t_{j^*}|}{\tau}} & t_{j^*} < t_i \end{cases} \tag{13}$$

where Ψ is the normalized CDNA of pre-synaptic neuron for the sample class c_n as below

$$\Psi = \frac{\psi_{i,c_n}^{l-1}}{\sum_{j=1}^{N_{l-1}} \psi_{j,c_n}^{l-1}} \tag{14}$$

In CDNA-Weighted STDP the weights of connections from all presynaptic neurons to a postsynaptic neuron are updated

based on the CDNAs of the corresponding presynaptic neurons for the class of the current spike pattern. CDNA-Weighted STDP aims to control the plasticity during the learning in a way that presynaptic neurons can build strong connections with the post synaptic neurons from the same assembly using STDP. Also, this learning rule weakens the connections between pre-synaptic neurons in $\Omega_{c_n}^{l-1}$and post-synaptic neurons from other assemblies that have high firing rate for c_n by anti-STDP.

The overall layer-wise algorithm for training CDNA-SNN is summarized in Algorithm 2. According to Algorithm 1, hypoactive neurons might join one of the class assemblies throughout training; Otherwise, they will be removed after the training of each layer. Thus, at the end of training process, each neuron in the network would be associated with a specific class assembly. Given an input spike pattern X_n with class label c_n. The predicted class $\hat{c}_n$ for this input is determined based on the highest CDNA for the most active neuron (j^*) in the output layer (L), given by

$$\hat{c}_n = \text{argmax}_k\left(\psi_{j^*,k}^{L}\right) \tag{15}$$

III. Performance Evaluation

This section defines the performance metrics used for implementation, explains hyper-parameter optimization, and finally describes datasets and experimental settings for each dataset. CDNA-SNN was implemented in Python 3.7 using NVIDIA Tesla v100 GPU and 64 GB memory. In all the simulations for CDNA-SNN, the LIF model is used, and a rate encoder converted real-valued data into spikes.

In the next subsection, we first present the approach for hyperparameter optimization, and the metrics used for the evaluation. Then, datasets and experimental settings are explained.

A. Performance Metrics

Performance evaluation is conducted using testing classification accuracy and number of trainable parameters. classification accuracy (η) is computed as the number of correctly classified samples, given as

$$\eta = \frac{\text{Number of correctly classified samples}}{\text{Total number of samples}} \tag{16}$$

One-way ANOVA [34] has been utilized to evaluate if testing classification accuracies obtained from various learning algorithms are significantly different. When statistical significance was observed using ANOVA, pairwise comparison is done using Fisher's Least Significant Difference (LSD) method [35].

The number of trainable parameters for two-layer and three-layer fully-connected networks is equal to ($N_i \times N_o$) and ($N_i \times N_h + N_h \times N_o$), respectively. CDNA-SNN estimates CDNAs while training the synaptic weights. Accordingly the number of trainable parameters for CDNA-SNN is equal to ($N_i \times N_h + N_h \times N_o$)+($N_h + N_o$)$\times N_c$.

B. Hyper-parameter Optimization

The proposed training algorithm includes parameters α^l and β^l for hidden layers and output layer. In this paper, hyperparameter optimisation is done by Nested Cross-Validation (N-CV). N-CV nests cross-validation and optimization of hyperparameters. Without N-CV, the same data might be employed to optimize parameters and evaluate the network performance. This may lead to a biased model evaluation and inaccurate estimation of errors in training or testing the network due to data leakage between training and testing data [36].

C. Datasets and Experimental Settings

UCI Benchmark Datasets: Five benchmark classification datasets from the UCI machine learning repository [37] are used to evaluate the classification performance of the CDNA-SNN with 2-layers and 3-layers. All benchmark datasets are numerical datasets that consist of Iris, a three-class problem and four binary classification problems, namely Pima Indians diabetes, breast cancer Wisconsin (Original), Liver disorders and Ionosphere. Datasets are randomly divided into N-CV folds that contain a balanced number of samples in each class. Table I shows the number of features and classes of all five datasets and the number of folds in the inner and outer loops of N-CV.

Table II describes the parameters of CDNA-SNN used for UCI Datasets. parameters α^l and β^l for hidden layer and output layer were optimized by N-CV given the ranges in the table. Initial number of neurons decreased after removing the hypoactive neurons from each layer. The simulation parameters of SpikeProp, SRESN, and SWAT have been set as in [38].

MNIST: CDNA-SNN is also evaluated on the MNIST dataset, which contains 28*28 grayscale images of handwritten digits 0-9 with 60000 training and 10000 testing samples. Table III describes the parameters of CDNA-SNN used for MNIST. Other simulation parameters are as in Table II.

Fashion MNIST: Compared to the original MNIST, Fashion-MNIST is a more complex dataset consisting of 60,000 training and 10,000 testing samples. It includes 28x28 grayscale images from the following ten categories: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot. Fashion-MNIST is intended to be a drop-in replacement for the original MNIST dataset for evaluating machine learning methods. The simulation parameters for Fashion MNIST are the same as MNIST.

N-MNIST: The Neuromorphic-MNIST dataset [39] is a spiking version of the original MNIST. It consists of the same 60000 training and 10000 testing samples of MNIST converted into neuromorphic data using a dynamic vision sensor (DVS). This DVS was mounted on a motorized pan-tilt unit to capture MNIST samples played on a monitor. DVS generates a spike whenever a brightness change is detected, on or off. N-MNIST consists of spatial-temporal patterns with a dimension of 34x34x2 recorded for 300ms with 1us resolution. In this paper, the time step of N-MNIST is reduced to 1ms. Because of the saccadic motion, this dataset is more challenging than MNIST.

IV. RESULTS

In this section, the results of CDNA-SNN on different benchmark datasets are presented and compared with the state-of-the-art SNNs. Also, a frequency analysis is presented to examine the firing rate of assemblies in response to the samples from different classes. Finally, CDNA-weighted STDP is compared with the conventional STDP.

A. UCI Benchmark Datasets

Table IV compares the results of applying the two and three-layer CDNA-SNN to other learning algorithms for SNNs, which include SWAT [25], SpikeProp [8], and SRESN [40]. The simulation parameters of SpikeProp, SRESN, and SWAT have been set as in [38]. Results are reported for both two and three-layer CDNA SNN, and the comparison is based on network architecture, the number of trainable parameters and testing accuracy. Accuracies are reported in the form of mean (standard deviation) across outer folds of nCV. The network architecture format is N_i-N_h-N_o, denoting the number of input, hidden, and output neurons. For CDNA-SNN and SRESN the number of trainable parameters is equal to $N_i \times N_h + N_h \times N_o$. SRESN is an evolving network that has a different number of neurons for each outer fold. Also, since CDNA-SNN removes the hypoactive neurons after training, it does not have equal neurons for all folds. Accordingly, for these two algorithms the number of neurons and number of trainable parameters are reported as a range. SpikeProp is a three-layer SNN with 16 synapses with different delays between every pre- and post-synaptic neuron. Accordingly, the number of weights for SpikeProp is equal to $16 \times (N_i \times N_h + N_h \times N_o)$. CDNA-SNN uses rate encoding that makes its number of inputs equal to the number of data features plus a bias neuron. Other SNNs in this comparison encode the data using population coding and consequently have more input neurons. SWAT's hidden layer serves as a frequency filter for feature extraction from input patterns. Accordingly, this method has $N_h \times N_o$ trainable parameters.

