Neuronal Competition Groups with Supervised STDP for Spike-Based Classification

Gaspard Goupy[1], Pierre Tirilly[1], and Ioan Marius Bilasco[1,*]

[1]Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
[*]Corresponding author: marius.bilasco@univ-lille.fr

Abstract

Spike Timing-Dependent Plasticity (STDP) is a promising substitute to backpropagation for local training of Spiking Neural Networks (SNNs) on neuromorphic hardware. STDP allows SNNs to address classification tasks by combining unsupervised STDP for feature extraction and supervised STDP for classification. Unsupervised STDP is usually employed with Winner-Takes-All (WTA) competition to learn distinct patterns. However, WTA for supervised STDP classification faces unbalanced competition challenges. In this paper, we propose a method to effectively implement WTA competition in a spiking classification layer employing first-spike coding and supervised STDP training. We introduce the Neuronal Competition Group (NCG), an architecture that improves classification capabilities by promoting the learning of various patterns per class. An NCG is a group of neurons mapped to a specific class, implementing intra-class WTA and a novel competition regulation mechanism based on two-compartment thresholds. We incorporate our proposed architecture into spiking classification layers trained with state-of-the-art supervised STDP rules. On top of two different unsupervised feature extractors, we obtain significant accuracy improvements on image recognition datasets such as CIFAR-10 and CIFAR-100. We show that our competition regulation mechanism is crucial for ensuring balanced competition and improved class separation.

1 Introduction

Neuromorphic computing [1] with Spiking Neural Networks (SNNs) [2] is a promising solution to address the high energy consumption of Artificial Neural Networks (ANNs) on von Neumann architectures [3]. However, direct training of SNNs on neuromorphic hardware faces a major constraint: implementing network-level communication is difficult and requires significant circuitry overhead [4]. As a result, the learning mechanisms should be local, i.e., with weight updates based only on the activity of the two neurons that the synapse connects.

Training SNNs to achieve state-of-the-art (SOTA) performance is typically accomplished with adaptations of backpropagation (BP) [5, 6]. However, these methods are challenging to implement on neuromorphic hardware since they employ non-local learning [4, 7]. In addition, they only rely on supervised learning, making them highly dependent on labeled data. We believe that machine learning algorithms should minimize this dependence on supervision by leveraging unsupervised feature learning [8]. Hence, an appealing classification system may comprise both unsupervised and supervised components, for feature extraction and classification, respectively.

Hebbian learning [9] is an unsupervised and local alternative to BP, inspired by the principal form of plasticity observed in biological synapses. Specifically, Spike Timing-Dependent Plasticity (STDP) [10] is a form of Hebbian learning where the time difference between the input and output neuron spikes defines synaptic plasticity. STDP could solve all the aforementioned limitations of BP, making it more suitable for on-chip training on neuromorphic hardware [11, 12]. STDP is particularly effective with first-spike coding [13, 14], where neurons can fire at most once per sample. Using one spike per neuron presents several advantages, including energy efficiency [15, 16], fast information transfer [17], and high information capacity [18]. While primarily used for unsupervised feature learning [19, 20, 21], STDP can be extented to supervised learning [22, 23, 24]. As a result, SNNs can perform classification tasks by combining unsupervised STDP for feature extraction and supervised STDP for classification [25, 26, 27, 28, 29]. Employing the same type of local learning rule for both feature extraction and classification ensures consistency and may facilitate hardware implementation.

Unsupervised STDP is commonly paired with Winner-Takes-All (WTA) competitive learning to promote the discovery of distinct patterns [30, 19, 31, 20]. In a WTA framework with first-spike coding, lateral inhibition is implemented to ensure that only the first neuron to fire receives a weight update. In addition, homeostatic mechanisms, such as threshold adaptation, must be employed to regulate the competition among neurons [32, 20, 33, 34]. For supervised STDP, WTA competition is also appealing as it may improve the learning capabilities of a classification layer with multiple neurons per class [22]. Specifically, intra-class WTA can promote the learning of various class-specific patterns. However, supervised STDP classification with WTA competition has been poorly studied in the literature [22, 35] and presents unbalanced competition challenges. Indeed, there is a lack of competition regulation methods, and regular threshold adaptation rules can lead to unfair decision-making since output neurons may use different thresholds for inference.

In this paper, we address WTA-based competitive learning in supervised STDP. We aim to implement effective WTA competition in a spiking classification layer employing first-spike coding and SOTA supervised STDP rules. Our main contributions can be summarized as follows:

1. We introduce the Neuronal Competition Group (NCG), an architecture that improves classification capabilities by promoting the learning of various patterns per class. In the classification layer, each class is mapped to an NCG: a group of neurons using intra-class WTA and competition regulation.
2. To ensure both balanced intra-class competition and fair decision-making, we design a competition regulation mechanism based on two-compartment thresholds. Neurons are equipped with a fixed threshold for decision-making, along with an adaptive threshold used to regulate the frequency at which they update their weights on samples of their class.
3. To validate our architecture with input features of varying quality, we incorporate NCGs into spiking classification layers placed on top of two Hebbian-based feature extractors. Using NCGs with SOTA supervised STDP rules, we obtain significant accuracy gains on image recognition datasets: MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100. We show that our competition regulation mechanism is crucial for ensuring balanced competition and improved class separation.

The source code is publicly available at: `https://gitlab.univ-lille.fr/fox/snn-ncg`.

2 Related Work

Supervised Training with STDP Supervised training of SNNs with STDP introduces an error signal [36] that is used to guide the STDP updates. Several supervised adaptations of STDP are reported in the literature [37, 25, 38, 39, 34, 23, 40]. Yet, the aforementioned rules are designed to train SNNs with multiple spikes per neuron, which is not as efficient as first-spike coding. The literature exploring supervised STDP training of SNNs with one spike per neuron is limited [22, 24, 29]. Reward-Modulated STDP (R-STDP) [22] involves supervised training by adjusting the sign of STDP. The employed error is fairly simple ($+1$ or -1), resulting in inaccurate weight updates. SSTDP [24] and S2-STDP [29] are more recent methods that compute temporal errors to adjust both the sign and the intensity of weight updates, making them more accurate. However, unlike R-STDP, these methods cannot be used with various neurons per class in a classification layer.

