Unsupervised and efficient learning in sparsely activated convolutional spiking neural networks enabled by voltage-dependent synaptic plasticity

Gaspard Goupy, Alexandre Juneau-Fecteau, Nikhil Garg, Ismael Balafrej, F. Alibart, Luc Frechette, Dominique Drouin, Yann Beilliard

Unsupervised and efficient learning in sparsely activated convolutional spiking neural networks enabled by voltage-dependent synaptic plasticity

NEUROMORPHIC
Computing and Engineering

PAPER

Unsupervised and efficient learning in sparsely activated convolutional spiking neural networks enabled by voltage-dependent synaptic plasticity

Gaspard Goupy[1,*], **Alexandre Juneau-Fecteau**[1], **Nikhil Garg**[1,2,3], **Ismael Balafrej**[1,4], **Fabien Alibart**[1,2,3], **Luc Frechette**[1,2], **Dominique Drouin**[1,2] and **Yann Beilliard**[1,2]

[1] Institut Interdisciplinaire d'Innovation Technologique (3IT), Université de Sherbrooke, Sherbrooke, Canada
[2] Laboratoire Nanotechnologies Nanosystèmes (LN2)—CNRS UMI-3463, Université de Sherbrooke, Sherbrooke, Canada
[3] Institute of Electronics, Microelectronics and Nanotechnology (IEMN), Université de Lille, Villeneuve d'Ascq, France
[4] NECOTIS Research Lab, Electrical and Computer Engineering Department, Université de Sherbrooke, Sherbrooke, Canada
* Author to whom any correspondence should be addressed.

E-mail: gaspard.goupy@protonmail.com

Keywords: convolutional spiking neural networks, sparsely activated neural networks, single spike neuron, unsupervised learning, voltage-dependent synaptic plasticity, hardware-friendly STDP

OPEN ACCESS

RECEIVED
30 September 2022

REVISED
26 November 2022

ACCEPTED FOR PUBLICATION
21 December 2022

PUBLISHED
6 January 2023

Abstract

Spiking neural networks (SNNs) are gaining attention due to their energy-efficient computing ability, making them relevant for implementation on low-power neuromorphic hardware. Their biological plausibility has permitted them to benefit from unsupervised learning with bio-inspired plasticity rules, such as spike timing-dependent plasticity (STDP). However, standard STDP has some limitations that make it challenging to implement on hardware. In this paper, we propose a convolutional SNN (CSNN) integrating single-spike integrate-and-fire (SSIF) neurons and trained for the first time with voltage-dependent synaptic plasticity (VDSP), a novel unsupervised and local plasticity rule developed for the implementation of STDP on memristive-based neuromorphic hardware. We evaluated the CSNN on the TIDIGITS dataset, where, helped by our sound preprocessing pipeline, we obtained a performance better than the state of the art, with a mean accuracy of 99.43%. Moreover, the use of SSIF neurons, coupled with time-to-first-spike (TTFS) encoding, results in a sparsely activated model, as we recorded a mean of 5036 spikes per input over the 172 580 neurons of the network. This makes the proposed CSNN promising for the development of models that are extremely efficient in energy. We also demonstrate the efficiency of VDSP on the MNIST dataset, where we obtained results comparable to the state of the art, with an accuracy of 98.56%. Our adaptation of VDSP for SSIF neurons introduces a depression factor that has been very effective at reducing the number of training samples needed, and hence, training time, by a factor of two and more, with similar performance.

1. Introduction

Over the last decade, convolutional neural networks (CNNs) have been widely used in deep learning [1] to solve several types of tasks, such as visual [2–4] or auditory [5–7] ones, outperforming previous methods. However, although CNNs can achieve high performance, they are still limited by their computational cost and their significant energy consumption. Indeed, CNNs use second-generation artificial neurons based on the McCulloch-Pitts model [8], which states that neurons have floating and continuous activations, which limits their implementation on resource-restricted hardware.

Towards a bio-inspired approach, spiking neural networks (SNNs) [9], known as the third generation of artificial neural networks, are increasingly studied because of their low energy consumption. In these networks, neurons transmit and process information similarly to biological neurons, with asynchronous

spikes. They integrate a temporal dynamic, making activations event-driven and binary. Convolutional SNNs (CSNNs) have already been used to solve various tasks with great performance [10–13], although their development is still in its early stages compared with CNNs.

Given the non-differentiable nature of their activation functions, classical gradient-based learning algorithms are not directly applicable in SNNs. Adaptations of Backpropagation for SNNs have already been proposed [10, 14, 15]. However, these rules are limited by their global nature, needing network-level communications, which is difficult to implement in hardware. While recent works [16, 17] have proposed a hardware-friendly and local backpropagation rule, its supervised structure requires the use of labels as well as a cost function. Also, its hardware implementation is not memory-efficient since feedback synapses must be stored. Inspired by biology, new unsupervised rules based on mechanisms of local plasticity have been developed. Spike timing-dependent plasticity (STDP) is the most widely used in this context, where the relative time difference between the pre- and postsynaptic neuron spikes defines the plasticity [18]. STDP is an interpretation of the main learning rule observed in biological synapses, described by Hebb's theory [19]. The literature has already shown the efficiency of learning with STDP to solve classical deep learning tasks [11, 13, 20–22]. Yet, standard STDP still faces a significant challenge: it requires recording and updating the spike traces of the neurons, i.e. the timing of their last spike, and applies to both pre- and postsynaptic neuron spikes. This imposes an additional memory requirement, which may limit large-scale software applications or implementations on resource-restricted hardware, and also imposes an energy requirement to update the traces. Moreover, STDP is constrained by its time window parameter, in which the spike time difference must fall to update the weight significantly, which needs to be optimised for each task.

Consequently, a novel unsupervised and local plasticity rule, called voltage-dependent synaptic plasticity (VDSP) [23], has recently been developed to address STDP issues. VDSP is a hardware-friendly alternative approach to STDP (and other Hebbian-based rules) developed for the implementation of Hebb's plasticity mechanism on memristive-based neuromorphic hardware. In this type of hardware, the synapses are implemented with an emerging memory technology that enables in-memory computing [24]. The rule uses the membrane potential of the presynaptic neuron instead of its spike timing to evaluate pre/post neurons correlation, with the following assumption: high membrane potential reflects a neuron that is about to fire, whereas a negative membrane potential reflects a neuron that has recently fired. Hence, no extra memory is needed for storing spike traces, as membrane potential is readily available as part of the neuron implementation. In addition, only the postsynaptic neuron spike event is used to trigger the updating of the weights, which reduces by two the number of updates with respect to standard STDP. Finally, VDSP is not based on a fixed scale of spike time difference to update the weights, which removes the need for the time window parameter. Note that the suitability of VDSP in neuromorphic hardware is dependent on its architecture. In the case of in-memory computing based neuromorphic hardware, the synapses are physically connected to both pre- and postsynaptic neurons. The membrane potential is, hence, readily and locally available to the synapses. VDSP is new and has been implemented only on a one-layer fully connected network for now [23]. Further research has to be done to evaluate the scalability and the performance of VDSP in other network architectures.