According to Table IV, for IRIS dataset, which is the three-class classification problem, three-layer CDNA-SNN achieves a testing accuracy that is 0.09-3.96% higher than others. ANOVA results revealed that at least one method performs significantly different from others ($p < 0.0001$). The LSD pairwise statistical analysis confirmed that three-layer CDNA-SNN significantly outperforms SpikeProp ($p < 0.05$) and SWAT ($p < 0.00001$). Even two-layer CDNA-SNN with considerably less trainable parameters has significantly better performance than SpikeProp ($p < 0.05$) and SWAT ($p < 0.00001$).

For a simple binary classification dataset like breast cancer, three-layer CDNA-SNN has the best performance among all approaches. A one-way ANOVA proves that not all SNNs have equal performance ($p < 0.005$), and post-hoc pairwise comparison indicated that three-layer CDNA-SNN significantly only outperformed SWAT for the breast cancer dataset ($p < 0.0005$).

For Pima diabetes, a relatively complex classification task, SpikeProp has the best results in terms of accuracy. Although three-layer CDNA-SNN has 0.46% less accuracy than SpikeProp, it uses a considerably more compact network with almost 97% less network parameters. ANOVA shows a significant difference between the compared SNNs ($p < 0.000001$); however, LSD pairwise comparison revealed that SpikeProp is not significantly better than the two CDNA-SNNs represented in Table IV ($p > 0.2$).

Regarding the most challenging dataset, liver disorders, ANOVA results again indicate significant differences among algorithms ($p < 0.000005$). Also, LSD proves that the two CDNA-SNN architectures in the table significantly outperform SRESN ($p < 0.0005$), SpikeProp ($p < 0.05$) and SWAT ($p < 0.0005$).

TABLE I
DESCRIPTION OF FIVE UCI BENCHMARK DATASETS AND THEIR N-CV SETUP

Data Set	# Classes	# Samples	# Features	# Outer Loop Folds	# Inner Loop Folds
Iris	3	150	4	5	5
Breast Cancer	2	683	9	10	10
Pima diabetes	2	768	9	10	10
Liver disorders	2	345	6	5	5
Ionosphere	2	351	34	5	5

TABLE II
PARAMETERS OF CDNA-SNN USED FOR UCI DATASETS

Parameter Description	Value/Range
$\boldsymbol{\alpha^l}$	[5,10,15,20,25] Hz
$\boldsymbol{\beta^l}$	[0,5,10,15] Hz
Initial Number of Neurons in the Hidden Layer	[30,50,70,90,110]
Initial Number of Neurons in the Output Layer	[20,40,60,80,100]
Time Constant of STDP ($\boldsymbol{\tau}$).	50 ms
Learning Rate for CDNAs	1e-3
Learning Rate for Synaptic Weights	1e-2
Simulation Time	300 ms
Simulation Time Step	1 ms
Encoder's Frequency Range	20-280 Hz
Batch Size	5

TABLE III
PARAMETERS OF CDNA-SNN USED FOR MNIST, FASHION MNIST AND N-MNIST

Parameter Description	Value/Range
$\boldsymbol{\alpha^l}$	[40,50,60,70,80] Hz
$\boldsymbol{\beta^l}$	[30,40,50,60,70] Hz
Initial Number of Neurons in the Hidden Layer	[150,200,250,300,350]
Initial Number of Neurons in the Output Layer	[100,150,200,250,300]
Batch Size	100

TABLE IV
COMPARISON OF CDNA-SNN, SRESN, SPIKEPROP, AND SWAT ON FIVE UCI BENCHMARK CLASSIFICATION DATASETS

Data Set	Algorithm	Network Architecture	# Trainable Parameters	Testing Accuracy (%)
Iris	2L-CDNA-SNN	5-(5-8)	40-64	97.75(0.92)
	3L-CDNA-SNN	**5-(5-8)-(7-10)**	**96-174**	**97.84(0.33)**
	SRESN	24-(5-11)	120-264	97.01(0.73)
	SpikeProp	25-10-3	4480	96.13(0.83)
	SWAT	24-312-3	936	93.88(1.80)
Breast Cancer	2L-CDNA-SNN	10-(6-9)	(72-108)	97.35(1.66)
	3L-CDNA-SNN	**10-(6-9)-(5-7)**	**112-185**	**97.81(1.76)**
	SRESN	54-(9-13)	486-702	97.10(0.20)
	SpikeProp	55-15-2	13680	97.04 (0.53)
	SWAT	54-702-2	1404	95.66 (0.08)
Pima diabetes	2L-CDNA-SNN	10-(16-19)	192-228	76.57(1.17)
	3L-CDNA-SNN	10-(16-19)-10	372-438	76.92(2.48)
	SRESN	54-(10-13)	540-702	70.06 (1.82)
	SpikeProp	**55-20-2**	**16640**	**77.38(1.03)**
	SWAT	54-702-2	1404	72.11(1.38)
Liver disorders	2L-CDNA-SNN	7-(18-20)	162-180	70.33(4.65)
	3L-CDNA-SNN	**7-(18-20)-(9-10)**	**342-400**	**75.07(2.95)**
	SRESN	36-(7-10)	252-360	60.17(1.78)
	SpikeProp	37-15-2	9360	64.23(3.92)
	SWAT	36-468-2	936	60.43(2.72)
Ionosphere	2L-CDNA-SNN	35-(17-19)	629-703	89.78(1.26)
	3L-CDNA-SNN	**35-(17-19)-(8-12)**	**781-955**	**90.64(1.34)**
	SRESN	204-(15-21)	3060-4284	88.52(1.07)
	SpikeProp	205-25-2	82800	86.89(2.00)
	SWAT	204-2652-2	5304	90.04(1.87)