Competitive Learning for Classification Employing groups of neurons is an effective approach for improving the learning capabilities of a classification layer [41, 42, 43]. To maximize knowledge within the layer and learn distinct patterns, WTA-based competitive learning can be employed [31]. While WTA competition is widely adopted in unsupervised learning [30, 19, 31, 20], its application to supervised learning is limited. Prior work [32, 44, 45] implemented WTA competition at the classification layer but only with one neuron per class, making it impossible to learn various class-specific patterns. Conversely, reward-based approaches [22, 35], such as R-STDP, are the only methods that implement WTA with multiple neurons per class. Through lateral inhibition, neurons compete for weight updates, both within the same class (intra-class WTA) and across different classes (inter-class WTA). Intra-class WTA enables neurons to learn patterns from distinct samples. However, inter-class WTA prevents accurate control over the time difference between the spikes of target and non-target neurons (i.e. neurons mapped or not to the class), as only one neuron is updated per sample. In [29], solely intra-class WTA and two neurons per class were employed to promote specialization toward target and non-target samples. Nonetheless, to the best of our knowledge, no prior work solely employed intra-class WTA to promote the learning of various class-specific patterns.

Competition Regulation In WTA-based competitive learning, it is crucial to implement regulation (also called homeostatic) mechanisms to ensure balanced competition among neurons [31, 20, 33]. A simple solution is to use dropout [46] on the output neurons, as done with R-STDP [22], where some neurons of each class are randomly deactivated during training to encourage weight updates on distinct samples. Yet, this solution is not optimal due to its stochastic nature. Other regulation mechanisms involve threshold adaptation [32, 20, 34], by increasing or reducing thresholds to promote or discourage firing. While threshold adaptation is an effective solution to ensure balanced competition, using different thresholds across neurons may prevent fair decision-making since their firing time is tied to their thresholds. Prior work employed multiple thresholds per neuron [27, 47] but the authors did not incorporate threshold adaptation mechanisms. In this work, we draw inspiration from multi-thresholds and threshold adaptation to design a competition regulation mechanism based on two-compartment thresholds, ensuring both balanced competition and fair decision-making.

3 Preliminaries

3.1 Neuron Model

To align with first-spike coding, we use the Single-Spike Integrate-and-Fire (SSIF) model [48], where neurons can fire at most once per sample. Since each neuron emits a single spike, the intensity of its activation is encoded via a firing timestamp: the most activated neuron fires first. The membrane potential V_j of a neuron n_j is expressed as:

$$\begin{aligned} \frac{\partial V_j\left(t\right)}{\partial t} &= \sum_i W_{ij} \cdot S_i\left(t\right) \\ S_i\left(t\right) &= \begin{cases} 1 & \text{if } V_i\left(t\right) \geq \theta \\ 0 & \text{o.w.} \end{cases}, \end{aligned} \tag{1}$$

where t is the timestamp, $S_i\left(t\right)$ indicates the presence or absence of a spike from input neuron n_i at timestamp t, and W_{ij} is the weight of the synapse from n_i to n_j. When the membrane potential of a neuron reaches its firing threshold θ, the neuron emits a spike, resets its membrane potential to zero, and remains deactivated until the next sample is shown. In our simulations, firing timestamps are represented by floating-point values to align with event-driven neuromorphic hardware.

3.2 Spiking Classification Layer

The spiking classification layer is a fully-connected architecture comprising, for a C-class problem, $N = C \times M$ neurons $\left(n_1, \ldots, n_N\right)$, where M is the number of neurons per class. Each neuron n_j is mapped to a class c_j. Aligned with the SSIF model, we employ first-spike-based decision-making: the first output neuron to fire predicts the class. This method removes the need to propagate the entire input for inference, which can reduce computation time and the number of generated spikes. Formally, the prediction $\hat{y}$ of the SNN is defined as:

$$\begin{aligned} \hat{y} &= c_{j^*} \\ j^* &= \underset{j \in [1,N]}{\operatorname{argmin}} \left(t_j\right), \end{aligned} \tag{2}$$

where t_j denotes the firing timestamp of neuron n_j. If multiple neurons fire at the same timestamp, the one with the highest membrane potential is selected. In practice, the method employed to select a neuron in the event of a tie has little effect on performance. In this work, the classification layer is placed on top of an unsupervised feature extraction network.

3.3 Supervised STDP Training

Neurons of the classification layer are trained with a supervised STDP rule. At the end of a sample presentation, weights of non-inhibited neurons are updated with an error-modulated additive STDP:

$$\Delta W_{ij} = \begin{cases} e_j \times A^+ & \text{if } t_j \geq t_i \\ e_j \times A^- & \text{o.w.} \end{cases}, \tag{3}$$

where ΔW_{ij} is the weight change (such as $W_{ij} := W_{ij} + \Delta W_{ij}$), e_j is the error of neuron n_j, $A^+ > 0$ and $A^- < 0$ are the learning rates. These two learning rates control learning speed and determine the relative importance of long-term potentiation (A^+) versus long-term depression (A^-) in the learning process. Weights are manually clipped in $[w_{\min}, w_{\max}]$ after each update to ensure that they remain within a controlled range.

R-STDP Reward-Modulated STDP (R-STDP) [22] is a rule combined with WTA competition. For each sample, only the first neuron to fire receives a weight update. The error is $e_j = +1$ if n_j is mapped to the class of the sample, $e_j = -1$ otherwise. In practice, a classification layer trained with R-STDP requires multiple neurons per class to achieve reasonable performance. R-STDP is usually employed in conjunction with adaptive learning rates to reduce overfitting, and dropout to facilitate the learning of various patterns per class [22].

SSTDP Supervised STDP (SSTDP) [24] is a rule with SOTA performance. It is employed with one neuron per class and without WTA. This rule provides high adaptability to input data by dynamically computing temporal errors for each sample, based on the average firing time $\overline{T}$ in the layer:

$$e_j = t_j - \begin{cases} \min\left\{t_j, \overline{T} - \frac{C-1}{C} g\right\} & \text{if } c_j = y \\ \max\left\{t_j, \overline{T} + \frac{1}{C} g\right\} & \text{if } c_j \neq y \end{cases}, \tag{4}$$

where y is the class of the sample, and g is a hyperparameter that controls the desired distance from $\overline{T}$. The optimal value of g partly depends on the input spike distribution: a narrower distribution requires a smaller g. For each sample, due to the $\min$ and $\max$ functions, only the target neuron firing after $\overline{T} - \frac{C-1}{C} g$ and the non-target neurons firing before $\overline{T} + \frac{1}{C} g$ update their weights.

S2-STDP Stabilized Supervised STDP (S2-STDP) [29] addresses two limitations of SSTDP: the limited number of updates per epoch and the saturation of firing timestamps toward the maximum firing time. In this rule, neurons are trained to fire at desired timestamps instead of time ranges:

$$e_j = t_j - \begin{cases} \overline{T} - \frac{C-1}{C} g & \text{if } c_j = y \\ \overline{T} + \frac{1}{C} g & \text{if } c_j \neq y \end{cases}. \tag{5}$$

This enables more accurate control over the output firing times and reduces the saturation effect. Also, weight normalization is used to keep a similar weight average across neurons during learning [29].