In this paper, inspired by the model developed in [11] and the input encoding used in [21], we implemented for the first time VDSP in a CSNN. We aimed to study the influence of VDSP in convolutional networks on the performance, while minimising spike counts to create a sparsely activated model suitable for edge computing with implementation on neuromorphic hardware. The model comprises an input layer, a convolutional layer, and a max-pooling layer. The input image is encoded efficiently into spike bins with time-to-first-spike (TTFS) [25] temporal algorithm. Neurons of the CSNN are single-spike integrate-and-fire (SSIF), which are IF neurons that can fire at most once. The output of the CSNN is used to train the readout, or output layer, implementing a linear support vector machine (SVM). We show that the proposed CSNN can be used for various applications, as we have evaluated it on the speaker-independent isolated spoken digits classification task with the dataset TIDIGITS [26] and on the handwritten digits classification task with the dataset MNIST [27]. The accuracy of 98.56%, recorded for MNIST, is comparable to the state of the art. On TIDIGITS, it, helped by our preprocessing pipeline, outperforms the state of the art, obtaining 99.43%. Further analysis demonstrates that, besides handling unlabelled data, VDSP requires only a few samples for training and the weights tend toward binary values. Also, TTFS and SSIF neurons make the network sparsely activated: we recorded a mean of 2.9% activations over 172 580 neurons per input for TIDIGITS, which is promising for edge computing with implementation on low-power hardware. Our adaptation of VDSP introduces a depression factor that has been extremely effective at reducing the number of training samples needed, and hence, training time, by a factor of two and more, with similar performance. The source code of the proposed CSNN is publicly available at [28].

2. Methods

2.1. Network architecture

The architecture of the CSNN is composed of an input layer, a convolutional layer, and a max-pooling layer, illustrated in figure 1. First, TTFS is used to convert efficiently an image into spike bins, with at most one spike per pixel. The input layer propagates the spike bins to the convolutional layer, which learns features with a winner-take-all based adaptation of VDSP. The max-pooling layer compresses the feature maps to reduce the size of the output and provide invariance to translation on the input image. The neuron model used in the CSNN is SSIF, so they can fire at most once per input, leading to a sparsely activated model. The sum of the spikes for each max-pooling neuron over all timesteps is recorded and gathered in a 1-dimensional output vector. As neurons can fire at most once, this vector represents the binary state of the max-pooling neurons and is used to train the readout layer, a linear SVM, for the classification task.

2.2. Input preprocessing

Because of their architecture, CNNs can work with images as input. Indeed, they use two-dimensional kernels to extract patterns between neighbouring pixels. SNNs integrate a time dimension in their functioning. Hence, it is necessary to encode the inputs into discrete spike bins before propagating them to the network. In this way, we designed a preprocessing pipeline for acoustic signals, illustrated in figure 2, to transform a sound sample into an image and encode it into spike bins. Note that when addressing plain image inputs, the pipeline is only composed of the encoding step.

For sound inputs, the first stage of the pipeline consists in trimming the samples to extract the human voice. However, as the CSNN can not handle inputs of various sizes, we then zero-padded all samples to match the length of the longest trimmed one. Secondly, we transformed the sounds into images with a log-mel spectrogram (LMS), which is a visual representation of a sound, including both time and frequency information, and obtained by using the discrete Fourier transform (DFT). LMS is bio-inspired, as it is based on a mel-scale, proposed by Stevens and Volkmann [29], showing that humans do not hear frequencies on a linear scale.

The last stage of the pipeline consists in encoding the image into spike bins. Several coding schemes have already been proposed in the literature [30], grouped into two main categories: rate coding and temporal coding. Rate coding integrates the information in the neuron firing rate, which can be time-consuming and inefficient in energy. In contrast, temporal coding represents the information through the precise timing of the spike. TTFS [25] is a temporal algorithm, already used in [21, 31, 32], that encodes information by the time difference between the onset of a stimulus and the first spike of the neuron. For an image input, the pixel value is inversely proportional to its response time. Thus, each pixel is represented by a single spike, which makes the encoding fast and efficient in energy. This encoding is biologically plausible as it has been found in the human visual [33] and auditory [34] sensory systems. TTFS encodes an image into N spike bins, two-dimensional images with the same size as the input image, containing binary pixels, i.e. spikes, for a precise timestep.

2.3. Spiking neuron model

We call the model of the neurons used in the CSNN the SSIF model, presented in figure 1. This model is described in [11] by integrate-and-fire (IF) neurons that can fire at most once. The IF model is one of the simplest models, as the neuron membrane potential V is incremented when it receives spikes from presynaptic neurons, but it does not decrease with time, unlike the leaky IF (LIF) model. In addition, the single-spike constraint ensures a sparsely activated model, which is relevant for edge computing with low-power hardware. The membrane potential of neuron i is initialised to V_{rest} and, at each timestep t, it is updated according to the following rule:

$$V_i(t) = V_i(t-1) + \sum_j w_{ji} \cdot S_j(t-1) \tag{1}$$

with S_j spikes of presynaptic neurons j and w_{ji} synaptic weights of the connection between neurons j and i. When V exceeds a threshold V_{thr}, it is reset to V_{reset} and the neuron fires. Between samples, the neurons are reinitialised to V_{rest}. It is important to mention that V_{reset} and V_{rest} must be different for VDSP to work properly. Hence, we set the voltage convention of $V_{\text{reset}} = -1$ and $V_{\text{rest}} = 0$. The single-spike model is consistent with the TTFS encoding used, as neurons can fire at most once. Thus, during a simulation of t timesteps, the network is guaranteed to emit no more than N spikes, with N the total number of neurons.

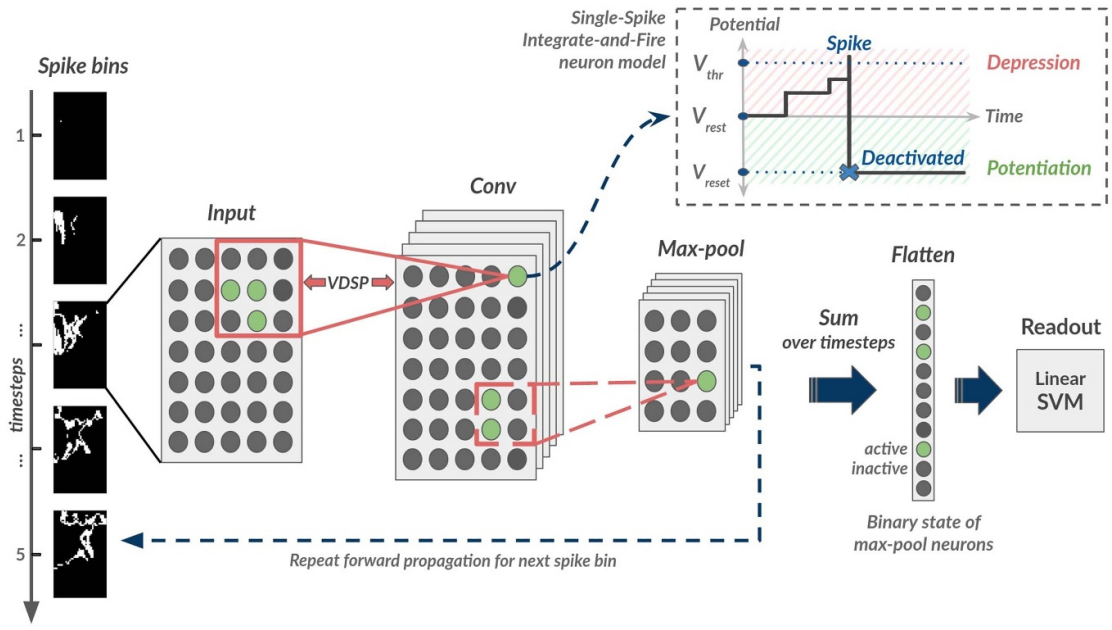

Figure 1. Architecture of the CSNN, composed of an input layer, a convolutional layer, and a max-pooling layer. The neuron model is the single-spike integrate-and-fire (SSIF), which are IF neurons that can fire at most once. Spike bins, encoded efficiently with time-to-first-spike (TTFS), are propagated successively in the network by the input layer. The convolutional layer extracts features from the input image (encoded into spike bins) and is trained in an unsupervised and local fashion with voltage-dependent synaptic plasticity (VDSP), a novel hardware-friendly learning rule based on Hebb's plasticity mechanism. VDSP uses the presynaptic neuron membrane potential V to evaluate pre/post neurons correlation: connection where presynaptic neuron membrane potential $V > V_{\text{rest}}$ is depressed whereas it is potentiated when $V = V_{\text{reset}}$. Max-pooling neurons perform a maximum operation in their corresponding window to reduce the size of the feature maps and provide invariance to translation. The output of the CSNN is a flattened feature vector representing the sum of spikes of each max-pooling neuron over all timesteps. As neurons can fire at most once, the vector's values are binary. Finally, the output is propagated to the readout layer, implementing a linear support vector machine (SVM), for the classification task.