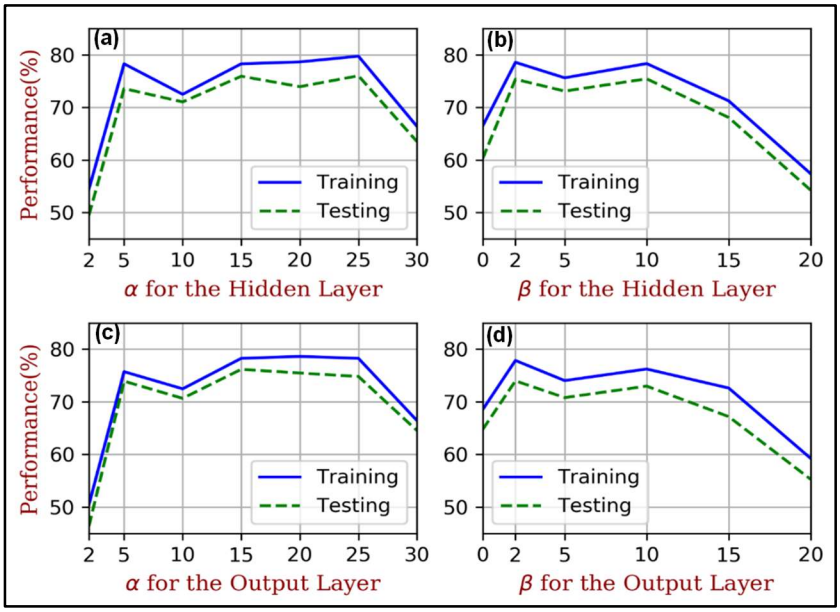

Figure 2. Variation in training and testing accuracy of a three-layer CDNA-SNN against variation in (a) $\alpha^{l=2}$, (b) $\beta^{l=2}$, (c) $\alpha^{l=3}$, and (d) $\beta^{l=3}$ for the Liver disorders problem. In each case the other three parameters are fixed

For Ionosphere, which has the highest number of features among all five datasets, insignificant difference between the two-layer CDNA-SNN and SWAT were observed $(0.26\%, p = 0.79)$, although SWAT uses almost six times the number of CDNA-SNN's parameters. Additionally, adding another layer to the CDNA-SNN improves the network performance and the three-layer CDNA-SNN has the highest accuracy. One-way ANOVA indicated that at least one approach is significantly different from others ($p < 0.01$), and the Three-layer CDNA-SNN significantly outperforms SRESN ($p < 0.05$) and SpikeProp ($p < 0.005$).

Figure 2 shows an example of the influence of α^l and β^l

on the overall training and testing accuracy of a three-layer CDNA-SNN for the Liver disorders problem. The upper row (Figure 2 (a-b)) presents the effect of parameters in the hidden layer, and Figure 2 (c-d) is allocated to the parameters in the output layer. Only one parameter varies in each case, and the other three are fixed. As can be seen, the variations are not linear, and a hyperparameter optimization algorithm is necessary.

B. MNIST

Table V compares CDNA-SNN with state-of-the-art SNNs. All compared networks are feedforward, fully connected, multi-layer SNNs with supervised training algorithms. The method proposed by Jin et al. [14] has slightly higher accuracy than CDNA-SNN, while CDNA-SNN employs almost 93% fewer trainable parameters.

C. Fashion MNIST

Table VI compares the results of CDNA-SNN with state-of-the-art feedforward, fully connected, multi-layer SNNs on the Fashion MNIST dataset. CDNA-SNN outperformed other feedforward SNNs by 3-6% in classification performance while employing 72-99 % fewer trainable parameters.

TABLE V
COMPARISON OF CDNA-SNN WITH SOME STATE-OF-THE-ART FEEDFORWARD, FULLY CONNECTED, MULTI-LAYER SNNS ON MNIST DATASET

Paper	Learning Method	Network Architecture	#Trainable Parameters	Testing Accuracy (%)	#Epochs	#Time Steps
Diehl et al., 2015[3]	Backpropagation, ANN to SNN conversion	28x28-1200-1200-10	2,392,800	98.68	50	500
O'Connor and Welling, 2016[10]	Fractional Stochastic Gradient descent	28x28-300-300-10	328,200	97.80	50	10
Wu et al., 2018[13]	Spatio-temporal backpropagation	28x28-800-10	635,200	98.89	200	30
Jin et al., 2018[14]	**Hybrid macro/micro level backpropagation**	**28x28-800-10**	**635,200**	**98.93**	**100**	**400**
Tavanaei and Maida 2019[15]	STDP-based backpropagation	28x28-500-150-10	468,500	97.20	1	9
Zhang et al. 2018[41]	Equilibrium learning +STDP + backpropagation	28x28-4500-10	3,573,000	98.52	100	-
Hao et al. 2020[42]	Unsupervised and supervised STDP	28x28-10000-10	7,850,000	96.73	20	1000
Zhao et al. 2020 [43]	Global Feedback + STDP	28x28-800-800-800-10	1,915,200	98.62	100	10
Kheradpisheh et al., 2020 [44]	Backpropagation with temporal encoding	28x28-400-10	317,600	97.4	-	256
Zhang et al., 2020 [45]	Threshold-driven Plasticity Algorithm	28x28-800-10	635,200	96.8	-	-
Comsa et al. 2021 [46]	Backpropagation with temporal encoding	28x28-340-10	269,960	97.96	1000	-
Kheradpisheh et al., 2022 [47]	Backpropagation with temporal encoding + Binary synaptic weights	28x28-600-10	476,400	97.0	500	256
Zhang et al., 2022 [48]	Spike-Timing-Dependent Backpropagation	28x28-800-10	635,200	98.5	150	-
CDNA-SNN	Supervised STDP + CDNA-Weighted STDP	28x28-55-38	45,396	98.91	100	300

TABLE VI
COMPARISON OF CDNA-SNN WITH SOME STATE-OF-THE-ART FEEDFORWARD, FULLY CONNECTED, MULTI-LAYER SNNS ON FASHION MNIST DATASET

Paper	Learning Method	Network Architecture	#Trainable Parameters	Testing Accuracy (%)	#Epochs	#Time Steps
Zhang et al., 2020 [49]	Temporal Spike Sequence Learning Backpropagation	28x28-400-400-10	477,600	89.80	100	5
Hao et al., 2020 [42]	Unsupervised and supervised STDP	28x28-6400-10	5,081,600	85.47	10	1000
Zhao et al., 2020 [43]	Global Feedback + STDP	28x28-200-200-200-200-200-10	318,800	89.05	200	10
Perez-Nieves and Goodman, 2021 [50]	Backpropagation using Surrogate Gradient	28x28-200-10	158,800	82.2	100	100
Perez-Nieves and Goodman, 2021 [50]	Backpropagation using Surrogate Gradient	28x28-300-300-300-300-300-10	598,200	82.7	100	100
Kheradpisheh et al., 2022 [47]	Backpropagation with temporal encoding + Binary synaptic weights	28x28-1000-10	794,000	87.3	500	256
Zhang et al., 2022 [48]	Spike-Timing-Dependent Backpropagation	28x28-1000-10	794,000	88.1	150	-
CDNA-SNN	**Supervised STDP + CDNA-Weighted STDP**	**28x28-53-46**	**44,188**	**90.12**	**100**	**300**

TABLE VII
COMPARISON OF CDNA-SNN WITH SOME STATE-OF-THE-ART FEEDFORWARD, FULLY CONNECTED, MULTI-LAYER SNNS ON N-MNIST DATASET

Paper	Learning Method	Network Architecture	#Trainable Parameters	Testing Accuracy (%)	#Epochs	#Time Steps
Shrestha and Orchard, 2018[9]	**Backpropagation**	**34x34x2-500-500-10**	**1,411,000**	**98.95**	**100**	**300**
Lee et al., 2016 [51]	Backpropagation	34x34x2-800-10	1,857,600	98.74	200	300
Cohen et al., 2016 [52]	Synaptic Kernel Inverse Method	34x34x2-10000-10	23,220,000	92.87	-	360
Wu et al., 2018[13]	Spatio-temporal backpropagation	34x34x2-800-10	1,857,600	98.78	200	30
Jin et al. 2018 [53]	Hybrid macro/micro level backpropagation	34x34x2-800-10	1,857,600	98.88	60	500
Perez-Nieves and Goodman, 2021 [50]	Backpropagation using Surrogate Gradient	34x34x2-200-10	464,400	92.7	100	300
CDNA-SNN	Supervised STDP + CDNA-Weighted STDP	34x34x2-69-48	163,074	98.43	100	300

D. Neuromorphic MNIST

In Table VII CDNA-SNN is compared to other fully connected multi-layer SNNs with supervised training algorithms. CDNA-SNN achieves a testing accuracy of 98.43 which is closer to the performance of the best-performing method i.e. SLAYER [9], which achieves an accuracy of 98.95%. It may be noted that CDNA-SNN requires 88% fewer parameters compared to SLAYER for this performance.