4 Methods

4.1 Neuronal Competition Group

Training a classification layer involves teaching neurons to recognize a pattern specific to their class from the input samples. Different samples from a given class can contain distinct, mutually exclusive patterns or combinations of patterns. Learning all these patterns concurrently with one neuron can be challenging and impose strong generalization constraints on its weights, especially when using a single supervised layer. Employing multiple neurons per class to learn various class-specific patterns may reduce these constraints and enable the emergence of more specialized patterns that better represent the training set distribution. Building on this concept, we introduce the Neuronal Competition Group (NCG), an architecture promoting the learning of various class-specific patterns through intra-class WTA and competition regulation.

The NCG architecture, illustrated in Figure 1, augments a classification layer by mapping each class to an NCG instead of independent neurons. An NCG is a group of M neurons that aim to learn different patterns from samples of their mapped class. Neurons of an NCG are interconnected with

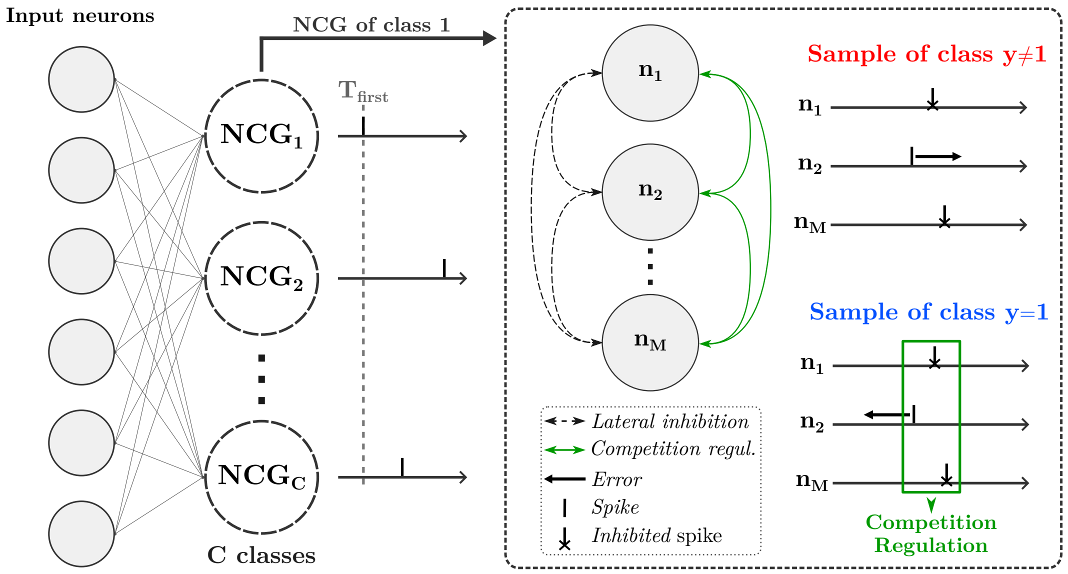


Figure 1: Spiking classification layer with Neuronal Competition Groups (NCGs). In this layer, each class is mapped to an NCG and the prediction is based on the first spike. An NCG is a group of M neurons connected with lateral inhibition to enable intra-class WTA competition: the first neuron to fire inhibits the other ones and undergoes a weight update based on a temporal error (which depends on the learning rule considered). The sign and amplitude of the error pushes neurons to fire earlier (positive sign) or later (negative sign). Competition regulation occurs only within the NCG mapped to the class of the input sample to ensure balanced competition among neurons on samples of their class. NCGs improve the classification capabilities of a layer by promoting the learning of various patterns per class.

lateral inhibition, such as, for a given sample, the first neuron to fire within a group emits an inhibitory signal that prevents the other ones from firing. Lateral inhibition induces competitive learning through intra-class WTA: only the first neuron to fire undergoes the weight update. There is no lateral inhibition between NCGs (i.e. inter-class WTA). Hence, each sample triggers exactly one weight update per NCG. Removing inter-class WTA enables more accurate control over the time difference between the spikes of target and non-target neurons, which can improve class separation [29]. Each time a sample is presented during training, competition regulation is triggered in the NCG mapped to the class of the sample. This mechanism ensures balanced competition within the NCGs, which facilitate the learning of various class-specific patterns.

4.2 Competition Regulation

Preliminary experiments highlighted that intra-class WTA competition provided by lateral inhibition is not enough to ensure balanced competition. In practice, for each NCG, one neuron tends to dominate the others, receiving the majority of the weight updates from samples of its class. Although threshold adaptation can be employed to regulate competition [32, 20], in a decision-making context, different thresholds between neurons may lead to unfair decisions because predictions are based on the first spike. To ensure both balanced intra-class competition and fair decision-making, we introduce a competition regulation mechanism based on two-compartment thresholds.

In the classification layer, all the neurons are equipped with an identical and fixed threshold, denoted as the test threshold θ. This threshold remains fixed to ensure fair decision-making during inference, as the class is predicted by the neuron that fires first. On top of that, neurons are equipped with an additional varying threshold, denoted as the training threshold θ'. Neurons switch to their θ' only when they are exposed to samples of their class during training. Otherwise, they always employ θ, both for inference and for samples of other classes during training. θ' is the key component to balance intra-class competition: it can be increased or decreased to encourage or reduce neuron firing on samples of its class. Each time a neuron receives a weight update from a sample of its class, competition regulation is triggered across neurons of its NCG. Their θ' are updated as follows:

$$\begin{aligned} \Delta\theta'_j &= \begin{cases} +\eta_{\text{th}} \cdot \frac{M-1}{M} & \text{if } t_j = \min\{t_1, \cdots, t_M\} \\ -\eta_{\text{th}} \cdot \frac{1}{M} & \text{o.w.} \end{cases} \\ \theta'_j &:= \max\left\{\theta_j, \theta'_j + \Delta\theta'_j\right\}, \end{aligned} \tag{6}$$

where θ'_j and θ_j are the training and test thresholds of neuron n_j, M is the number of neurons in the NCG, η_{th} is the threshold learning rate, and t_j is the firing timestamp of neuron n_j. If several neurons fire at the same timestamp, the one with the highest membrane potential is selected (this has no impact on performance). θ' is reset to θ between epochs and its minimum achievable value is θ. These two components ensure that neurons learn patterns consistent with θ, which is the threshold that they use for inference. η_{th} defines the strength of competition regulation: higher values favor more balanced competition but may deteriorate pattern learning since θ' tend to increase within an epoch. It should be chosen together with the initial threshold (a higher threshold may require a higher η_{th}). To achieve better convergence and robustness, an annealing factor β_{th} can be added to reduce η_{th} after each epoch, such as $\eta_{\text{th}} := \eta_{\text{th}} \cdot \beta_{\text{th}}$. β_{th} affects the number of epochs during which competition regulation occurs and should be adjusted according to η_{th}: higher η_{th} requires lower β_{th}.

