Improving STDP-based Visual Feature Learning with Whitening

Pierre Falez[1], Pierre Tirilly[1], and Ioan Marius Bilasco[1]
[1] *Univ. Lille, CNRS, Centrale Lille, UMR 9189 – CRIStAL – Centre de Recherche en Informatique, Signal et Automatique de Lille*
F-59000, Lille, France
Email: firstname.lastname@univ-lille.fr

***Abstract*—In recent years, spiking neural networks (SNNs) emerge as an alternative to deep neural networks (DNNs). SNNs present a higher computational efficiency – using low-power neuromorphic hardware – and require less labeled data for training – using local and unsupervised learning rules such as spike timing-dependent plasticity (STDP). SNNs have proven their effectiveness in image classification on simple datasets such as MNIST. However, to process natural images, a pre-processing step is required. Difference-of-Gaussians (DoG) filtering is typically used together with on-center / off-center coding, but it results in a loss of information that decreases the classification performance. In this paper, we propose to use whitening as a pre-processing step before learning features with STDP. Experiments on CIFAR-10 show that whitening allows STDP to learn visual features that are visually closer to the ones learned with standard neural networks, with a significantly increased classification performance as compared to DoG filtering. We also propose an approximation of whitening as convolution kernels that is computationally cheaper to learn and more suited to be implemented on neuromorphic hardware. Experiments on CIFAR-10 show that it performs similarly to regular whitening. Cross-dataset experiments on CIFAR-10 and STL-10 also show that it is stable across datasets, making it possible to learn a single whitening transformation to process different datasets.**

***Index Terms*—Convolutional neural networks, Pattern recognition, Unsupervised learning**

I. Introduction

In recent years, deep neural networks (DNNs) have become a *de facto* standard in machine learning, thanks to their ability to learn complex representations from large amounts of data. They have demonstrated their superiority over other models in many tasks, including image and video classification, speech recognition, and natural language understanding. However, they suffer from two major drawbacks that hamper their adoption at a large scale. First, training a DNN is computationally expensive, due to its large number of parameters and the large amounts of data required to effectively estimate them. As a consequence, DNN training is usually performed on GPUs or TPUs, which consume large amounts of energy. Moreover, DNNs mostly rely on supervised learning, which requires these large amounts of data (e.g., millions of samples in the case of image classification [1]) to be annotated manually beforehand, making them difficult to apply to new tasks, unless one is willing to spend large amounts of time and money on the labeling process. Spiking neural networks (SNNs) offer an alternative to DNNs; they can be implemented efficiently through low-power neuromorphic hardware [2], which solves one issue of DNNs. Data labeling can also be avoided – to some extent – through the use of unsupervised learning rules. Spike-timing-dependent plasticity (STDP) is one of those rules, that can enable effective unsupervised learning in SNNs [3] and is compatible with neuromorphic hardware [4].

In our work, we aim at learning visual features through SNNs trained with STDP, with the long-term goal of producing end-to-end spiking architectures compatible with low-power, neuromorphic, hardware. Such a system includes image pre-processing, neural coding of the pre-processed images into spikes, neuron and synapse models, learning rules, and finally the feature classifier; all these elements should ideally be implementable through neuromorphic hardware. In [5], an in-depth study of STDP-based feature learning for image recognition concluded that STDP-based SNNs cannot currently compete with traditional neural networks models of visual feature learning (namely, auto-encoders), and pointed out some reasons for the ineffectiveness of SNNs, especially the pre-processing of images and the inhibition mechanisms.

In this paper, we specifically address the issue of image pre-processing for visual feature learning and natural image classification with STDP-based SNNs. To be processed by STDP networks, natural images must first be pre-processed so that the spike trains representing them can encode relevant visual information. Indeed, directly encoding pixel values as spikes leads to learning mostly patterns consisting of uniform regions, as shown empirically in [5]. To prevent this, images are usually converted to grayscale, then on-center / off-center (OC/OC) coding [6], [7] (or some equivalent edge-extraction method such as Gabor filters [8]) is applied. This coding is inspired by biological vision, and is also related to the SIFT keypoint detector widely used in computer vision [9]. It extracts edges from the images by applying a difference of Gaussians (DoG) filter (see Figure 1(b)). The spike trains can then encode edge information, which is richer than raw pixel information. However, it prevents the STDP networks from learning also visual patterns based on colors, as standard deep neural networks do [1], [10]. Applying OC/OC coding to the R, G, and B color channels independently (see Figure 1)

This work has been partly funded by IRCICA (Univ. Lille, CNRS, USR 3380 – IRCICA, F-59000 Lille, France) under the Bioinspired Project.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

does not solve this issue, as it only allows to learn edge patterns specific to one of the three color channels rather than actual color patterns. Consequently, OC/OC coding has been empirically shown to be detrimental to image classification [5].