E. Frequency Analysis

Here we evaluate the firing rate of neurons in each assembly after training in response to the testing patterns from different classes. Figure 3 represents the mean firing rate of assemblies of a 3-layer CDNA-SNN in response to testing spike patterns from different classes for MNIST and Fashion MNIST. The desired scenario for these assemblies is to have relatively higher activity for their class and a lower firing rate for the spike patterns from other classes. As can be seen, assemblies in the hidden layer (a,c) could not properly fulfil this objective. However, assemblies in the output layer (b,d) have a relatively higher mean firing rate and a considerably lower average activity for other classes. Two-layer CDNA-SNN (input-hidden) could achieve 94.89% and 90.12% accuracy for MNIST and Fashion MNIST, respectively. However, three-layer CDNA-SNN improved the performance of two-layer CDNA-SNN to 98.91% for MNIST and 94.33% for Fashion MNIST. It should be noted that CDNA-SNN relies on the most active neuron in each assembly but having assemblies with relatively higher average firing rates for the spike patterns from their class will lead to better classification performance.

F. CDNA-weighted STDP versus STDP

In this section CDNA-weighted STDP is compared with conventional STDP regarding classification performance and the synaptic connections between assemblies in the hidden layer and assemblies in the output layer. Figure 4 compares the average synaptic connections between assemblies in the hidden layer and output layer of CDNA-SNN, which is trained using algorithm 2 and different STDP rules in the output layer, including (a) Conventional STDP, and (b) CDNA-Weighted STDP for MNIST and Fashion MNIST datasets. As shown, CDNA-Weighted STDP could build stronger connections between neurons from the same class than conventional STDP, while separating different assemblies by weak or negative connections. The reason for this result is that CDNA-Weighted STDP adjusts the weights connected to all presynaptic neurons with respect to their CDNAs of the sample class. In this example, all networks are initialised with 200 hidden and 100 output neurons. classification accuracies of conventional STDP are 95.26% and 86.94% for MNIST and Fashion MNIST, respectively. CDNA-Weighted STDP achieved 98.91% and 90.12% accuracy for MNIST and Fashion MNIST. Employing CDNA-SNN led to 38 and 46 output neurons for MNIST and Fashion MNIST, respectively. In comparison, training with conventional STDP resulted in 45 and 58 output neurons for MNIST and Fashion MNIST.

G. Number of Layers of CDNA-SNN

It is possible to use the proposed algorithm for networks with any number of layers. However, the results have been reported for at most 3-layer CDNA-SNN with one hidden layer because adding more layers did not improve the performance of CDNA-SNN for the datasets used for evaluation. Figure 5 shows the testing accuracy of CDNA-SNN with different numbers of layers for MNIST and Fashion MNIST. It can be observed that the network's performance does not improve when more than three layers are used.

IV. DISCUSSION

The proposed CDNA-SNN is a new type of SNN that draws inspirations from assemblies of biological neurons. The training algorithm for CDNA-SNN employs learnable values known as CDNAs to form assemblies. Like biological assemblies, these CDNA-based assemblies vary throughout training, share some neurons with other assemblies, and are associated with a specific piece of information (class). The assemblies formed by CDNA have a relatively higher average firing rate for training and testing patterns from their associated class and a lower rate for other classes (Figure 3).

The training algorithm of CDNA-SNN concentrates on the most active neurons in response to the spike patterns from each class and estimates the average relative firing rate of

neurons for all classes using CDNAs. CDNAs are updated throughout training such that neurons that are relatively more active for a specific class are added to the relevant assembly, and hypoactive neurons are identified. CDNA-SNN divides neurons into class assemblies, trains each assembly to be more active for its class, and removes untrained hypoactive neurons, which has led to excellent performance with a minimal number of neurons. Regarding the five UCI datasets, CDNA-SNN achieved the best performance among the compared methods on Iris, breast cancer, liver disorders, and Ionosphere datasets and reached the close second-best performance for Pima diabetes (TABLE IV). For all of these datasets, three-layer CDNA-SNN used the least number of parameters, which was considerably less than SpikeProp (81-99% less) and SWAT (57-86% less). Considering MNIST, CDNA-SNN has reached the near second-best performance, with 83-99% fewer parameters (TABLE V). For Fashion MNIST, CDNA-SNN has achieved 0.32-4.65% better accuracy compared to all feedforward fully connected SNNs, while requiring 72-99% fewer parameters (TABLE VI). Finally for N-MNIST, CDNA-SNN has achieved close to the best performance using 88% fewer network parameters (TABLE VII).

The main advantage of CDNA-SNN over the other methods in comparisons is using fewer parameters while achieving higher or comparable classification performance, as indicated by the results. CDNA-SNN utilises a small number of neurons, and therefore it consumes low power after implementation, making it an excellent candidate for applications such as autonomous robotics [54]. Furthermore, CDNA-SNN uses a simple training procedure based on STDP and Anti-STDP, which is more suitable for directly training SNNs on edge devices than gradient-based algorithms [55].

CDNA-weighted STDP has been introduced in this paper as a new type of STDP. CDNA-weighted STDP controls the amount of plasticity with respect to the overall activity of presynaptic neurons for the class associated with the presented spike pattern (c_n). In this approach, presynaptic neurons with higher values of CDNA for c_n will receive more weight adjustment. Compared with conventional STDP, CDNA-weighted STDP could build stronger connections between assemblies of the same class in different layers and weaker connections between assemblies of different classes (Figure 4). The results show that CDNA-weighted STDP outperformed conventional STDP by 3.65% and 3.18% for MNIST and Fashion MNIST.