4.3 Neuron Labeling

In [29], WTA competition enhances a classification layer with two neurons per class and S2-STDP training by naturally promoting, for each class, neuron specialization toward target or non-target samples. This behavior can also be implemented with NCGs, but it requires explicit neuron labeling to ensure that all neurons but one specialize toward samples of their class. In such cases, one neuron within each NCG can be labeled as non-target, whereas the others can be labeled as target. All the neurons are connected with lateral inhibition but only target neurons are connected with competition regulation. Hence, if the non-target neuron fires first for a sample of the class, it prevents target neurons from updating their weights and applying competition regulation. Regardless of the class of the sample, a non-target neuron n_j winning the competition always updates its weights as if $c_j \neq y$ in Equation 5. STDP training remains unchanged for target neurons. However, in Equation 6, M must be updated as it refers to the number of target neurons. In Supplementary Material (Section 1), we provide the overall algorithm for training a spiking classification layer with our proposed methods.

5 Experiments

5.1 Experimental Setup

5.1.1 Datasets

We select four image recognition datasets of growing complexity: MNIST [49], Fashion-MNIST [50], CIFAR-10 [51], and CIFAR-100 [51]. MNIST and Fashion-MNIST comprise 28×28 grayscale images, $60,000$ samples for training and $10,000$ for testing, categorized into 10 classes. CIFAR-10 and CIFAR-100 comprise 32×32 RGB images, $50,000$ for training and $10,000$ for testing. They consist of, respectively, 10 and 100 classes.

5.1.2 Classification Pipeline

Our classification system consist of a feature extractor trained with unsupervised Hebbian-based learning, followed by a spiking classification layer trained with supervised STDP. Training is layer-wise: the feature extractor is trained entirely before the training of the classification layer starts. The complete pipeline of our classification system is illustrated in Supplementary Material (Section 2.1).

5.1.3 Unsupervised Feature Extractors

To improve image representation before classification without labeled data, we consider two Hebbian-based unsupervised feature extractors built on Convolutional Neural Networks (CNNs):

1. STDP-CSNN [20], a single-layer spiking CNN trained with STDP;
2. SoftHebb-CNN [52], a three-layer non-spiking CNN trained with SoftHebb.

Employing various feature extractors allows us to validate our methods with input features of varying quality. These two feature extractors are SOTA in their category (spiking/non-spiking), share local learning properties, and offer different baseline performances. In particular, SoftHebb-CNN, while not spike-based, is a relevant alternative for exploring classification using features provided by effective multi-layer local learning. The extracted feature maps are flattened to match the fully-connected architecture of the classification layer. Since SoftHebb-CNN is not spike-based, its output features are encoded into spike timestamps with a form of first-spike coding [53]. STDP-CSNN outputs $4,608$ features for MNIST/Fashion-MNIST, and $6,272$ for CIFAR-10/100. SoftHebb-CNN outputs $13,824$ features for MNIST/Fashion-MNIST, and $24,576$ for CIFAR-10/100. Aligned with first-spike coding, each feature is a single floating-point spike timestamp in $[0,1]$. Additional details are reported in Supplementary Material (Section 2.2).

5.1.4 Spiking Classification Layers

We train fully-connected spiking classification layers with three existing supervised STDP rules designed for one spike per neuron: R-STDP [22], SSTDP [24], and S2-STDP [29]. We incorporate the NCG architecture into classification layers trained with SSTDP and S2-STDP, denoted as SSTDP+NCG and S2-STDP+NCG, respectively. R-STDP is incompatible with NCGs since it requires inter-class WTA for weight convergence. Unless otherwise specified, we set $M = 5$ neurons per class for NCG-based methods, which is the smallest value providing, on average, near-optimal performance on the evaluated datasets (see Section 3.2 of Supplementary Material). We evaluated R-STDP with both $M = 5$ and $M = 20$, the value providing near-optimal performance for this rule. With S2-STDP+NCG, one neuron of each NCG is labeled as non-target, as detailed in Section 4.3.

5.1.5 Protocol

We divide our experimental protocol into two phases: hyperparameter optimization and evaluation. In both phases, we employ an early stopping mechanism (with a patience ρ) during training to prevent overfitting. For hyperparameter optimization, we construct a validation set from the training set by randomly selecting, for each class, a percentage ν of its samples. Then, we use the gridsearch algorithm to optimize the hyperparameters of the spiking classification layer (for each rule, dataset, and feature extractor). No gridsearch is performed on CIFAR-100: we employ the optimized hyperparameters from CIFAR-10, given the similarities between the two datasets. Additional details regarding hyperparameters are provided in Supplementary Material (Section 2.3). For evaluation, we employ the K-fold cross-validation strategy. We divide the training set into K subsets and train K models, each using a different subset for validation while the remaining $K - 1$ subsets are used for training. Each model is trained with a different seed. Then, we evaluate the trained models on the test set and we compute the mean test accuracy and standard deviation (1-sigma). We use $\rho = 10$, $K = 10$ and $\nu = \frac{1}{K}$ (i.e. we allocate 10% of the training sets for validation).

5.2 Accuracy Comparison

We compare, in Table 1, the performance of the different STDP-based methods for training a spiking classification layer (see Section 5.1.4) placed on top of each unsupervised feature extractor (see Section 5.1.3). Our proposed NCG architecture consistently improves the performance of SSTDP and S2-STDP across all datasets and feature extractors. The accuracy improvement tends to scale with the complexity of the dataset. With S2-STDP and the STDP-CSNN feature extractor, we measure an increase of 1.18 pp on MNIST, 2.83 pp on Fashion-MNIST, 5.33 pp on CIFAR-10, and 6.51 pp on CIFAR-100. S2-STDP always outperforms SSTDP and enables higher accuracy improvement when paired with NCG as it leverages neuron labeling. While S2-STDP surpasses R-STDP when the input features are well-captured, it falls behind in scenarios involving lower-quality features (CIFAR-10 with STDP-CSNN, CIFAR-100), as R-STDP can learn various patterns per class. S2-STDP+NCG effectively bridges this gap, outperforming R-STDP on both simpler and harder tasks while requiring four times fewer neurons per class. When R-STDP is used with the same number of neurons as S2-STDP+NCG, the accuracy gap is even larger. These results highlight that WTA-based competitive learning in a supervised context can be achieved without inter-class WTA. Employing solely intra-class WTA and accurate STDP updates enables more effective training.