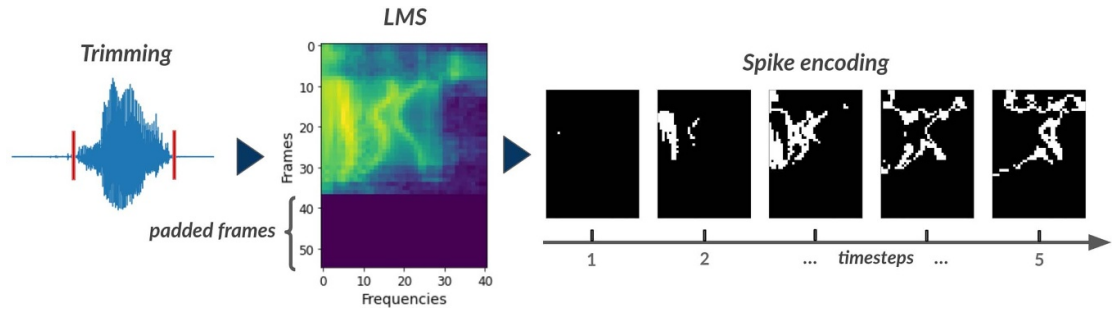

Figure 2. Sound preprocessing pipeline. First, the signal is trimmed to capture the human voice and padded at the end, according to the longest trimmed input, to get samples of the same size. Then, the log-mel spectrogram (LMS) is extracted to obtain an image of size *Frames* × *Frequency bands*. Lastly, the image is encoded efficiently into spike bins with time-to-first-spike (TTFS) temporal algorithm.

2.4. Input layer

The input layer is used in an offline fashion with spike bins as inputs. In this way, the CSNN is compatible with both plain images, that we can encode into spike bins, or already encoded ones. However, note that the layer can also be implemented as an online TTFS encoder, as it is in [21]. The input layer is composed of SSIF neurons organised in a two-dimensional grid, with the same size as the input images, each neuron corresponding to a pixel. First, the layer stacks all spike bins from an input image. Then, at each timestep t, the membrane potential of the input neuron i is updated as follows:

$$V_i(t) = V_i(t-1) + S_i$$
$$\text{with } S_i = \frac{V_{\text{thr}}}{t_i} \tag{2}$$

S_i is called the potential step of the neuron i and t_i is the firing timestep (i.e. the spike bin number) of the pixel corresponding to the neuron i. Neurons fire when their membrane potential exceeds V_{thr}, which corresponds to the timestep where their corresponding pixel is activated in the input spike bins. Hence, forward propagation is done in the same number of timesteps as the number of spike bins. The purpose of the input layer is to propagate the input spike bins to the convolutional layer with SSIF neurons, which is mandatory for VDSP.

2.5. Convolutional layer

The convolutional layer is composed of several feature maps, containing SSIF neurons organised in a two-dimensional grid. Each neuron of a specific feature map is receptive to a unique 2D window in the input layer, corresponding to the convolution operation carried out to update the neuron's potential. The dimension of the 2D windows is equal to the dimension of the kernels containing the synaptic weights for each feature map. Weights are shared between neurons of the same map, which makes it possible to detect the same features at different locations of the input. In addition, it makes training much faster, as the number of weights is considerably reduced.

The convolutional layer also implements a lateral inhibition mechanism, often used in SNNs: when a neuron of a feature map fires, it deactivates all neurons at the same position in the other feature maps, resetting their potential to V_{rest} and preventing them from updating it until the end of the propagation. If several neurons at the same position in different feature maps fire at the same time, the spike of the one with the highest potential is preserved and other spikes are inhibited. This principle reduces redundant information and ensures the model has few activations. E.g., for a convolutional layer of $C \times N \times M$ neurons (with C the channels, N the rows, M the columns), only $N \times M$ spikes can be emitted per input.

2.6. Max-pooling layer

The max-pooling layer is identical to the ones used in second generation CNNs. It is composed of the same number of feature maps as the convolutional layer. Each neuron of a feature map is connected to a unique 2D window in the corresponding map of the convolutional layer and performs a maximum operation on the output spikes in the window. Its synaptic weights and its threshold V_{thr} are both fixed to 1. Hence, the neuron fires when a presynaptic neuron of its window emits a spike. As with the SSIF model, max-pooling neurons can also fire at most once per input. The purpose of the layer is to compress the feature maps so as to reduce the size of the output and make the network robust to translations of the input image.

2.7. Learning with VDSP

Learning in the CSNN is performed online, in an unsupervised and local fashion, with a winner-take-all based adaptation of VDSP [23] for SSIF neurons. VDSP is a novel, hardware-friendly, alternative approach to STDP, developed for the implementation of Hebb's plasticity mechanism on memristive-based neuromorphic hardware. It is implemented in the convolutional layer and performed at each timestep on the spikes of the postsynaptic neurons. Unlike global rules, where the update of the weights considers the output of the model, thus requiring backpropagation through all layers, local rules use only the local information of the neurons, making them more efficient in terms of computation time. Moreover, this locality significantly facilitates hardware implementation by avoiding the need for network-level communications. The main idea behind VDSP is that a high membrane potential reflects a neuron that is about to fire, whereas a negative membrane potential reflects a neuron that has recently fired. However, as the SSIF model differs from the LIF model used in the original paper, we made an adaptation of the rule, formulated as follows:

$$\Delta w_{ji} = \begin{cases} w_{ji}(w_{\max} - w_{ji}) \cdot lr & \text{if } V_{\text{pre}} = V_{\text{reset}} \\ w_{ji}(w_{\max} - w_{ji}) \cdot lr \cdot (V'_{\text{pre}} - f_{\text{dep}}) & \text{if } V_{\text{pre}} \geqslant V_{\text{rest}} \end{cases} \tag{3}$$

where i and j and respectively refer to the index of post- and presynaptic neurons, Δw_{ji} is the change in weights, w_{ji} are the current weights, $w_{\max}$ the maximum weight value, lr the learning rate, V_{pre} the membrane potential of the presynaptic neuron j, and f_{dep} the depression factor (must be $\geqslant 1$). Note that V'_{pre} is the value of V_{pre} normalised to the range $[0, 1]$. Also, $w_{\max} - w_{ji}$ is a soft-bound term used to clip weights in the range $[0, w_{\max}]$, to prevent the explosion of the values of the weights.