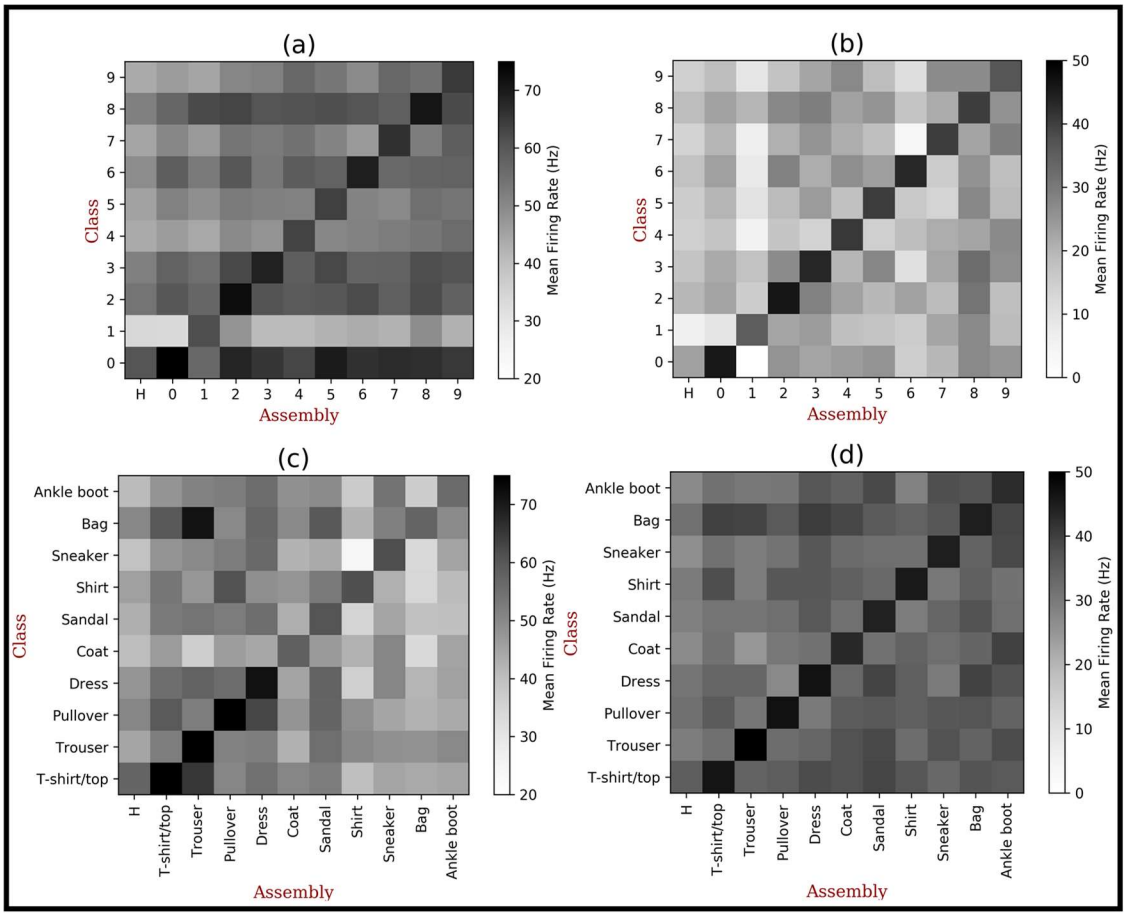

Figure 3. Mean firing rate of assemblies of a three-layer CDNA-SNN in response to testing spike patterns from different classes. The upper row shows the mean firing rate of assemblies in the (a) hidden, and (b) output layer for MNIST and the lower row illustrates the mean firing rate of assemblies in the (c) hidden and (d) output layer for Fashion MNIST. H represents the hypoactive assembly.

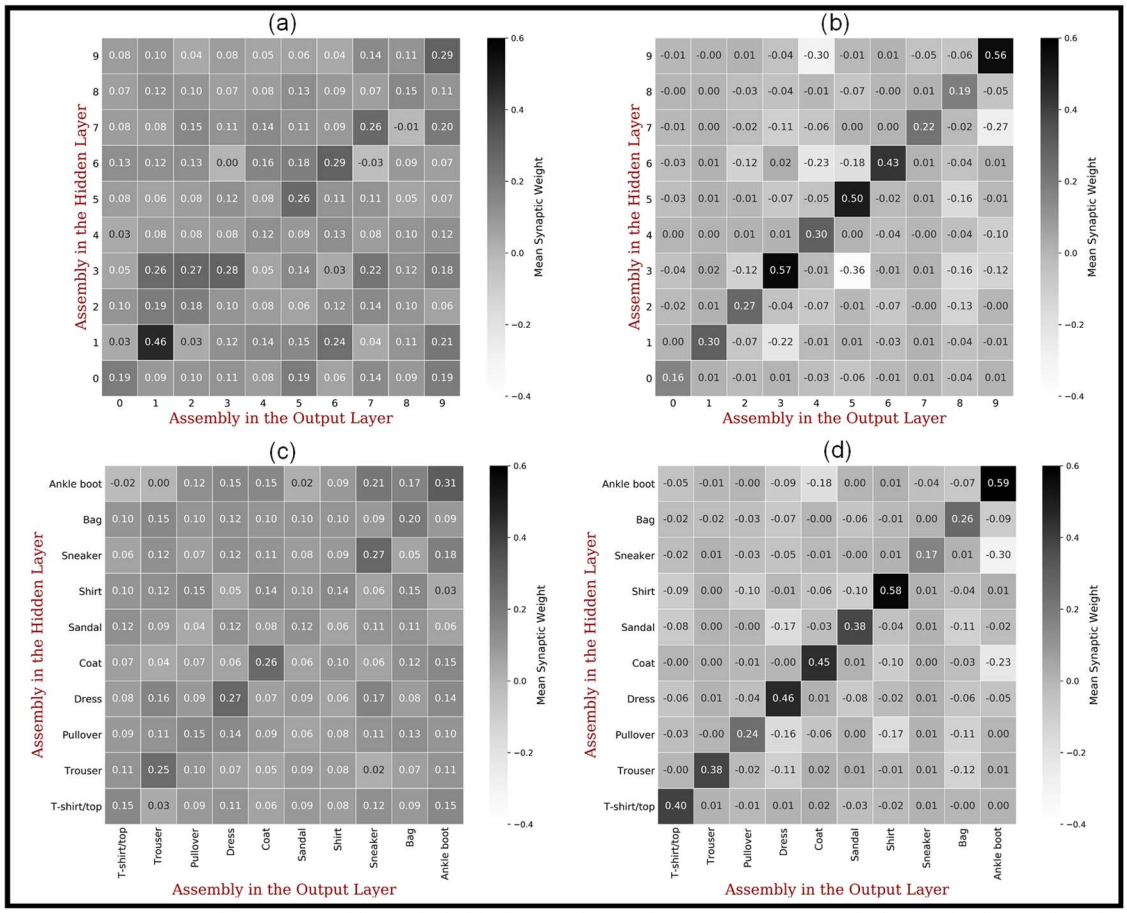

Figure 4. Average synaptic connections between assemblies in the hidden layer and output layer of CDNA-SNN, which are trained using algorithm 2 and different STDP rules in the output layer. The upper row shows average connections using (a) Conventional STDP, and (b) CDNA-Weighted STDP for MNIST, and the lower row displays average connections using (c) Conventional STDP, and (d) CDNA-Weighted STDP for Fashion MNIST