Regarding the literature on SNNs with fully-supervised local-based learning, SOTA performance is achieved by STiDi-BP (one spike per neuron) [54] on MNIST (99.20% with a 3-layer SNN) as well

Table 1: Accuracy of spiking classification layers trained with STDP-based methods, on top of Hebbian-based unsupervised feature extractors.

Dataset	Method	Neurons per class	Accuracy (Mean±Std %) STDP-CSNN	Accuracy (Mean±Std %) SoftHebb-CNN
MNIST	R-STDP	5	96.82 ± 0.29	97.72 ± 0.26
		20	97.49 ± 0.12	98.24 ± 0.15
	SSTDP	1	96.44 ± 0.09	98.52 ± 0.16
	SSTDP+NCG *(ours)*	5	97.30 ± 0.09	98.96 ± 0.06
	S2-STDP	1	97.74 ± 0.06	98.81 ± 0.09
	S2-STDP+NCG *(ours)*	5	**98.92** $\pm$ **0.07**	**99.17** $\pm$ **0.07**
Fashion-MNIST	R-STDP	5	78.40 ± 0.89	87.32 ± 0.76
		20	82.17 ± 0.38	88.06 ± 0.29
	SSTDP	1	85.26 ± 0.17	89.36 ± 0.24
	SSTDP+NCG *(ours)*	5	87.59 ± 0.11	91.06 ± 0.10
	S2-STDP	1	85.89 ± 0.27	90.61 ± 0.19
	S2-STDP+NCG *(ours)*	5	**88.72** $\pm$ **0.23**	**91.86** $\pm$ **0.14**
CIFAR-10	R-STDP	5	62.12 ± 0.62	74.12 ± 0.34
		20	65.92 ± 0.68	75.54 ± 0.57
	SSTDP	1	60.87 ± 0.53	76.57 ± 0.58
	SSTDP+NCG *(ours)*	5	64.05 ± 0.48	78.53 ± 0.32
	S2-STDP	1	61.08 ± 0.17	76.90 ± 0.27
	S2-STDP+NCG *(ours)*	5	**66.41** $\pm$ **0.17**	**79.55** $\pm$ **0.23**
CIFAR-100	R-STDP	5	32.07 ± 0.38	48.27 ± 0.36
		20	34.77 ± 0.44	49.25 ± 0.48
	SSTDP	1	28.49 ± 0.49	48.73 ± 0.39
	SSTDP+NCG *(ours)*	5	31.19 ± 0.27	49.81 ± 0.23
	S2-STDP	1	29.39 ± 0.19	49.17 ± 0.29
	S2-STDP+NCG *(ours)*	5	**35.90** $\pm$ **0.42**	**53.49** $\pm$ **0.18**

as Fashion-MNIST (92.80% with a 4-layer SNN), and by EMSTDP (multiple spikes per neuron) [55] on CIFAR-10 (64.40% with a 4-layer SNN). We did not find any work reporting results on CIFAR-100. For approaches combining unsupervised and supervised local learning, SOTA performance is achieved by R-STDP [26] on MNIST (97.20% with a 3-layer SNN) and by Sym-STDP [34] on Fashion-MNIST (85.31% with a 2-layer SNN). We did not find any work reporting results on CIFAR-10/100. Our best models, comprising 4-layer networks with only one supervised layer, achieve 99.17% on MNIST, 91.86% on Fashion-MNIST, and 79.55% on CIFAR-10. Our results closely match or surpass fully-supervised SOTA work and outperform semi-supervised SOTA work. Yet, it is important to acknowledge the role of the feature extractor in the final performance. There remains a huge gap between local-based and global-based approaches in terms of accuracy. In Supplementary Material (Section 4), we compare our methods with global-based approaches to highlight that, despite the accuracy gap, local-based methods show greater computational efficiency, lower memory usage, reduced energy consumption, and easier hardware implementation, justifying further exploration.

5.3 Ablation Study

We conduct, in Table 2, an ablation study on S2-STDP+NCG to evaluate each component of our methods. *M-1* and *M-5* represent S2-STDP+NCG with $M = 1$ (one neuron per class, which is similar to S2-STDP) and $M = 5$, without competition regulation and neuron labeling. *CR-1* denotes our competition regulation mechanism with a single threshold per neuron (i.e. $\theta' = \theta$ in Equation 6),

Table 2: Ablation study on S2-STDP+NCG. *M* is the number of neurons per class, *CR* is competition regulation with 1 or 2 thresholds, *L* is neuron labeling, and *Drop* is dropout.

(a) Fashion-MNIST

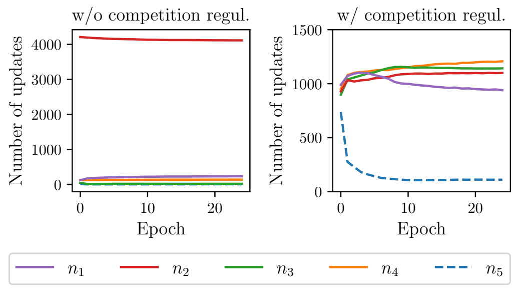
Method	Accuracy (Mean±Std %) STDP-CSNN	Accuracy (Mean±Std %) SoftHebb-CNN
M-1	85.89 ± 0.27	90.61 ± 0.19
M-5	86.74 ± 0.25	91.25 ± 0.20
M-5+CR-1	86.77 ± 0.22	85.16 ± 5.34
M-5+CR-2	87.76 ± 0.16	91.33 ± 0.22
M-5+CR-1+L	87.14 ± 0.41	89.24 ± 0.89
M-5+CR-2+L	**88.72 ± 0.23**	**91.86 ± 0.14**
M-5+Drop+L	87.33 ± 0.20	91.34 ± 0.08