Instead, we propose to use whitening as a pre-processing step to handle natural images with STDP-based SNNs. It is commonly used in computer vision to pre-process images [10], [11]. Generally speaking, whitening is used to standardize and de-correlate data; it projects the data into a new, orthonormal, space so that its components are centered, independent, and have unit variance. When applied to images, it discards first-order correlations, which correspond to the fact that pixels that are spatially close to each other tend to have similar values; visually, it highlights edges and high frequency features. It helps learn non-trivial correlations between pixels [10].

The whitening transformation is typically computed from a dataset using principal component analysis (PCA) or zero-phase component analysis (ZCA) [12], and applied to whole images. As we aim at training end-to-end STDP-based systems that can be implemented on energy-efficient hardware, whitening cannot be applied as is, for three reasons:

- learning a PCA or ZCA transformation on whole images is computationally expensive and involves operations on dense matrices that cannot be implemented simply through neuromorphic hardware;
- only applying a learned whitening transformation to whole images cannot be implemented efficiently through neuromorphic hardware either, because it is not local;
- the transformation is data-dependent, so a new transformation should be computed for every new dataset.

Using whitening as a pre-processing for SNN-based image analysis is only valuable if these issues can be circumvented.

The contribution of this paper is three-fold.

1) We show that using whitening as a pre-processing step allows STDP to learn patterns that are similar to what standard deep neural networks can learn, and is superior to OC/OC coding when performing image classification based on features learned with STDP.
2) We propose an approximation of ZCA whitening based on convolution kernels, that can be computed more efficiently and fits the constraints of neuromorphic hardware. Experiments show that this approximation yields the same performances as standard ZCA whitening.
3) We show through cross-dataset experiments that it is possible to pre-compute a single whitening transformation on one dataset and apply it to other datasets with no significant impact on the classification performance.

II. Related Work

a) SNN-based Visual Feature Learning: Image classification and visual feature learning with SNNs have received an increasing interest over the last years (see [13] for a recent survey). Most authors focus on simple datasets like MNIST, which offer limited challenges. Some models were evaluated on more complex datasets of natural images such as CIFAR-10, but they usually use training procedures that cannot be implemented on neuromorphic hardware (e.g., converting pre-trained DNN models to SNNs). Such models offer limited benefits over DNNs, since training the model is the most computationally expensive step. SNN models that can be trained on energy-efficient hardware usually use STDP learning rules. Their performances are still behind other models, especially DNNs; as a consequence, most work still focuses on simple datasets like MNIST [14]–[16] and ETH-80 [7], [8]. One reason is the difficulty to train multi-layer STDP networks: multi-layer models based on STDP [5], [7] have only been proposed very recently. Another reason is the difficulty to handle natural color images, due to the ineffectiveness of OC/OC coding [5]. As a result, complex datasets of natural images are seldom used to evaluate STDP-based SNNs; recent examples include the Caltech Faces/Motorbikes dataset [7], [17], and CIFAR-10, CIFAR-100 or STL-10 [5]. In this paper, we aim at improving the ability of STDP-based networks to learn from complex natural images.

b) Whitening for Feature Learning and Deep Learning: Whitening has been especially studied as a pre-processing step in unsupervised visual feature learning [10], [11]. The reported results were sometimes contradictory: Krizhevsky and Hinton [11] evaluated the role of whitening in image classification based on raw pixels or on visual features learned with restricted Boltzmann machines (RBM), and concluded that whitened images do not provide any improvement, whereas Coates *et al.* [10] reported significant and consistent improvements in classification performance when learning visual features on whitened images with k-means, mixtures of Gaussians, auto-encoders, and RBMs. Whatever the actual outcomes of whitening can be with traditional algorithms, the debate is not relevant with STDP-based SNNs, as learning from raw images is not an option in this case [5]. Whitening can also be used to normalize network activations between layers of a DNN [18], in a process similar to batch normalization. However, this is not related to our goal, as we only consider whitening as an alternative to OC/OC coding.