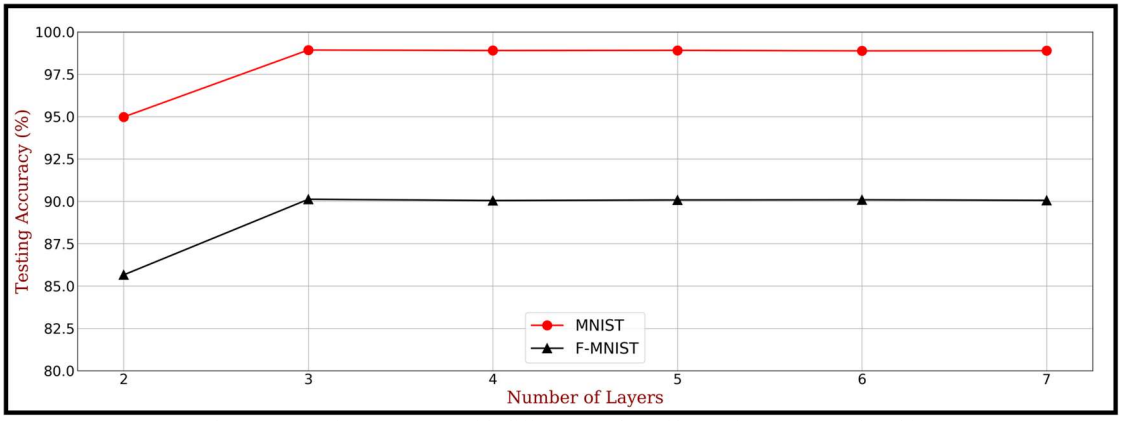

Figure 5. The accuracy of CDNA-SNN with different number of layers for MNIST and Fashion MNIST

A. Limitations

The CDNA-SNN employs a layerwise training algorithm. Layerwise training demands more epochs than end-to-end training because the total number of epochs in layerwise learning is the sum of training epochs for each layer. The training method of CDNA-SNN has no inhibitory neuron/strategy, and its only obstacle towards becoming an end-to-end approach is hypoactive neurons. According to Algorithm 1, removing hypoactive neurons in each layer is required before training the next layer. These neurons are untrained and keeping them in the network may compromise the overall classification performance of CDNA-SNN. Additionally, the training algorithm for CDNA-SNN is intended for fully connected architectures. It has been shown that for image datasets like MNIST, convolutional SNNs may reach better classification performance with fewer parameters than fully connected networks with the same training algorithm [3][11][13]. Convolutional SNNs have achieved 99.67% accuracy [56] for MNIST which is 0.76% better than CDNA-

SNN. Also, for Fashion MNIST multi-layer convolutional SNNs could reach a higher accuracy of 94.38% [57], which is 4.26% more than CDNA-SNN. This situation is even more challenging for the colored image dataset CIFAR-10 [58] and more difficult CIFAR10-DVS [59]. These datasets require several layers of convolutional spiking neural networks to achieve high performance [60][61]. CDNA-SNN with fully connected layers did not achieve comparable results to state-of-the-art convolutional SNNs on these datasets. Adapting CDNA-SNN for CNNs may yield better performance for complex image datasets like CIFAR-10 and CIFAR10-DVS and even further reductions in the parameters associated with CNNs.

V. Conclusion

In this paper, a new spiking neural network known as CDNA-SNN has been introduced, inspired by the concept of neuronal assemblies in the human brain. The proposed training algorithm for CDNA-SNN allocates learnable values called CDNAs to all neurons based on their relative firing rate for the samples from different classes. CDNAs enable this approach to categorize the neurons into different assemblies associated with each class while adjusting the synaptic weights. The novelty of this training method is to identify the most active neurons for each class, add them to the corresponding class assemblies, increase each assembly's activity for its class, and lower its firing rate for other classes. CDNAs enable the training approach to employ layer-wise learning via supervised STDP for all layers and identify and remove neurons with relatively low spiking activity, known as hypoactive neurons.

The results on multiple benchmarks indicate that the proposed network can achieve higher performance with considerably fewer network parameters than other SNNs in comparison. These characteristics make CDNA-SNN an excellent candidate for autonomous robotics and edge computing applications. As a part of the new training method, a new type of STDP entitled CDNA-weighted STDP has been proposed. In CDNA-weighted STDP CDNAs of the presynaptic neurons control the amount of weight update for their corresponding neurons. CDNA-weighted STDP, compared to conventional STDP, builds stronger connections between assemblies of the same class in consecutive layers and weaker connections between different class assemblies, leading to better classification performance.

Future work will develop an end-to-end training algorithm for CDNA-SNN to make the training process faster. Also, the proposed learning method is meant for fully connected SNNs. It is intended to adapt CDNA-SNN for convolutional architectures to improve its performance on image datasets like CIFAR10 and CIFAR10-DVS while simultaneously lowering the number of network parameters.

V. Acknowledgement

This project was supported by Dr George Moore PhD scholarship in intelligent data analytics. We are grateful for access to the Tier 2 High Performance Computing resources provided by the Northern Ireland High Performance Computing (NI-HPC) facility funded by the UK Engineering and Physical Sciences Research Council (EPSRC), Grant No. EP/T022175. Damien Coyle holds a UKRI Turing AI Fellowship 2021-2025, funded by the EPSRC (Grant number EP/V025724/1). This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems on December 1, 2023, DOI: 10.1109/TNNLS.2024.3353571. This is the author accepted manuscript (AAM) version of the article. The final published version may differ from this version. For the purpose of open access, the author(s) has applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising.

References

[1] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Networks*, vol. 111. Elsevier Ltd, pp. 47–63, 01-Mar-2019.

[2] W. Maass, "Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons," in *Advances in Neural Information Processing Systems*, 1997, pp. 211–217.

[3] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, vol. 2015-Sep, pp. 1–8.

[4] E. Hunsberger and C. Eliasmith, "Spiking Deep Networks with LIF Neurons," *[Online]. Available: http://arxiv.org/abs/1510.08829*, Oct. 2015.

[5] S. K. Esser *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 113, no. 41, pp. 11441–11446, 2016.

[6] B. Rueckauer, I. A. Lungu, Y. Hu, M. Pfeiffer, and S. C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Front. Neurosci.*, vol. 11, no. DEC, p. 682, Dec. 2017.

[7] Q. Meng, M. Xiao, S. Yan, Y. Wang, Z. Lin, and Z.-Q. Luo, "Training High-Performance Low-Latency Spiking Neural Networks by Differentiation on Spike Representation," May 2022.

[8] S. M. Bohte, J. N. Kok, and H. La Poutre, "SpikeProp: Backpropagation for networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1–4, p. 17, 2002.

[9] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," *Adv. Neural Inf. Process. Syst.*, vol. 2018-Decem, no. NeurIPS, pp. 1412–1421, 2018.

[10] P. O'Connor and M. Welling, "Deep Spiking Networks," *[Online]. Available: https://arxiv.org/abs/1602.08323*, 2016.