(b) CIFAR-10

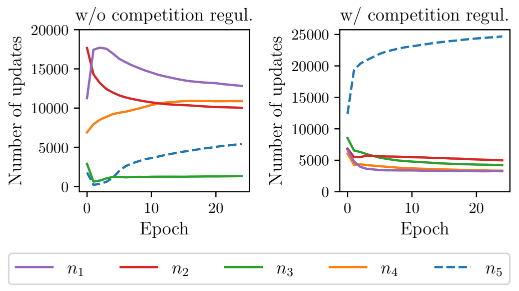
Method	Accuracy (Mean±Std %) STDP-CSNN	Accuracy (Mean±Std %) SoftHebb-CNN
M-1	61.08 ± 0.17	76.90 ± 0.27
M-5	62.61 ± 0.27	78.29 ± 0.27
M-5+CR-1	64.73 ± 0.41	77.35 ± 0.22
M-5+CR-2	65.51 ± 0.26	78.78 ± 0.15
M-5+CR-1+L	65.46 ± 0.40	78.67 ± 0.19
M-5+CR-2+L	**66.41 ± 0.17**	**79.55 ± 0.23**
M-5+Drop+L	63.15 ± 0.11	77.98 ± 0.21

as commonly used in WTA-based SNNs [32, 13]. Thresholds are not clipped nor reset between epochs, and the learned values are used for inference. *CR-2* denotes our competition regulation with two-compartment thresholds. *L* is neuron labeling. *Drop* is dropout on the output neurons, an alternative competition regulation mechanism employed with R-STDP [22]. For each method, we optimized hyperparameters with gridsearch (see Section 5.1.5). Results on both Fashion-MNIST and CIFAR-10 show that each component of our methods (cf. *M-5*, *CR-2*, *L*) brings an individual and significant accuracy gain. Competition regulation tends to be crucial for benefiting from improved class separation, especially with STDP-CSNN. The accuracy gain gets lower with SoftHebb-CNN as the extracted features exhibit higher class separability. Neuron labeling enhances the performance of our models through neuron specialization. Our competition regulation mechanism based on two-compartment thresholds (cf. *CR-2*) outperforms the existing threshold adaptation with one threshold (cf. *CR-1*), as well as dropout (cf. *Drop*). In a first-spike-based decision-making context, we find that learning thresholds (cf. *CR-1*) is not mandatory for successfully learning various patterns. Instead, it is more important to ensure fair decision-making with fixed thresholds and use threshold adaptation as a competition regulation mechanism only. In Supplementary Material, we provide an ablation study on SSTDP+NCG with similar results (Section 3.4), as well as additional studies on the impact of neuron labeling (Section 3.1) and hyperparameters (Section 3.3).

5.4 Impact of Competition Regulation



(a) Number of target updates



(b) Number of non-target updates

Figure 2: Number of weight updates per epoch received by the neurons of class 0 trained with S2-STDP+NCG, with and without competition regulation, on CIFAR-10. n_1 to n_4 are labeled as target neurons and n_5 is labeled as non-target. The features are extracted with STDP-CSNN.

In this section, we show that competition regulation is crucial for ensuring balanced competition and improved class separation. Figure 2 illustrates the number of updates per epoch received by the neurons of class 0 trained with S2-STDP+NCG, with and without competition regulation, on CIFAR-10. Target (resp. non-target) updates are triggered by samples of the class (resp. another class). Without competition regulation, no competition takes place between target neurons. Target neuron n_2 receives the majority of the target updates, while the other target neurons n_1, n_3, n_4 assume the role of non-target neurons (i.e. receive mainly non-target updates), as they are inhibited by n_2 on samples of the class. With competition regulation, the target neurons (n_1 to n_4) effectively specialize toward samples of their class, while the non-target neuron (n_5) specializes toward samples of other classes. Regarding target neurons, we observe a balanced competition in their target updates, illustrating the effectiveness of our competition regulation mechanism. In Supplementary Material (Section 3.5), we show similar results for other classes and datasets, as well as for SSTDP+NCG.

In another experiment, we analyze the weights trained using S2-STDP+NCG, with and without competition regulation. Figure 3 shows t-SNE [56] visualizations of the learned weights on CIFAR-10. Without competition regulation, there is a single cluster at the center, comprising the weights of the neurons that receive few target updates during training. With competition regulation, the weights of the target neurons tend to form, for each class, distinct clusters. The spread of their clusters suggests that they have learned various class-specific patterns. The weights of the non-target neurons form a single cluster at the center since their weights are very similar. This similarity arises because non-target neurons, regardless of their class, are trained to fire at the same desired timestamp. In Supplementary Material (Section 3.5), we further show that competition regulation increases the intra-class distinctiveness among weights, which improves class separation.

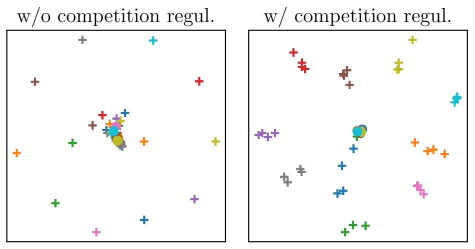


Figure 3: t-SNE plots of the weights learned with S2-STDP+NCG on CIFAR-10, with and without competition regulation. Crosses and circles respectively represent the weights of target and non-target neurons, and colors indicate classes. The features are extracted with STDP-CSNN.

6 Discussion

The NCG architecture implements effective intra-class WTA in a spiking classification layer employing first-spike coding and supervised STDP training. Our competition regulation mechanism based on two-compartment thresholds ensures both balanced competition and fair decision-making. We showed that this mechanism improves class separation and achieves better performance than existing regulation methods. As a result, NCGs significantly increased the accuracy of SOTA supervised STDP rules. This work highlights that more effective supervised competitive learning can be achieved without inter-class WTA. Also, the success in learning various patterns per class via threshold adaptation does not depend on learning various thresholds.

In this work, supervised STDP rules are employed in the classification layer to ensure consistency with the training of the feature extraction network. However, our contributions focus on the architecture of the classification layer, which is independent of the learning rule used to train it. Thus, NCGs may theoretically be used with any other rule designed for training SNNs with one spike per neuron. We performed preliminary experiments with S4NN [57], a gradient-based rule, and observed that the addition of NCGs led to accuracy improvements consistent with our previous results (see Section 3.6 of Supplementary Material). Yet, further research is required to validate the effectiveness of NCGs with gradient-based rules.

While NCGs successfully improve the performance of a classification layer, they also come with several limitations. First, they increase the costs in terms of parameters, computation, and hardware. The computational overhead of NCGs scales linearly with the number of neurons. In hardware design, they introduce another overhead due to the additional connections, both to the previous layer and within the layer. Second, increasing the number of neurons strengthens specialization on the training set, especially when faced with a higher number of input features (cf. SoftHebb-CNN). This behavior limits the generalization capabilities of our models and requires additional research to fully exploit their potential in these scenarios. Third, the NCG architecture applies only to the output layer of a network. Nevertheless, this work is the first to introduce WTA and competition regulation mechanisms specifically designed for classification. It establishes the relevance of such mechanisms in this context, laying the foundations for future research on WTA-based supervised competitive learning in multi-layer networks.

Acknowledgements

This work is funded by Chaire Luxant-ANVI (Métropole de Lille) and supported by IRCICA (CNRS UAR 3380). Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see `https://www.grid5000.fr`).