c) Whitening and SNNs: To our knowledge, only Burbank [19] used whitening as a pre-processing step before learning visual features from natural images with SNNs. No specific reason for using whitening was provided, other than re-using the data of Olshausen & Field [20]. The evaluation of the resulting features does not include recognition performance and the performance of whitening w.r.t. other pre-processing methods (e.g., OC/OC coding) was not assessed.

III. Background

A. Unsupervised Feature Learning and Image Classification

The problem of image classification can be modeled as finding a function $f : \mathbb{I} \rightarrow \mathbb{C}$ which assigns to an image $I \in \mathbb{I}$ the label $c \in \mathbb{C}$ of the class it belongs to. A typical DNN-based approach would directly infer f from labeled training data. Other approaches model f as the composition of three individual functions: a feature extractor f_e, a feature aggregator f_a, and a supervised classifier f_c. The feature extractor is a function $f_e : \mathbb{I} \rightarrow \mathbb{R}^{m \times d}$ that converts an

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

(a) (b) (c) (d)

Fig. 1: Pre-processing for color images: (a) raw RGB image, (b) on-center / off-center coding on grayscale image (c) on-center / off-center coding on color image, and (d) whitened RGB image.

image I into a set of m visual features representative of its visual content (shape, color...); each feature is modeled as a vector of dimension d. The feature aggregator is a function $f_a : \mathbb{R}^{m \times d} \rightarrow \mathbb{R}^{d'}$ that aggregates the m feature vectors into a single feature vector of dimension d', typically through some pooling operation. Finally, the classifier $f_c : \mathbb{R}^{d'} \rightarrow \mathbb{C}$ assigns a class $c \in \mathbb{C}$ to an image $I \in \mathbb{I}$ based on its aggregated feature vector $(f_a \circ f_e(I)) : c = f_c \circ f_a \circ f_e(I)$.

In this work, the feature extractor f_e is a convolutional single-layer SNN that learns features from image data using an unsupervised STDP learning rule. As we aim at evaluating only the ability of STDP to learn visual features, we rely on more classical tools for the feature aggregator f_a (sum pooling) and the classifier f_c (SVM). The details of our recognition system are provided in Section V-A.

B. SNN Model

An STDP-based SNN typically includes the following components: a neural coding model, which converts input data into spikes, a spiking neuron model, an STDP learning rule, and homeostasis mechanisms that ensure that the activity of the network remains consistent. Since we aim at learning visual features for classification, we also need a "neural decoding" model that converts output spikes back into numerical values that can be fed to the feature aggregator or the classifier. We use the same components as in [17], which provide state-of-the-art performance for STDP-based visual feature learning.

a) Neural Coding: We use latency coding [21], which is one variant of temporal coding, to convert input values x into spikes. Earlier spikes encode larger values. Spike timestamps are generated as follows:

$$t = t_{\mathrm{s}}(1 - x) \tag{1}$$

with t the timestamp of the spike, x the input value, and t_{s} the duration of the exposition of a sample to the network. Hence, there is at most one spike per input per sample.

b) Neuron Model: The SNN uses integrate-and-fire (IF) neurons [22]. This model is defined as follows:

$$c_{\mathrm{m}} \frac{\partial v}{\partial t} = z(t), v \leftarrow v_{\mathrm{rest}} \text{ when } v \geq v_{\mathrm{th}} \tag{2}$$

with v the membrane potential, v_{rest} the resting potential, c_{m} the membrane capacitance, v_{th} the threshold of the neuron, and $z(t)$ the input current of the neuron ($z(t) = 1$ if an input spike is received at time t, and $z(t) = 0$ otherwise).