[11] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Front. Neurosci.*, vol. 10, no. NOV, 2016.

[12] S. R. Kulkarni and B. Rajendran, "Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization," *Neural Networks*, vol. 103, pp. 118–127, Jul. 2018.

[13] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks," *Front. Neurosci.*, vol. 12, no. MAY, pp. 1–12, May 2018.

[14] Y. Jin, W. Zhang, and P. Li, "Hybrid Macro/Micro Level Backpropagation for Training Deep Spiking Neural Networks," in *Advances in Neural Information Processing Systems*, 2018, vol. 31.

[15] A. Tavanaei and A. Maida, "BP-STDP: Approximating backpropagation using spike timing dependent plasticity," *Neurocomputing*, vol. 330, pp. 39–47, 2019.

[16] S. B. Shrestha and Q. Song, "Robust spike-train learning in spike-event based weight update," *Neural Networks*, vol. 96, pp. 33–46, Dec. 2017.

[17] S. McKennoch, L. Dingding, and L. G. Bushnell, "Fast modifications of the SpikeProp algorithm," in *IEEE International Conference on Neural Networks - Conference Proceedings*, 2006, pp. 3970–3977.

[18] N. Rathi, G. Srinivasan, P. Panda, and K. Roy, "Enabling Deep Spiking Neural Networks with Hybrid Conversion and Spike Timing

Dependent Backpropagation," in *International Conference on Learning Representations (Virtual Conference)*, 2020.
[19] D. O. Hebb, *The Organization of Behavior*. Psychology Press, 2005.
[20] A. Shrestha, K. Ahmed, Y. Wang, and Q. Qiu, "Stable spike-timing dependent plasticity rule for multilayer unsupervised and supervised learning," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, vol. 2017-May, pp. 1999–2006.
[21] C. Lee, G. Srinivasan, P. Panda, and K. Roy, "Deep Spiking Convolutional Neural Network Trained With Unsupervised Spike-Timing-Dependent Plasticity," *IEEE Trans. Cogn. Dev. Syst.*, vol. 11, no. 3, pp. 384–394, Sep. 2019.
[22] H. Markram, "A history of spike-timing-dependent plasticity," *Front. Synaptic Neurosci.*, vol. 3, no. AUG, pp. 1–24, 2011.
[23] Y. Dan and M.-M. Poo, "Spike Timing-Dependent Plasticity: From Synapse to Perception," *Physiol. Rev.*, vol. 86, no. 3, pp. 1033–1048, Jul. 2006.
[24] F. Ponulak and A. Kasiński, "Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting," *Neural Comput.*, vol. 22, no. 2, pp. 467–510, Feb. 2010.
[25] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, "SWAT: A Spiking Neural Network Training Algorithm for Classification Problems," *IEEE Trans. Neural Networks*, vol. 21, no. 11, pp. 1817–1830, Nov. 2010.
[26] A. Jeyasothy, S. Sundaram, and N. Sundararajan, "SEFRON: A New Spiking Neuron Model With Time-Varying Synaptic Efficacy Function for Pattern Classification," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 4, pp. 1231–1240, Apr. 2019.
[27] J. C. Thiele, O. Bichler, and A. Dupret, "A Timescale Invariant STDP-Based Spiking Deep Network for Unsupervised Online Feature Extraction from Event-Based Sensor Data," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, vol. 2018-July, pp. 1–8.
[28] J. C. Thiele, O. Bichler, and A. Dupret, "Event-Based, Timescale Invariant Unsupervised Online Deep Learning With STDP," *Front. Comput. Neurosci.*, vol. 12, no. June, pp. 1–13, Jun. 2018.
[29] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh, "First-Spike-Based Visual Categorization Using Reward-Modulated STDP," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 29, no. 12, pp. 6178–6190, Dec. 2018.
[30] W. Gerstner, M. Lehmann, V. Liakoni, D. Corneil, and J. Brea, "Eligibility Traces and Plasticity on Behavioral Time Scales: Experimental Support of NeoHebbian Three-Factor Learning Rules," *Front. Neural Circuits*, vol. 12, no. July, pp. 1–16, Jul. 2018.
[31] C. H. Papadimitriou, S. S. Vempala, D. Mitropolsky, M. Collins, and W. Maass, "Brain computation by assemblies of neurons," *Proc. Natl. Acad. Sci.*, vol. 117, no. 25, pp. 14464–14472, Jun. 2020.
[32] G. L. Gerstein, P. Bedenbaugh, and A. M. H. J. Aertsen, "Neuronal assemblies," *IEEE Trans. Biomed. Eng.*, vol. 36, no. 1, pp. 4–14, 1989.
[33] R. A. Fisher, "The design of experiments," 1949.
[34] R. A. Fisher, "On the'probable error'of a coefficient of correlation deduced from a small sample," *Metron*, vol. 1, pp. 1–32, 1921.
[35] R. A. Fisher, "The Design of Experiments," *British Medical Journal*, vol. 1, no. 3923. p. 554, Mar-1936.
[36] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," *Int. Jt. Conf. Artif. Intell.*, 1995.
[37] D. Dua and C. Graff, "UCI machine learning repository," 2017.
[38] S. Dora, S. Sundaram, and N. Sundararajan, "An Interclass Margin Maximization Learning Algorithm for Evolving Spiking Neural Network," *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 989–999, Mar. 2019.
[39] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Front. Neurosci.*, vol. 9, no. NOV, p. 437, 2015.
[40] S. Dora, K. Subramanian, S. Suresh, and N. Sundararajan, "Development of a Self-Regulating Evolving Spiking Neural Network for classification problem," *Neurocomputing*, vol. 171, pp. 1216–1229, Jan. 2016.
[41] T. Zhang, Y. Zeng, D. Zhao, and M. Shi, "A plasticity-centric approach to train the non-differential spiking neural networks," *32nd AAAI Conf. Artif. Intell. AAAI 2018*, pp. 620–627, 2018.
[42] Y. Hao, X. Huang, M. Dong, and B. Xu, "A biologically plausible supervised learning method for spiking neural networks using the symmetric STDP rule," *Neural Networks*, vol. 121, pp. 387–395, Jan. 2020.
[43] D. Zhao, Y. Zeng, T. Zhang, M. Shi, and F. Zhao, "GLSNN: A Multi-Layer Spiking Neural Network Based on Global Feedback Alignment and Local STDP Plasticity," *Front. Comput. Neurosci.*, vol. 14, p. 576841, Nov. 2020.
[44] S. R. Kheradpisheh and T. Masquelier, "Temporal Backpropagation for Spiking Neural Networks with One Spike per Neuron," *Int. J. Neural Syst.*, vol. 30, no. 06, p. 2050027, Jun. 2020.
[45] M. Zhang *et al.*, "An Efficient Threshold-Driven Aggregate-Label Learning Algorithm for Multimodal Information Processing," *IEEE J. Sel. Top. Signal Process.*, vol. 14, no. 3, pp. 592–602, Mar. 2020.
[46] I.-M. Comsa, K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo, and J. Alakuijala, "Temporal Coding in Spiking Neural Networks With Alpha Synaptic Function: Learning With Backpropagation," *IEEE Trans. Neural Networks Learn. Syst.*, pp. 1–14, 2021.
[47] S. R. Kheradpisheh, M. Mirsadeghi, and T. Masquelier, "BS4NN: Binarized Spiking Neural Networks with Temporal Coding and Learning," *Neural Process. Lett.*, vol. 54, no. 2, pp. 1255–1273, Apr. 2022.
[48] M. Zhang *et al.*, "Rectified Linear Postsynaptic Potential Function for Backpropagation in Deep Spiking Neural Networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 33, no. 5, pp. 1947–1958, May 2022.
[49] W. Zhang and P. Li, "Temporal spike sequence learning via backpropagation for deep spiking neural networks," in *Advances in Neural Information Processing Systems*, 2020, vol. 2020-Decem.
[50] N. Perez-Nieves and D. F. M. Goodman, "Sparse Spiking Gradient Descent," *Adv. Neural Inf. Process. Syst.*, vol. 15, pp. 11795–11808, May 2021.
[51] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training Deep Spiking Neural Networks Using Backpropagation," *Front. Neurosci.*, vol. 10, no. NOV, Nov. 2016.
[52] G. K. Cohen, G. Orchard, S.-H. Leng, J. Tapson, R. B. Benosman, and A. van Schaik, "Skimming Digits: Neuromorphic Classification of Spike-Encoded Images," *Front. Neurosci.*, vol. 10, no. APR, p. 184, Apr. 2016.
[53] Y. Jin, W. Zhang, and P. Li, "Hybrid macro/micro level backpropagation for training deep spiking neural networks," in *Advances in Neural Information Processing Systems*, 2018, vol. 2018-Decem, pp. 7005–7015.
[54] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. C. Knoll, "A Survey of Robotics Control Based on Learning-Inspired Spiking Neural Networks," *Front. Neurorobot.*, vol. 12, p. 35, Jul. 2018.
[55] K. Bai, S. Liu, and Y. Yi, "High Speed and Energy Efficient Deep Neural Network for Edge Computing," *Proc. 4th ACM/IEEE Symp. Edge Comput.*, 2019.
[56] G. Shen, D. Zhao, and Y. Zeng, "Backpropagation with biologically plausible spatiotemporal adjustment for training deep spiking neural networks," *Patterns*, vol. 3, no. 6, p. 100522, Jun. 2022.
[57] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 2641–2651.
[58] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, "CIFAR10-DVS: An event-stream dataset for object classification," *Front. Neurosci.*, vol. 11, no. MAY, p. 244131, May 2017.
[59] G. Kevin Cohen *et al.*, "Citation: CIFAR10-DVS: An Event-Stream Dataset for Object Classification," *Front. Neurosci. | www.frontiersin.org*, vol. 1, p. 309, 2017.
[60] H. Wu *et al.*, "Training Spiking Neural Networks with Accumulated Spiking Flow," 2021.
[61] A. Safa, I. Ocket, A. Bourdoux, H. Sahli, F. Catthoor, and G. Gielen, "A New Look at Spike-Timing-Dependent Plasticity Networks for Spatio-Temporal Feature Learning," Nov. 2021.