References

[1] Catherine D. Schuman, Thomas E. Potok, Robert M. Patton, J. Douglas Birdwell, Mark E. Dean, Garrett S. Rose, and James S. Plank. A Survey of Neuromorphic Computing and Neural Networks in Hardware. *ArXiv*, arXiv:1705.06963 [cs.NE], 2017.

[2] Filip Ponulak and Andrzej Kasinski. Introduction to Spiking Neural Networks: Information Processing, Learning and Applications. *Acta Neurobiologiae Experimentalis*, 71:409–433, 2011.

[3] Xingqi Zou, Sheng Xu, Xiaoming Chen, Liang Yan, and Yinhe Han. Breaking the Von Neumann Bottleneck: Architecture-Level Processing-in-Memory Technology. *Science China Information Sciences*, 64, 2021.

[4] Friedemann Zenke and Emre Neftci. Brain-Inspired Learning on Neuromorphic Substrates. *Proceedings of the IEEE*, 109:935–950, 2021.

[5] Jason K. Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training Spiking Neural Networks Using Lessons From Deep Learning. *ArXiv*, arXiv:2109.12894 [cs.NE], 2021.

[6] Manon Dampfhoffer, Thomas Mesquida, Alexandre Valentian, and Lorena Anghel. Backpropagation-Based Learning Techniques for Deep Spiking Neural Networks: A Survey. *Transactions on Neural Networks and Learning Systems*, 2023.

[7] Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. Backpropagation and the Brain. *Nature Reviews Neuroscience*, 21:335–346, 2020.

[8] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. *Transactions on Pattern Analysis and Machine Intelligence*, 35:1798–1828, 2013.

[9] Donald Hebb. *The Organization of Behavior*. Springer, Berlin, Heidelberg, 1949.

[10] Natalia Caporale and Yang Dan. Spike Timing–Dependent Plasticity: A Hebbian Learning Rule. *Annual Review of Neuroscience*, 31:25–46, 2008.

[11] Sylvain Saïghi, Christian G. Mayr, Teresa Serrano-Gotarredona, Heidemarie Schmidt, Gwendal Lecerf, Jean Tomas, Julie Grollier, Sören Boyn, Adrien F. Vincent, Damien Querlioz, Selina La Barbera, Fabien Alibart, Dominique Vuillaume, Olivier Bichler, Christian Gamrat, and Bernabé Linares-Barranco. Plasticity in Memristive Devices for Spiking Neural Networks. *Frontiers in Neuroscience*, 9, 2015.

[12] Lyes Khacef, Philipp Klein, Matteo Cartiglia, Arianna Rubino, Giacomo Indiveri, and Elisabetta Chicca. Spike-Based Local Synaptic Plasticity: A Survey of Computational Models and Neuromorphic Circuits. *Neuromorphic Computing and Engineering*, 3, 2023.

[13] Pierre Falez. *Improving Spiking Neural Networks Trained with Spike Timing Dependent Plasticity for Image Recognition*. PhD thesis, Université de Lille, 2019.

[14] Wenzhe Guo, Mohammed E. Fouda, Ahmed M. Eltawil, and Khaled Nabil Salama. Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems. *Frontiers in Neuroscience*, 15, 2021.

[15] Bodo Rueckauer and Shih-Chii Liu. Conversion of Analog to Spiking Neural Networks Using Sparse Temporal Coding. In *International Symposium on Circuits and Systems*, 2018.

[16] Seongsik Park, Seijoon Kim, Byunggook Na, and Sungroh Yoon. T2FSNN: Deep Spiking Neural Networks with Time-to-First-Spike Coding. In *Design Automation Conference*, 2020.

[17] Rufin Van Rullen and Simon J. Thorpe. Rate Coding Versus Temporal Order Coding: What the Retinal Ganglion Cells Tell the Visual Cortex. *Neural Computation*, 13:1255–1283, 2001.

[18] Daniel Auge, Julian Hille, Etienne Mueller, and Alois Knoll. A Survey of Encoding Techniques for Signal Processing in Spiking Neural Networks. *Neural Processing Letters*, 53:4693–4710, 2021.

[19] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J. Thorpe, and Timothée Masquelier. STDP-Based Spiking Deep Convolutional Neural Networks for Object Recognition. *Neural Networks*, 99:56–67, 2018.

[20] Pierre Falez, Pierre Tirilly, Ioan Marius Bilasco, Philippe Devienne, and Pierre Boulet. Multi-Layered Spiking Neural Network with Target Timestamp Threshold Adaptation and STDP. In *International Joint Conference on Neural Networks*, 2019.

[21] Mireille El-Assal, Pierre Tirilly, and Ioan Marius Bilasco. 2D Versus 3D Convolutional Spiking Neural Networks Trained with Unsupervised STDP for Human Action Recognition. In *International Joint Conference on Neural Networks*, 2022.

[22] Milad Mozafari, Saeed Reza Kheradpisheh, Timothee Masquelier, Abbas Nowzari-Dalini, and Mohammad Ganjtabesh. First-Spike-Based Visual Categorization Using Reward-Modulated STDP. *Transactions on Neural Networks and Learning Systems*, 29:6178–6190, 2018.

[23] Dongcheng Zhao, Yi Zeng, Tielin Zhang, Mengting Shi, and Feifei Zhao. GLSNN: A Multi-Layer Spiking Neural Network Based on Global Feedback Alignment and Local STDP Plasticity. *Frontiers in Computational Neuroscience*, 14, 2020.

[24] Fangxin Liu, Wenbo Zhao, Yongbiao Chen, Zongwu Wang, Tao Yang, and Li Jiang. SSTDP: Supervised Spike Timing Dependent Plasticity for Efficient Spiking Neural Network Training. *Frontiers in Neuroscience*, 15, 2021.

[25] Amar Shrestha, Khadeer Ahmed, Yanzhi Wang, and Qinru Qiu. Stable Spike-Timing Dependent Plasticity Rule for Multilayer Unsupervised and Supervised Learning. In *International Joint Conference on Neural Networks*, pages 1999–2006, 2017.

[26] Milad Mozafari, Mohammad Ganjtabesh, Abbas Nowzari-Dalini, Simon J. Thorpe, and Timothée Masquelier. Bio-Inspired Digit Recognition Using Reward-Modulated Spike-Timing-Dependent Plasticity in Deep Convolutional Networks. *Pattern Recognition*, 94, 2019.

[27] Johannes C. Thiele, Olivier Bichler, and Antoine Dupret. Event-Based, Timescale Invariant Unsupervised Online Deep Learning with STDP. *Frontiers in Computational Neuroscience*, 12, 2018.

[28] Chankyu Lee, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Deep Spiking Convolutional Neural Network Trained with Unsupervised Spike-Timing-Dependent Plasticity. *Transactions on Cognitive and Developmental Systems*, 11:384–394, 2019.