As the neurons of the CSNN are single-spike, it is not possible to exploit the timing assumption with $e^{V_{\text{pre}}}$ from the original formula of the VDSP paper, making the value of Δw proportional to the last spike timing of the presynaptic neuron (or the magnitude of its potential). Indeed, in the original formula, the higher V_{pre} or $-V_{\text{pre}}$ is, the bigger is the weight update. This mechanism works as the neurons are LIF and can fire multiple times. With IF neurons, it is not relevant to consider spike timing for potentiation, as the membrane potential of a postsynaptic neuron can not decrease, making presynaptic neurons that have fired equally important as

each other. However, for depression, we reproduced the idea of the original formula by introducing a depression factor with the term $V'_{\text{pre}} - f_{\text{dep}}$, making the depression of connections where the presynaptic neuron has a membrane potential close to V_{rest} faster than when it is too close to V_{thr}. Here, instead of seeking the last spike time, we make an assumption about the next spike time, as neurons are single-spike. Therefore, connections where presynaptic neurons that are likely to fire in a long time are depressed quicker, leading to a more efficient training. Also, when f_{dep} is sufficiently high, it may speed up training considerably.

Inspired by biological processes in visual search tasks [35], a winner-take-all (WTA) topology is used during learning. WTA plays an important role in avoiding having neurons at neighbouring positions in different feature maps react to the same pattern, but it also increases the efficiency of the training and reduces the computational cost. To do so, at each timestep t, only k winning neurons in the layer are allowed to update their weights with VDSP. The choice of winners is made by taking the neurons that are about to fire and have the highest potential. In addition, there can be only one winner per feature map (i.e. global intra-map competition) and only one winner in the neighbourhood of a position (i.e. local inter-map competition). The neighbourhood is defined by a 2D window of size r_{inhib} around the winning neuron. Winning neurons disable the ability to carry out VDSP for all neurons in their feature map as well as neurons in their neighbourhood in other maps, thus preventing them from updating their weights until the end of the propagation.

While training continues, VDSP iterations are recorded and the learning rate is multiplied by two every lr_{step} learning steps, until reaching a maximum defined by lr_{max}. The learning rate is initially kept low to prevent the significant depression effect caused by neurons responding to every pattern. As neurons begin to recognise and react to fewer patterns, it is gradually increased to amplify long-term potentiation and depression. Training is stopped when the learning convergence C, described in [11] by the formula 4, is lower than 0.01, meaning that the weights are sufficiently close to $w_{\text{min}} = 0$ or w_{max}.

$$C = \frac{\sum_j \sum_i w_{ji} \times (w_{\text{max}} - w_{ji})}{n_w} \tag{4}$$

with w_{ji} the weights and n_w the total number of weights in the convolutional layer.

2.8. Readout

The readout is the output layer of the model. This layer uses the 1-dimensional feature vector produced by the CSNN, as described in figure 1, to classify the input. The feature vector corresponds to the binary state of the max-pooling neurons (i.e. if they have fired or not). To do so, the output value of each max-pooling neuron is summed over timesteps and then gathered into a 1-dimensional vector. As the neurons are single-spike, the values of the vector are binary. The main assumption of the readout is that the CSNN can extract sufficiently distinct features to make the output data linearly separable, allowing a simple algorithm to make the decisions. Hence, we implemented a linear SVM as a readout function that is trained on the CSNN outputs. Note that the SVM is only used to assess how discriminative the features the CSNN extracted are and it is not part of the model in itself.

3. Results

We evaluated the proposed CSNN on the speaker-independent isolated spoken digits classification task with the dataset TIDIGITS [26] and on the handwritten digits classification task with the dataset MNIST [27]. The parameters of the CSNN used in all of our experiments are described in table 1. The input images are encoded into 15 spike bins with TTFS. The weights of the convolutional layer are clipped between $w_{\text{min}} = 0$ and $w_{\text{max}} = 1$, and are initialised randomly, following a normal distribution with a mean of 0.8 and a standard deviation of 0.05. The learning rate of the VDSP in the convolutional layer was defined by $lr_{\text{init}} = 0.01$, $lr_{\text{final}} = 0.1$, and updated every $lr_{\text{step}} = 500$ learning steps, with $f_{\text{dep}} = 2$. The readout uses the implementation of a linear SVM in the library scikit-learn, with the regularisation parameter $C = 0.005$. To optimise the parameters of the CSNN, we used manual tuning and randomised search technique. All of the parameters used in our model are also listed in appendix (tables 4, 5, 6).

The evaluation process consists of:

(a) Train in an unsupervised manner the convolutional layer with VDSP on training samples
(b) Freeze VDSP and record output vectors for both training and testing samples
(c) Train in a supervised manner the readout with training vectors and predict classes for testing vectors.

Table 1. CSNN parameters used for experiments.

	Feature maps	Kernel	Stride	Padding	V_{thr}	$n_{winners}$	r_{inhib}
Conv	70	7	1	3	10	7	3
Pool	70	3	3	0	1	—	—

Table 2. Accuracy of proposed CSNN and other methods from the literature on the TIDIGITS dataset.

Model	Architecture	Learning type	Learning rule	Number of classes	Accuracy (%)
Shrestha and Orchard [36]	SNN	Supervised	SLAYER	11	99.09
Jia *et al* [37]	MR-SNN	Supervised	Reward learning	10	98.63
Jia *et al* [38]	NRR-SNN	Supervised	NRR	10	98.34
Wu *et al* [39]	SOM-SNN+ Tempotron-like	Unsupervised	Competitive learning	11	97.60
Dong *et al* [21]	CSNN+SVM	Unsupervised	STDP	10	97.50
Zhang *et al* [40]	CSNN	Supervised	BRP	10	94.86
This work	**CSNN+SVM**	**Unsupervised**	**VDSP**	**11**	**99.43**

3.1. Spoken digits classification with TIDIGITS

TIDIGITS is a dataset containing acoustic signals sampled at 20 kHz from 326 speakers (111 men, 114 women, 50 boys, and 51 girls), each pronouncing 77 sequences of varying lengths of digits from 'zero' to 'nine' and 'oh'. In our experiment, we used the 4950 isolated spoken digit utterances from men and women only. The audio signals were re-sampled at 16 kHz, trimmed with a threshold of 20 dB, and padded to a length of 13 824, or 864 ms (maximum sample length after trimming). Then, they were split randomly into training and test sets with a ratio 7:3. LMS were extracted with the following parameters: 512 FFT frames, hop length of 256 40 mel bins, frequency range from 0 Hz to 8000 Hz, producing images with a size of 55 frames $\times$ 40 frequency bands.

We evaluated the proposed CSNN on the TIDIGITS dataset and we obtained a mean accuracy of $99.43 \pm 0.14\%$ over ten tries, for the test set. In table 2, the performance of the CSNN is compared with the literature. With the proposed CSNN and our preprocessing pipeline, we obtained an accuracy higher than the state of the art for SNNs, with a shallow architecture and an unsupervised hardware-friendly plasticity rule. Note that our preprocessing pipeline, and especially the trimming step, plays an important role in the performance. Without trimming, the accuracy of the CSNN is $97.76 \pm 0.46\%$, which is similar to other works with unsupervised architectures. Also, the accuracy with and without trimming is, respectively, 98.57% and 94.26% for the linear SVM trained on LMS features. We observed a mean of 5036 spikes per sample in the network over 172 580 neurons, i.e. around 2.9% activation in the network, demonstrating that the use of TTFS and SSIF neurons results in a sparsely activated model. This makes the CSNN promising for hardware implementation on low-power devices, needing extremely energy-efficient models. In addition, with VDSP, we do not have to save the traces of the neurons, which saves 172 580 neurons $\times$ 4 bytes, i.e. 690 32 Kilobytes of memory, assuming the size of a trace is 4 bytes. Note that the amount of memory saved in this case with respect to STDP is sufficiently small to be ignored but it can be much bigger for large scale application. Also, for hardware implementation, circuit design and miniaturisation could be easier, as no extra memory is needed.