c) Synapse Model: Every time a neuron fires a spike, the weights of its input connections are updated following a STDP rule, according to the activity of the corresponding pre-synaptic neurons. We use multiplicative STDP [14] to train synaptic weights w:

$$\Delta_w = \begin{cases} \eta_w e^{-\beta \frac{w - w_{\min}}{w_{\max} - w_{\min}}} & \text{if } 0 \leq t_{\mathrm{post}} - t_{\mathrm{pre}} \leq t_{\mathrm{LTP}} \\ -\eta_w e^{-\beta \frac{w_{\max} - w}{w_{\max} - w_{\min}}} & \text{otherwise} \end{cases} \tag{3}$$

with $w_{\min}$ and $w_{\max}$ the bounds of the weights w, Δ_w the update applied the weight ($w_{t+1} = w_t + \Delta_w$), η_w the learning rate, and t_{pre} and t_{post} the firing timestamps of the pre-synaptic and post-synaptic neurons, respectively. β is a parameter that controls the saturation effect of the learning rule (increasing β reduces the saturation of weights).

d) Homeostasis: Homeostasis in the network is maintained by adapting neuron thresholds. Threshold values v_{th} are learned with the adaptation rule proposed in [17] and a winner-takes-all (WTA) mechanism: when a neuron wins the competition (i.e. it fires a spike first during the exposition of a sample), it applies the STDP rule to update its synaptic weights and it adapts its threshold v_{th} with the following update:

$$\Delta^1_{\mathrm{th}} = \eta_{\mathrm{th}} \tag{4}$$

with Δ^1_{th} the change applied to the neuron threshold v_{th} and η_{th} the learning rate of threshold adaptation. This rule ensures that no neuron will always be the first to emit a spike. The other neurons do not apply STDP and decrease their thresholds as follows:

$$\Delta^1_{\mathrm{th}} = -\frac{\eta_{\mathrm{th}}}{|l_{\mathrm{output}}|} \tag{5}$$

with $|l_{\mathrm{output}}|$ the number of neurons in competition. This second update promotes diversity in neurons by lowering the thresholds of neurons that emit few or no spikes.

Moreover, each time a neuron fires a spike, all the neurons in competition apply the following update to their threshold:

$$\Delta^2_{\mathrm{th}} = -\eta_{\mathrm{th}}(t - t_{\mathrm{exp}}) \tag{6}$$

with η_{th} the threshold learning rate and t the timestamp at which the neuron fired the spike. t_{exp} is a manually-defined

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

timestamp objective at which neurons should fire spikes: it controls the number of input spikes to be integrated before an output spike is emitted, and, so, the nature of the filters to be learned [5], [17].

e) Neural Decoding: Spikes generated at the output of the SNN are converted back into numerical values as follows:

$$y = \min\left(1, \max\left(0, 1 - \frac{t - t_{\text{exp}}}{t_{\text{s}} - t_{\text{exp}}}\right)\right) \tag{7}$$

C. ZCA Whitening

Whitening is a data-dependent transformation that decorrelates and standardizes the data. Several whitening transformations can exist for a given dataset, as whitened data remains whitened under rotations. Among these, ZCA whitening [12] is the transformation that produces the whitened data that remains the closest to the original data. When applied to images, ZCA whitening produces images that are still recognizable by the human eye (as opposed to, for instance, PCA whitening).

Let X be a centered data matrix and Σ its covariance matrix. Σ can be decomposed so that:

$$\Sigma = \mathrm{U}\,\Lambda\,\mathrm{U}^{-1} \tag{8}$$

with U the matrix of eigenvectors of Σ and Λ the diagonal matrix of its eigenvalues ($\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$).

The ZCA transformation matrix $\mathrm{W}_{\text{whiten}}$ for data X is computed as follows:

$$\mathrm{W}_{\text{whiten}} = \mathrm{U}\,\sqrt{(\Lambda + \epsilon)^{-1}}\,\mathrm{U}^{T} \tag{9}$$

with ϵ the whitening coefficient, which adds numerical stability and acts as a low pass filter. As in PCA, it is possible to retain only the k largest eigenvalues and their corresponding eigenvectors, to eliminate the least significant components of the data, which may correspond to noise. We note $\rho \in [0, 1]$ the ratio of the largest eigenvalues that are retained.