[29] Gaspard Goupy, Pierre Tirilly, and Ioan Marius Bilasco. Paired Competing Neurons Improving STDP Supervised Local Learning in Spiking Neural Networks. *Frontiers in Neuroscience*, 18, 2024.

[30] Peter Diehl and Matthew Cook. Unsupervised Learning of Digit Recognition Using Spike-Timing-Dependent Plasticity. *Frontiers in Computational Neuroscience*, 9, 2015.

[31] Paul Ferré, Franck Mamalet, and Simon J. Thorpe. Unsupervised Feature Learning with Winner-Takes-All Based STDP. *Frontiers in Computational Neuroscience*, 12, 2018.

[32] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training Deep Spiking Neural Networks Using Backpropagation. *Frontiers in Neuroscience*, 10, 2016.

[33] Lianhua Qu, Zhenyu Zhao, Lei Wang, and Yong Wang. Efficient and Hardware-Friendly Methods to Implement Competitive Learning for Spiking Neural Networks. *Neural Computing and Applications*, 32, 2020.

[34] Yunzhe Hao, Xuhui Huang, Meng Dong, and Bo Xu. A Biologically Plausible Supervised Learning Method for Spiking Neural Networks Using the Symmetric STDP Rule. *Neural Networks*, 121:387–395, 2020.

[35] Yeshwanth Bethi, Ying Xu, Gregory Cohen, André Van Schaik, and Saeed Afshar. An Optimized Deep Spiking Neural Network Architecture Without Gradients. *IEEE Access*, 10:97912–97929, 2022.

[36] Nicolas Frémaux and Wulfram Gerstner. Neuromodulated Spike-Timing-Dependent Plasticity, and Theory of Three-Factor Learning Rules. *Frontiers in Neural Circuits*, 9, 2015.

[37] Filip Ponulak and Andrzej Kasiński. Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting. *Neural Computation*, 22: 467–510, 2010.

[38] Amirhossein Tavanaei and Anthony Maida. BP-STDP: Approximating Backpropagation Using Spike Timing Dependent Plasticity. *Neurocomputing*, 330:39–47, 2019.

[39] Amar Shrestha, Haowen Fang, Qing Wu, and Qinru Qiu. Approximating Back-Propagation for a Biologically Plausible Local Learning Rule in Spiking Neural Networks. In *International Conference on Neuromorphic Systems*, 2019.

[40] Vahid Saranirad, Shirin Dora, T. M. McGinnity, and Damien Coyle. Assembly-Based STDP: A New Learning Rule for Spiking Neural Networks Inspired by Biological Assemblies. In *International Joint Conference on Neural Networks*, 2022.

[41] Michael Beyeler, Nikil D. Dutt, and Jeffrey L. Krichmar. Categorization and Decision-Making in a Neurobiologically Plausible Spiking Network Using a STDP-Like Learning Rule. *Neural Networks*, 48:109–124, 2013.

[42] Xiaoling Luo, Hong Qu, Yun Zhang, and Yi Chen. First Error-Based Supervised Learning Algorithm for Spiking Neural Networks. *Frontiers in Neuroscience*, 13, 2019.

[43] Tengxiao Wang, Cong Shi, Xichuan Zhou, Yingcheng Lin, Junxian He, Ping Gan, Ping Li, Ying Wang, Liyuan Liu, Nanjian Wu, and Gang Luo. CompSNN: A Lightweight Spiking Neural Network Based on Spatiotemporally Compressive Spike Features. *Neurocomputing*, 425:96–106, 2021.

[44] Shruti R. Kulkarni and Bipin Rajendran. Spiking Neural Networks for Handwritten Digit Recognition—Supervised Learning and Network Optimization. *Neural Networks*, 103:118–127, 2018.

[45] Sergey A. Lobov, Andrey V. Chernyshov, Nadia P. Krilova, Maxim O. Shamshin, and Victor B. Kazantsev. Competitive Learning in a Spiking Neural Network: Towards an Intelligent Pattern Classifier. *Sensors*, 20, 2020.

[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[47] Qiang Yu, Chenxiang Ma, Shiming Song, Gaoyan Zhang, Jianwu Dang, and Kay Chen Tan. Constructing Accurate and Efficient Deep Spiking Neural Networks with Double-Threshold and Augmented Schemes. *Transactions on Neural Networks and Learning Systems*, 33:1714–1726, 2022.

[48] Gaspard Goupy, Alexandre Juneau-Fecteau, Nikhil Garg, Ismael Balafrej, Fabien Alibart, Luc Frechette, Dominique Drouin, and Yann Beilliard. Unsupervised and Efficient Learning in Sparsely Activated Convolutional Spiking Neural Networks Enabled by Voltage-Dependent Synaptic Plasticity. *Neuromorphic Computing and Engineering*, 3, 2023.

[49] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86:2278–2323, 1998.

[50] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *ArXiv*, arXiv:1708.07747 [cs.LG], 2017.

[51] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, USA, 2009.

[52] Adrien Journé, Hector Garcia Rodriguez, Qinghai Guo, and Timoleon Moraitis. Hebbian Deep Learning Without Feedback. *International Conference on Learning Representations*, 2023.

[53] Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. Spike-Based Strategies for Rapid Processing. *Neural Networks*, 14:715–725, 2001.

[54] Maryam Mirsadeghi, Majid Shalchian, Saeed Reza Kheradpisheh, and Timothée Masquelier. Spike Time Displacement-Based Error Backpropagation in Convolutional Spiking Neural Networks. *Neural Computing and Applications*, 35:15891–15906, 2023.

[55] Amar Shrestha, Haowen Fang, Daniel Patrick Rider, Zaidao Mei, and Qinru Qiu. In-Hardware Learning of Multilayer Spiking Neural Networks on a Neuromorphic Processor. In *Design Automation Conference*, pages 367–372, 2021.

[56] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data Using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[57] Saeed Reza Kheradpisheh and Timothée Masquelier. Temporal Backpropagation for Spiking Neural Networks with One Spike per Neuron. *International Journal of Neural Systems*, 30, 2020.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Sections 4 and 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Sections 3, 4, 5.1. See Supplementary Material Sections 1 and 2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide our code as well as instructions for reproducing our main results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Section 5.1. See Supplementary Material Section 2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Results include the standard deviation reported from 10-fold experiments. See Section 5.1.5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Supplementary Material Section 2.4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This paper covers fundamental research. We do not foresee any such direct impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use public datasets, which we credited according to the authors' instructions. See Supplementary Material Sections 2.2 and 2.5.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide our code with documentation.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.