To better analyse the learning process with VDSP, we evaluated the performance of the CSNN on the test set at different times throughout the training. Figure 3 shows the mean accuracy measured over ten tries for different numbers of training samples. The accuracy without training is 98.77% and it stabilises at $\sim$99.3% after 250 samples. The accuracy increases quickly from 98.77% without training to 99.38% after 50 training samples and then stabilises at $\sim$99.3% after 250 samples. However, the convergence of the learning of the convolutional layer happens at approximately 200 samples, which normally stops the CSNN training. This makes the WTA-based VDSP learning rule particularly attractive because, besides being able to process unlabelled datasets, it requires little data for training. It is important to mention that TIDIGITS classification is a fairly simple task and the mean accuracy of the linear SVM trained on LMS features is already high, 98.57%, which explains why the increase in accuracy with the CSNN is small. Nonetheless, it is also known that the last gains in accuracy are the hardest to get. Larger accuracy differences are expected with more challenging classification tasks. The figure also plots the weight distributions after 0, 50, and 300 training samples. Initially, the weights are initialised randomly, following a normal distribution with a mean of 0.8 and a standard deviation of 0.05. During training, they are either depressed or potentiated until getting pretty close to $w_{min} = 0$ and $w_{max} = 1$, demonstrating another property of VDSP, useful for hardware

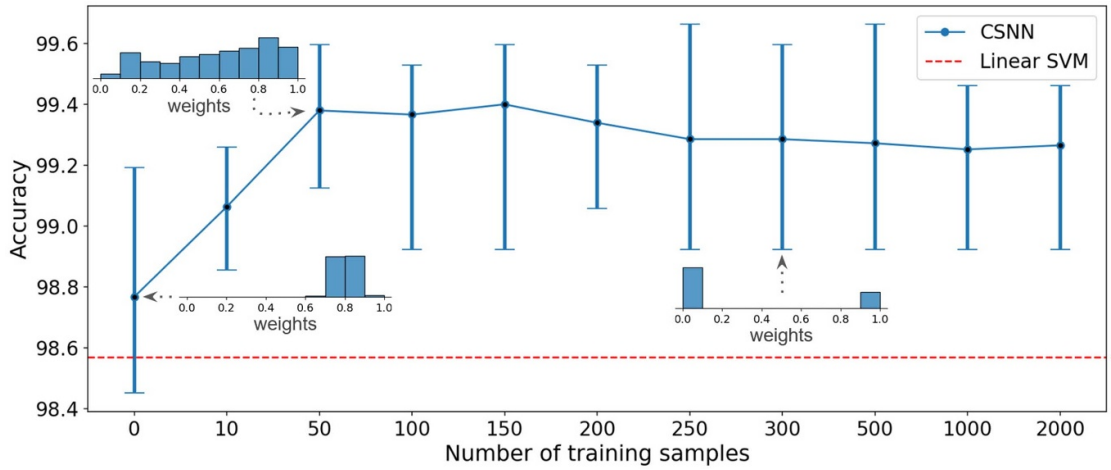

Figure 3. Mean accuracy averaged over ten tries on the test set for different numbers of training samples in the training of the CSNN. The mean accuracy of a linear SVM trained on LMS features is 98.57%. The learning process of the CSNN is illustrated: its accuracy increases as the number of samples used for training grows, until converging after 250 samples. The convergence of the learning of the convolutional layer is achieved, on average, after 202 samples, which demonstrates that VDSP requires few data for training. The weight distributions after training on 0, 50, and 300 samples are presented: learning with VDSP tends to binary weights.

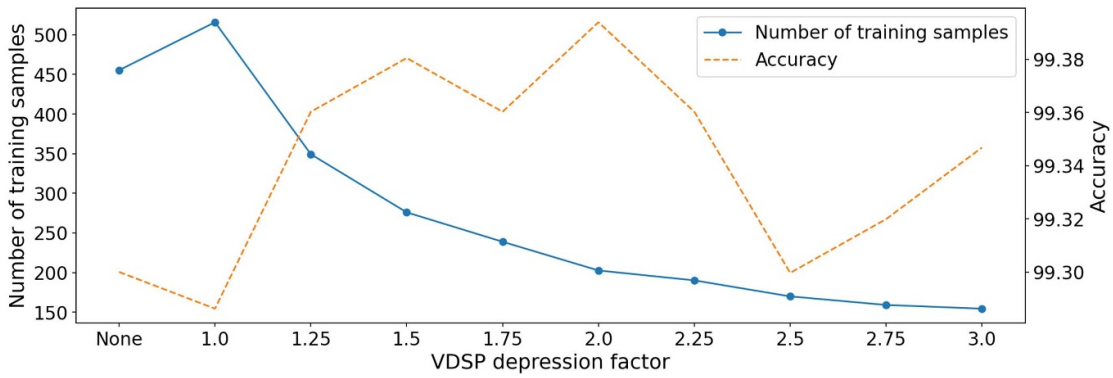

Figure 4. Mean number of training samples used by the CSNN until the learning converges, plotted against different values of the VDSP depression factor f_{dep}. The orange curve shows the accuracy averaged over 10 tries on the test set. When set to 'None', the term $V'_{\text{pre}} - f_{\text{dep}}$ of the VDSP formula is replaced by the constant 1. Increasing f_{dep} significantly speeds up the training while obtaining similar performance. The addition of the depression factor with the term $V'_{\text{pre}} - f_{\text{dep}}$ benefits the training time, as more training samples are needed for $f_{\text{dep}} = \text{None}$.

implementation of pre-trained networks, since the weights could be represented by open-and-closed gates. Note that the accuracy is higher for 50 than 300 training samples as the standard deviation is much higher, which leads the weights to be distributed between $w_{\min}$ and $w_{\max}$, and hence, the class separation by SVM is easier.

To validate the adaptation of VDSP that we introduced in the previous section, we analysed the benefit of the f_{dep} parameter. Figure 4 illustrates the number of training samples used for the convolutional layer to converge, plotted against the chosen value of f_{dep}. When f_{dep} is not specified, the term $V'_{\text{pre}} - f_{\text{dep}}$ of the VDSP formula is replaced by 1, making VDSP similar to an adaptation of STDP proposed in [41]. Without this depression factor, the model needs around 455 samples to converge and achieves an accuracy of 99.3%. Using a value of $f_{\text{dep}} > 1$ considerably reduces the number of training samples needed for convergence, as the connections where the presynaptic neuron has a low membrane potential are depressed more strongly. We present the average over 202 samples needed for $f_{\text{dep}} = 2$, with an accuracy of 99.39%, and 154 samples for $f_{\text{dep}} = 3$, with an accuracy of 99.35%. Hence, these results validate the assumption that presynaptic neurons more likely to fire in a long time can be depressed stronger than presynaptic neurons that are about to fire. Note that f_{dep} must be optimised for the dataset and the desired objective. Also, to facilitate hardware implementation, it may be more convenient to remove the depression term and only use the sign of the membrane potential. As shown in figure 4, it has a minimal impact on the CSNN's performance, but it may make it easier for data to spread throughout hardware.