Finally, the ZCA whitened data $\mathrm{X}_{\text{whiten}}$ is computed from the ZCA transformation $\mathrm{W}_{\text{whiten}}$ as:

$$\mathrm{X}_{\text{whiten}} = \mathrm{W}_{\text{whiten}}\,\mathrm{X} \tag{10}$$

IV. Contribution

A. Encoding Whitened Data as Spikes

The first part of our contribution is to enable the conversion of whitened data into spikes. The whitening transformation outputs both positive and negative values, which correspond to the positive or negative contributions of the data to the components of the transformation; larger values correspond to more significant contributions. These values must be converted into spikes following the principle of temporal coding: larger values must correspond to earlier spikes. Similarly to OC/OC coding, we split the values into two channels, a positive one and a negative one. The conversion process follows these steps:

1) The data matrix X is whitened using the learned ZCA transformation.
2) The components of each sample in $\mathrm{X}_{\text{whiten}}$ are scaled in $[-1, 1]$ according to the minimum and maximum values of the sample.
3) Positive and negative values are split into two channels X_+ and X_-: $X_+ = \max(0, \mathrm{X}_{\text{whiten}})$, $X_- = \max(0, -\mathrm{X}_{\text{whiten}})$.

The values can finally be converted into spikes by using latency coding (Equation 1).

B. Approximating Whitening with Convolution Kernels

Applying the whitening transformation to images is computationally expensive and is not easily implementable on neuromorphic architectures. In opposition, the DoG filter of OC/OC coding is a pre-processing which is already well-used with SNNs and can be computed by simply convolving an image with a suitable kernel. In this section, we show how to approximate whitening by convolution kernels, to benefit both from the ease of implementation of DoG filtering and from the performance of whitening. Our approach also reduces the cost of learning the whitening transformation matrix.

The general idea is to learn the whitening transform on small patches rather whole images, then to approximate the patch whitening transformation by the whitening transformation of the central pixel of the patches, which can be expressed as a convolution kernel. The overall process is illustrated in Figure 2. First, n_{p} patches of dimensions $[p_{\text{w}}, p_{\text{h}}, x_{\text{d}}]$ ($x_{\text{d}} = 3$ for RGB images) are extracted from the dataset (e.g. using dense or random sampling), forming a data array X_p of dimensions $[n_{\text{p}}, p_{\text{w}}, p_{\text{h}}, x_{\text{d}}]$. The patches are vectorized to form a data matrix $\widetilde{X_p}$ of dimensions $[n_{\text{p}}, p_{\text{w}} \times p_{\text{h}} \times x_{\text{d}}]$. A ZCA transformation matrix $\mathrm{W}_{\text{whiten}}$ of dimensions $[p_{\text{w}} \times p_{\text{h}} \times x_{\text{d}}, p_{\text{w}} \times p_{\text{h}} \times x_{\text{d}}]$ is computed from $\widetilde{X_p}$ (Equations 8 and 9). Finally, $\mathrm{W}_{\text{whiten}}$ is converted into x_{d} convolution kernels K_c of dimension $[p_{\text{w}}, p_{\text{h}}, x_{\text{d}}]$. To do so, an impulse response array J_c is created for each channel c with only its central value in this channel set to 1:

$$\mathrm{J}_c(i, j, k) = \begin{cases} 1 & \text{if } k = c \text{ and } i = \frac{p_{\text{w}}}{2} \text{ and } j = \frac{p_{\text{h}}}{2} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

with $i \in [0, p_{\text{w}}[$, $j \in [0, p_{\text{h}}[$, and $k \in [0, x_{\text{d}}[$ the coordinates in array J_c and $c \in [0, x_{\text{d}}[$ the corresponding channel. Each array J_c is vectorized into a vector $\widetilde{\mathrm{J}_c}$ of dimension $p_{\text{w}} \times p_{\text{h}} \times x_{\text{d}}$. A whitening kernel K_c is computed for each channel c as:

$$\widetilde{K_c} = \mathrm{W}_{\text{whiten}}\,\widetilde{\mathrm{J}_c} - \text{mean}(\mathrm{W}_{\text{whiten}}\,\widetilde{\mathrm{J}_c}) \tag{12}$$

where $\widetilde{K_c}$ is the vectorized version of K_c. An image can be whitened by convolving it with the whitening kernels, each providing the corresponding channel of the filtered image. Each whitening kernel K_c corresponds to the whitening transformation of the central pixel of a channel of the patches. Examples of whitening kernels generated by this method and the resulting filtered images are shown in Figure 3.