Vahid Saranirad received the B.S. degree in electrical engineering from the Sadjad University of Technology, Mashhad, Iran, in 2010 and the M.Sc. degree in biomedical engineering from the Ferdowsi University of Mashhad, Mashhad, Iran, in 2014. He is currently working toward the Ph.D. degree in computer science with Intelligent Systems Research Centre (ISRC), School of Computing, Engineering and Intelligent Systems, Ulster University, Derry/Londonderry, U.K. His research interests focus on enhancing the biological plausibility of deep learning for computer vision.

Shirin Dora is a lecturer in the Department of Computer Science at Loughborough University. Prior to joining Loughborough University, he briefly worked as a lecturer in the Intelligent Systems Research Centre at the Ulster University in Northern Ireland. He received his PhD from Nanyang Technological University, Singapore in 2017. During his PhD, he focused on developing energy-efficient learning algorithms for spiking neural networks using inspirations from the brain. After his PhD, he joined the Cognitive and Systems Neuroscience Group at the University of Amsterdam as a postdoctoral researcher. In Amsterdam, he delved deeper into the plasticity mechanisms in the brain specifically focusing on predictive coding for perception and multisensory integration. His current research interests include efficient techniques for lifelong learning, few-shot learning and distributed AI.

T. Martin McGinnity (SM'09) received the BSc degree in Physics (First Class hons.) from the New University of Ulster in 1975 and a PhD degree from the University of Durham, U.K. in 1979.
Currently he is an Emeritus Professor in the School of Computing, Engineering and Intelligent Systems at Ulster University (UU). Before taking semi-retirement in 2018, he was formerly Pro Vice Chancellor and Head of the College of Science and Technology at Nottingham Trent University (NTU), Dean of Science and Technology at NTU, Founding Director of the Intelligent Systems Research Centre at UU, and Head of the School of Computing and Intelligent Systems at UU. He is the author or co-author of over 300 research papers, has supervised over 30 PhD students to successful completion and attracted over £40 million in research funding. His research interests are focused on computational intelligence, computational neuroscience, modelling of biological information processing in FPGA reconfigurable hardware and sensory systems in cognitive robotics.

Damien Coyle (Senior Member, IEEE) is a Professor of Neurotechnology, Director of the Bath Institute for the Augmented Human and a UKRI Turing AI Acceleration Fellow 2021-25. He was Director of Ulster University's Intelligent Systems Research Centre 2017-2022. His research focuses on developing AI to address challenges associated with translating electrophysiological signals into control signals for brain-computer interface (BCI) based neurotechnology and trialing these developments on a large scale with patients and end-users. He has published over 190 research papers in areas such as computational intelligence/AI, bio-signal processing, computational neuroscience, neuroimaging, neurotechnology and brain-computer interface (BCI) applications and has won a number of prestigious international awards including the 2008 IEEE Computational Intelligence Society (CIS) Outstanding Doctoral Dissertation Award, the 2011 International Neural Network Society (INNS) Young Investigator of the Year Award and the IET and E&T Innovation of the Year Award 2018. He was an Ulster University Distinguished Research Fellow in 2011 and Ulster Senior Distinguished Research Fellow 2020, a Royal Academy of Engineering/The Leverhulme Trust Senior Research Fellow in 2013, and a Royal Academy of Engineering Enterprise Fellow in 2016-2017. He secured over £20m external grant income and managed several industry-led data analytics projects. He has supervised 22 PhD researchers (14 completed). He is a founding member of the International Brain-Computer Interface Society, an IEEE Brain Technical Community Steering Committee member, advisory board member for the UK Neurotechnology Innovation Network, and previously chaired the IEEE Computational Intelligence Society (CIS) UKIreland chapter. He is Founder and CEO of NeuroCONCISE Ltd (www.neuroconcise.co.uk), an award-winning, AI-enabled, wearable neurotechnology company.