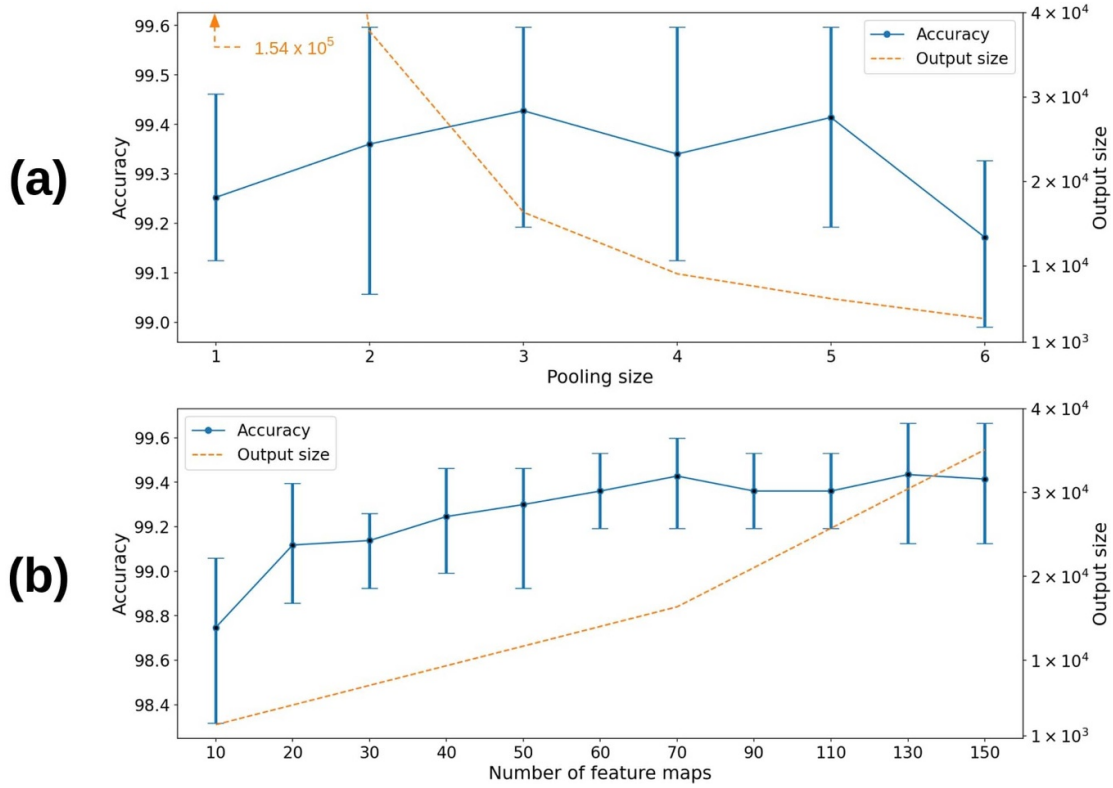

Figure 5. Accuracy averaged over ten tries on the test set vs, in (a), pooling size (kernel and stride sizes), and in (b), the number of feature maps. Orange curves illustrate the output size of the CSNN. Note that no padding is used in the pooling layer. High pooling sizes yield pretty high performance while having a good compression rate. Increasing the number of feature maps improves performance to some extent but also increases output size.

In another experiment, we studied the influence of two hyperparameters of the CSNN: the pooling size, representing both the pooling kernel and stride length, and the number of feature maps. Figure 5 shows the accuracy on the test set averaged over ten tries, plotted against these two parameters. The orange curves indicate the number of output features. Surprisingly, high pooling values can lead to high performance while greatly reducing the size of the output. For instance, an accuracy of 99.41% is achieved for a pooling of 5, compressing the feature maps of the convolutional layer by 96%. However, the best accuracy is obtained for a pooling of 3, with 99.43%, while giving a compression rate of 89%. Second, up to 70 feature maps, as the number of feature maps grows, accuracy improves, but it also increases the number of output features. Note that the number of winners n_{winners} has a slight impact on accuracy but also on training time. Nonetheless, small values help to learn distinct patterns between feature maps. Lastly, depending on the purpose, a bigger pooling size with fewer feature maps and a lower padding size in the convolutional layer could reduce the size of the input and thus transform the CSNN into an efficient encoder.

3.2. Handwritten digits classification with MNIST

MNIST is the standard dataset used in computer vision for benchmarking. It is composed of 28×28 grey-scale images of handwritten digits ranging from 0 to 9. The training set contains 60 000 images and the test set contains 10 000 images. Table 3 compares the accuracy achieved by the CSNN with other methods in the literature. We obtained an average accuracy (over ten tries) of $98.56 \pm 0.05\%$ on the test set, which is comparable to the state of the art performance on SNNs. Again, there were few activations per sample, with a mean of 561 spikes in the network over 61 334 neurons, i.e. around 0.9% activations. The proposed CSNN with a linear SVM also outperforms by 8% the accuracy reported in the original VDSP paper [23], where a one-layer fully connected network is used. However, it is important to note that the author used an unsupervised readout based on spike counts, which is much less complex. Our approach still has other advantages compared to the SNN of the author. First, the use of a convolutional architecture reduces the number of weights by 99%, with 392 000 weights in the SNN against 3430 in the CSNN, which is especially useful for implementation on analogue neuromorphic hardware. This number of weights can be further reduced without significantly harming the performance, for instance, by reducing the number of feature maps. Moreover, TTFS encodes inputs in 15 timesteps only, compared to 100 in the approach with the SNN,

Table 3. Accuracy of proposed CSNN and methods from the literature on MNIST dataset.

Model	Architecture	Learning type	Learning rule	Accuracy (%)
Lee *et al* [10]	CSNN	Supervised	Backpropagation-like	**99.31**
Falez *et al* [22]	CSNN+SVM	Unsupervised	STDP	98.60
Kheradpisheh *et al* [11]	CSNN+SVM	Unsupervised	STDP	98.40
Tavanaei and Maida [20]	SNN	Supervised	BP-STDP	97.20
Kheradpisheh *et al* [43]	BS4NN	Supervised	Backpropagation-like	97.00
Garg *et al* [23]	SNN	Unsupervised	VDSP	90.56
This work	**CSNN+SVM**	**Unsupervised**	**VDSP**	**98.56**

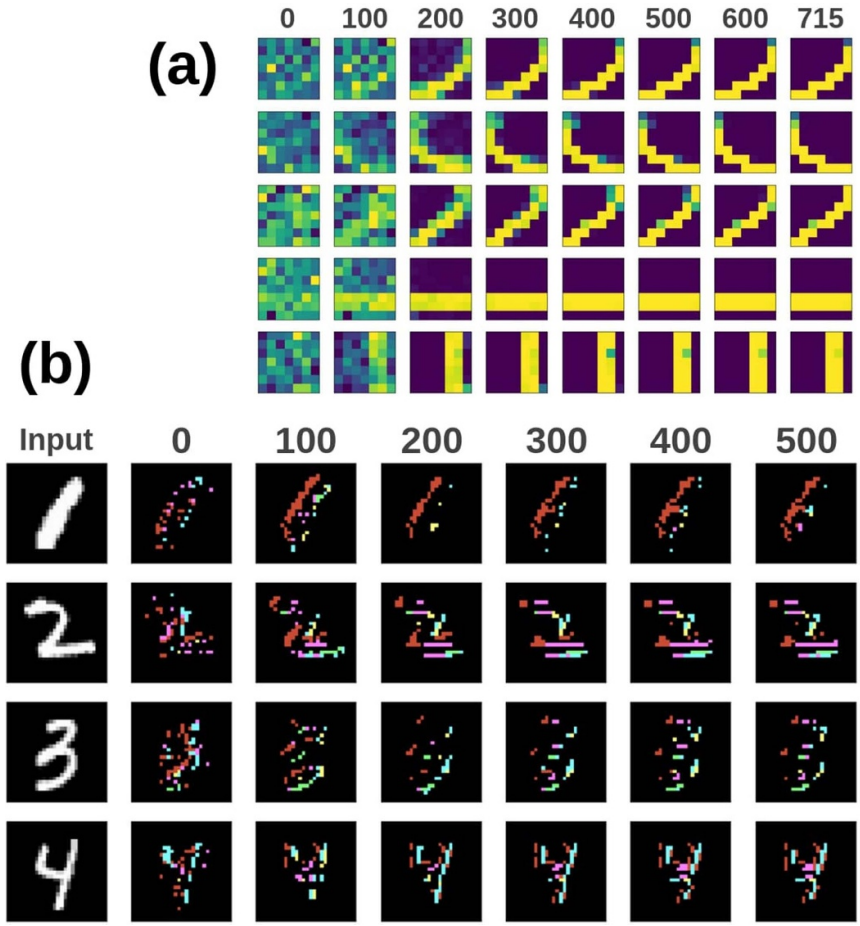

Figure 6. (a) Evolution of various convolution kernels at different steps in the training process. With an adaptive learning rate, learning occurs mainly between 100 and 200 training samples. In addition, kernels become selective to edges. (b) Evolution of output spikes from various feature maps, represented by distinct colours, throughout training. As training continues, the shapes of the digits become more and more visible and feature maps become receptive to different patterns of digits that match trained kernels.

using a rate coding scheme. Hence, the propagation is much faster and also more efficient in terms of energy. However, temporal encoding algorithms are not robust to noise, unlike rate coding ones [42].

To better understand how the convolutional layer learns features, we visualised kernels and feature maps throughout the training of the CSNN. Figure 6 presents the evolution of various convolution kernels and feature maps at different steps of the training process. In (a), we can see that some kernels become selective to edges, enabling the detection of the shapes of the digits. While the learning of the CSNN converges after 715 training samples, we observe only a few changes after 300 samples, meaning the learning is almost finished. Indeed, as the learning rate is adaptive, it reaches $lr_{\text{final}} = 0.1$ after 133 samples, which leads to a faster learning process. In (b), the output spikes of several feature maps are shown with distinct colours. Without training, the outputs of the feature maps are scattered and we can not observe distinct patterns, whereas after training, the feature maps are receptive to distinct patterns that match the trained kernels. For instance, we observe purple horizontal lines for the digit '2', and green diagonal lines for the digit '3'. In addition, the shape of the digit becomes more and more visible as training proceeds.

4. Conclusion

Unsupervised learning in SNNs is usually achieved with STDP [18]. However, STDP has some limitations that makes its hardware implementation difficult. Hence, a novel plasticity rule, called voltage-dependent synaptic plasticity (VDSP) [23], has been developed for the implementation of STDP on memristive-based neuromorphic hardware. This rule is also unsupervised and local, as it uses the membrane potential of the presynaptic neuron (instead of its spike timing in STDP) to evaluate pre/post neurons correlation. However, VDSP is new and so far has only been implemented on a one-layer fully connected network. Further research has to be done to evaluate its scalability and its performance in other network architectures.

In the present paper, we studied for the first time the behaviour of VDSP in a convolutional SNN (CSNN) and its implications with SSIF neurons and TTFS temporal encoding. We developed a WTA based adaptation of VDSP for SSIF neurons, where the estimated spike timing of a presynaptic neuron by its membrane potential is used to evaluate pre/post neurons correlation. Indeed, a high membrane potential, reflecting a neuron about to fire, leads to slow depression whereas a negative membrane potential, reflecting a neuron that is likely to fire only after a long time, leads to a strong depression, which makes training more efficient. Also, the WTA topology used here increases the efficiency of the training and reduces the computational cost. On top of that, we introduced a depression factor in the VDSP formula that may be used to considerably speed up the training, reducing by a factor of two or more the number of training samples needed for the model to converge, with similar performance. VDSP is hardware-friendly, which could make the implementation of the proposed CSNN on memristive-based neuromorphic hardware easier. Note that it could even be useful for large-scale software applications as it removes the need for additional memory to store traces compared with standard STDP. The use of SSIF neurons with TTFS encoding, also facilitated by a lateral inhibition mechanism, makes the CSNN sparsely activated, which is promising for the development of extremely energy-efficient models. We evaluated the proposed CSNN on a computer vision task with MNIST, where it achieved an accuracy of 98.56%, and on a speech recognition task with TIDIGITS, where, helped by our sound preprocessing pipeline, we obtained results better than the state of the art, with an accuracy of 99.43%. We proved that the max-pooling layer is highly efficient at compressing the feature maps while achieving the same performance. Also, we showed that VDSP requires few samples for training, which is useful for small and unlabelled datasets. It also makes the weights tend to binary values, which could facilitate hardware implementation for pre-trained networks. The proposed CSNN with a linear SVM outperforms the fully connected SNN of the original VDSP paper for the MNIST dataset, demonstrating the potential of VDSP implemented in CSNNs with SSIF neurons and TTFS encoding. In addition, when compared to a fully connected SNN, using a CSNN significantly reduces the number of weights, and the use of TTFS decreases the number of timesteps, making both the encoding and the propagation faster and more efficient.

In the future, we will explore supervised learning with spike-based classifiers [21, 44] to replace the SVM used in the readout layer. We are particularly interested in feedback connections, as studied in [44], because they can be used in conjunction with VDSP. We will also study hardware-friendly and energy-efficient preprocessing pipelines. We intend to create an end-to-end SNN solution suitable for neuromorphic hardware implementation.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/ggoupy/CSNN-VDSP.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Author contributions

G G, Y B, A J F, N G, and I B contributed to formulating the study. G G designed the model and implemented it in Python. G G and N G designed the adaptation of VDSP. G G, Y B, A J F, N G, I B defined experiments and G G performed them. G G analysed the data. All authors provided critical feedback and helped shape the research, analysis, and manuscript.

Funding

We acknowledge financial support from the EU: ERC-2017-COG Project IONOS (# GA 773228) and CHIST-ERA UNICO project. This work was also supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) [Funding Reference Number 559730].

Appendix. Parameters

Table 4. Parameters of the CSNN.

	Feature maps	Kernel	Stride	Padding	V_{thr}	n_{winners}	r_{inhib}
Conv	70	7	1	3	10	7	3
Pool	70	3	3	0	1	—	—

Table 5. Parameters of VDSP.

lr_{init}	lr_{final}	lr_{step}	f_{dep}
0.01	0.1	500	2

Table 6. Other parameters.

w_{min}	w_{max}	$w_{\text{init}}^{\text{mean}}$	$w_{\text{init}}^{\text{std}}$	C_{svm}	N_{bins}
0	1	0.8	0.05	0.005	15

ORCID iDs

Gaspard Goupy https://orcid.org/0000-0002-6992-3175
Alexandre Juneau-Fecteau https://orcid.org/0000-0002-0517-5278
Nikhil Garg https://orcid.org/0000-0002-9210-2826
Ismael Balafrej https://orcid.org/0000-0001-6730-0794
Yann Beilliard https://orcid.org/0000-0003-0311-8840

References

[1] LeCun Y, Bengio Y and Hinton G 2015 Deep learning *Nature* **521** 436–44
[2] Li Q, Cai W, Wang X, Zhou Y, Feng D D and Chen M 2014 Medical image classification with convolutional neural network *2014 13th Int. Conf. on Control Automation Robotics & Vision (ICARCV)* pp 844–8
[3] van den Oord A, Kalchbrenner N, Espeholt L, Kavukcuoglu K, Vinyals O and Graves A 2016 Conditional image generation with PixelCNN decoders *Advances in Neural Information Processing Systems* vol 29 (Curran Associates, Inc.)
[4] Badrinarayanan V, Kendall A and Cipolla R 2017 SegNet: a deep convolutional encoder-decoder architecture for image segmentation *IEEE Trans. Pattern Anal. Mach. Intell.* **39** 2481–95
[5] Valenti M, Squartini S, Diment A, Parascandolo G and Virtanen T 2017 A convolutional neural network approach for acoustic scene classification *2017 Int. Joint Conf. on Neural Networks* pp 1547–54
[6] Salamon J and Bello J P 2017 Deep convolutional neural networks and data augmentation for environmental sound classification *IEEE Signal Process. Lett.* **24** 279–83
[7] Duman T B, Bayram B and İnce G 2020 Acoustic anomaly detection using convolutional autoencoders in industrial processes *14th Int. Conf. on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019) (Advances in Intelligent Systems and Computing)* ed, F Martínez Álvarez, A Troncoso Lora, J A Sáez Muñoz, H Quintián and E Corchado (Cham: Springer) pp 432–42
[8] Hayman S 1999 The McCulloch-Pitts model *IJCNN'99. Int. Joint Conf. on Neural Networks Proc. (Cat. No.99CH36339)* vol 6 pp 4438–9
[9] Ghosh-Dastidar S and Adeli H 2009 Spiking neural networks *Int. J. Neural Syst.* **19** 295–308
[10] Lee J H, Delbruck T and Pfeiffer M 2016 Training deep spiking neural networks using backpropagation *Front. Neurosci.* **10** 508
[11] Kheradpisheh S R, Ganjtabesh M, Thorpe S J and Masquelier T 2018 STDP-based spiking deep convolutional neural networks for object recognition *Neural Netw.* **99** 56–67
[12] Zhang L, Zhou S, Zhi T, Du Z and Chen Y 2019 TDSNN: from deep neural networks to deep spike neural networks with temporal-coding *Proc. AAAI Conf. Artificial Intelligence* **33** 1319–26
[13] Lee C, Srinivasan G, Panda P and Roy K 2019 Deep spiking convolutional neural network trained with unsupervised spike-timing-dependent plasticity *IEEE Trans. Cogn. Dev. Syst.* **11** 384–94
[14] Wu Y, Deng L, Li G, Zhu J and Shi L 2018 Spatio-temporal backpropagation for training high-performance spiking neural networks *Front. Neurosci.* **12** 331

[15] Zhang W and Li P 2019 Spike-train level backpropagation for training deep recurrent spiking neural networks *Advances in Neural Information Processing Systems* vol 32 (Curran Associates, Inc.)
[16] Neftci E O, Augustine C, Paul S and Detorakis G 2017 Event-driven random back-propagation: Enabling neuromorphic deep learning machines *Front. Neurosci.* **11** 324
[17] Neftci E O, Mostafa H and Zenke F 2019 Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks *IEEE Signal Process. Mag.* **36** 51–63
[18] Caporale N and Dan Y 2008 Spike timing-dependent plasticity: a Hebbian learning rule *Annu. Rev. Neurosci.* **31** 25–46
[19] Hebb D O 1949 *The Organization of Behavior; A Neuropsychological Theory* (New York: Wiley)
[20] Tavanaei A and Maida A 2018 BP-STDP: approximating backpropagation using spike timing dependent plasticity *Neurocomputing* **330** 39–47
[21] Dong M, Huang X and Xu B 2018 Unsupervised speech recognition through spike-timing-dependent plasticity in a convolutional spiking neural network *PLoS One* **13** e0204596
[22] Falez P, Tirilly P, Marius Bilasco I, Devienne P and Boulet P 2019 Multi-layered spiking neural network with target timestamp threshold adaptation and STDP *2019 Int. Joint Conf. on Neural Networks* pp 1–8
[23] Garg N, Balafrej I, Stewart T C, Portal J-M, Bocquet M, Querlioz D, Drouin D, Rouat J, Beilliard Y and Alibart F 2022 Voltage-dependent synaptic plasticity (VDSP): Unsupervised probabilistic Hebbian plasticity rule based on neurons membrane potential *Front. Neurosci.* **16** 983950
[24] Li Y and Ang K-W 2021 Hardware implementation of neuromorphic computing using large-scale memristor crossbar arrays *Adv. Intell. Syst.* **3** 2000137
[25] Thorpe S, Fize D and Marlot C 1996 Speed of processing in the human visual system *Nature* **381** 520–2
[26] Leonard R G and Doddington G R 1993 TIDIGITS (https://doi.org/10.35111/72XZ-6X59)
[27] LeCun Y, Bottou L, Bengio Y and Haffner P 1998 Gradient-based learning applied to document recognition *Proc. IEEE* **86** 2278–324
[28] Gaspard Goupy 2022 *Source code* CSNN-VDSP (available at: https://github.com/ggoupy/CSNN-VDSP)
[29] Stevens S S and Volkmann J 1940 The relation of pitch to frequency: a revised scale *Am. J. Psychol.* **53** 329–53
[30] Auge D, Hille J, Mueller E and Knoll A 2021 A survey of encoding techniques for signal processing in spiking neural networks *Neural Process. Lett.* **53** 4693–4710
[31] Park S, Kim S, Na B and Yoon S 2020 T2FSNN: deep spiking neural networks with time-to-first-spike coding *2020 57th ACM/IEEE Design Automation Conf. (DAC)* pp 1–6
[32] Kheradpisheh S R and Masquelier T 2020 Temporal backpropagation for spiking neural networks with one spike per neuron *Int. J. Neural Syst.* **30** 2050027
[33] Reich D S, Mechler F and Victor J D 2001 Temporal coding of contrast in primary visual cortex: when, what and why *J. Neurophysiol.* **85** 1039–50
[34] Nelken I, Chechik G, Mrsic-Flogel T D, King A J and Schnupp J W H 2005 Encoding stimulus information by spike numbers and mean response time in primary auditory cortex *J. Comput. Neurosci.* **19** 199–221
[35] Almeida L D, Idiart M and Lisman J E 2009 A second function of gamma frequency oscillations: an E%-max Winner-take-all mechanism selects which cells fire *J. Neurosci.* **29** 7497–503
[36] Shrestha S B and Orchard G 2018 SLAYER: spike layer error reassignment in time *Advances in Neural Information Processing Systems* vol 31 (Curran Associates, Inc.)
[37] Jia S, Zuo R, Zhang T, Liu H and Xu B 2022 Motif-topology and reward-learning improved spiking neural network for efficient multi-sensory integration *ICASSP 2022-2022 IEEE Int. Conf. on Acoustics, Speech and Signal Processing* pp 8917–21
[38] Jia S, Zhang T, Cheng X, Liu H and Xu B 2021 Neuronal-plasticity and reward-propagation improved recurrent spiking neural networks *Front. Neurosci.* **15** 654786
[39] Wu J, Chua Y and Li H 2018 A biologically plausible speech recognition framework based on spiking neural networks *2018 Int. Joint Conf. on Neural Networks* pp 1–8
[40] Zhang T, Jia S, Cheng X and Xu B 2021 Tuning convolutional spiking neural network with biologically plausible reward propagation *IEEE Trans. Neural Netw. Learn. Syst.* 33 1–11
[41] Masquelier T and Thorpe S J 2007 Unsupervised learning of visual features through spike timing dependent plasticity *PLoS Comput. Biol.* **3** e31
[42] London M, Roth A, Beeren L, Häusser M and Latham P E 2010 Sensitivity to perturbations *in vivo* implies high noise and suggests rate coding in cortex *Nature* **466** 123–7
[43] Kheradpisheh S R, Mirsadeghi M and Masquelier T 2022 BS4NN: binarized spiking neural networks with temporal coding and learning *Neural Process. Lett.* **54** 1255–73
[44] Mozafari M, Kheradpisheh S R, Masquelier T, Nowzari-Dalini A and Ganjtabesh M 2018 First-spike-based visual categorization using reward-modulated STDP *IEEE Trans. Neural Netw. Learn. Syst.* **29** 6178–90