Computing the whitening transformation from a data matrix of dimensions $[M, N]$ has time complexity $O(MN^2 + N^\alpha), \alpha \in [2.3, 3]$, with α depending on the algorithm used for matrix multiplication (computing the covariance matrix is $O(MN^2)$ and eigenvalue decomposition and ZCA rotation are $O(N^\alpha)$). Our approach makes it faster to compute by several orders of magnitude by decreasing N drastically, from the size of an image to the size of a patch.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

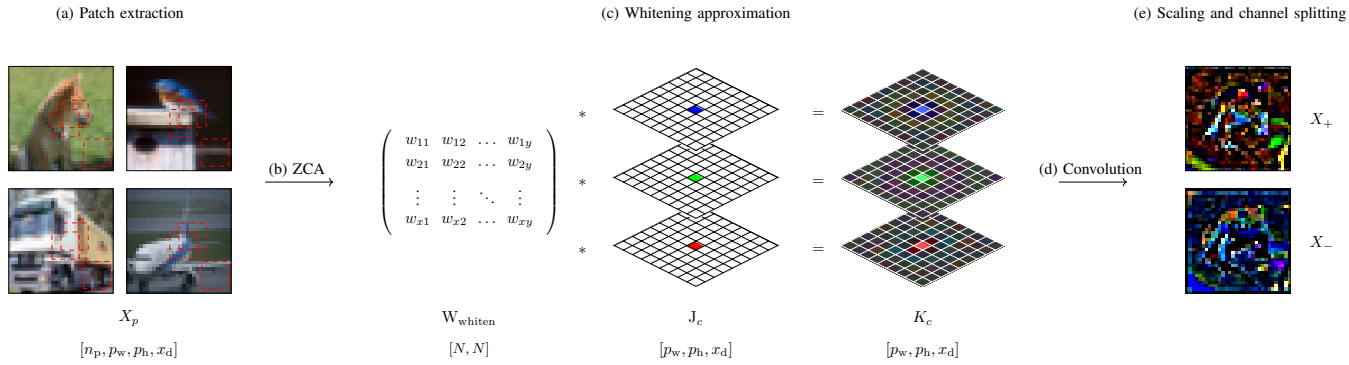
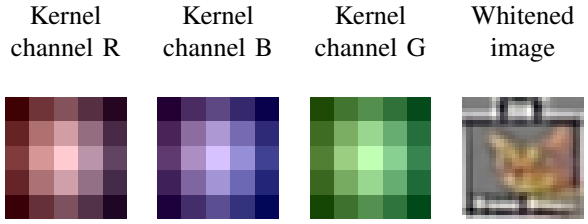


Fig. 2: Whitening approximation procedure ($N = p_\mathrm{w} \times p_\mathrm{h} \times x_\mathrm{d}$): computation of whitening kernels (a-c) and application of whitening to images (d-e).



(a) $p_\mathrm{w}, p_\mathrm{h} = 5 \times 5$, $\rho = 0.2$, $\epsilon = 10^{-3}$.

(b) $p_\mathrm{w}, p_\mathrm{h} = 9 \times 9$, $\rho = 0.6$, $\epsilon = 10^{-3}$.

(c) $p_\mathrm{w}, p_\mathrm{h} = 11 \times 11$, $\rho = 1.0$, $\epsilon = 10^{-3}$.

(d) $p_\mathrm{w}, p_\mathrm{h} = 9 \times 9$, $\rho = 1.0$, $\epsilon = 10^{-1}$.

Fig. 3: Examples of whitening kernels approximating the ZCA transformation on RGB images.

V. Results

A. Experimental Protocol

a) Objectives: In these experiments, we evaluate:

- the performance of our whitening kernels versus standard whitening, and its sensitivity to major parameters;
- the performance of whitening versus standard OC/OC coding as a pre-processing step for feature learning;
- the stability of whitening kernels across datasets, by performing cross-dataset experiments in which the whitening transformation is trained on one dataset and applied to another dataset for feature learning and image recognition.

b) Recognition System: Our recognition system follows the same general procedure as [10] and [5]. The system is depicted in Figure 4. The major stages of the system are:

1) Image pre-processing (Figure 4(a)) through OC/OC coding, standard whitening (see Section III-C), or whitening kernels (see Section IV-B).
2) Feature extraction (f_e) by a single-layer convolutional SNN following the model presented in Section III-B (see Figure 4(c)).
3) Feature aggregation (f_a) through sum pooling over 2×2 non-overlapping image regions (see Figure 4(d)).
4) Feature vector classification (f_c) with a linear SVM (see Figure 4(e)).

c) Datasets: We use CIFAR-10 [11] as a reference dataset. This dataset contains 60,000 color images of size 32×32, divided into 10 classes; it is split into a training set of 50,000 images and a test set of 10,000 images. We also use the labeled part of STL-10 [10] for cross-dataset experiments. It contains 13,000 96×96 color images split into 5,000 training images and 8,000 test images. Note that the scales of the images in the two datasets are different, making cross-dataset experiments more challenging for our whitening kernels.

d) Computation of the Whitening Transformations: Whitening transformations are computed on the training set of CIFAR-10 for regular experiments; for cross-dataset experiments, they are computed on STL-10 (resp. CIFAR-10) when feature learning and classification are performed on CIFAR-10 (resp. STL-10). The standard whitening transformation is learned using the whole training set. Patch-based whitening transformations are learned on 10^6 patches densely sampled with a stride of 2 from images of the training set.

e) Implementation Details: The parameters of the SNN are set to the values in Table I, unless otherwise specified. Average recognition rates and standard deviations over three runs are reported. The simulator CSNNS[1] [23] is used to implement all the experiments.

[1]This tool is open-source and can be downloaded at https://gitlab.univ-lille.fr/bioinsp/falez-csnn-simulator.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

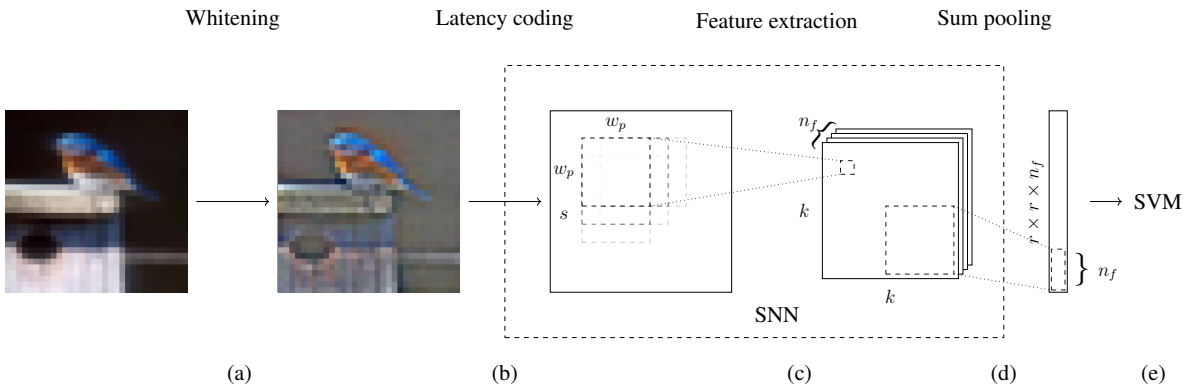


Fig. 4: Recognition system used in our experiments.

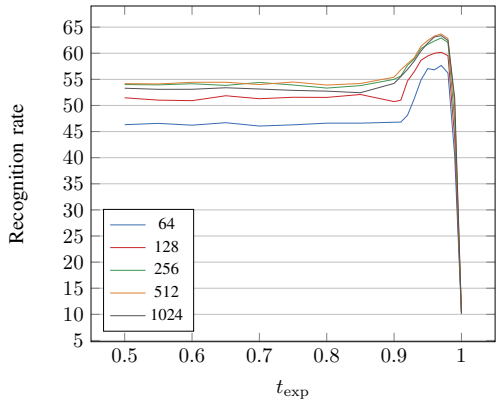
TABLE I: Default parameters used in the experiments.

Neural Coding			
t_s	1		
Neuron			
$v_{th}(0)$	$\sim \mathcal{G}(10, 0.1)$	v_{rest}	0
Threshold Adaptation			
t_{exp}	0.97	$\eta_{th}(0)$	1
Training			
α	0.95	n_{epoch}	100
STDP			
w_{min}	0	w_{max}	1
$\eta_w(0)$	0.1	$w(0)$	$\sim \mathcal{U}(0, 1)$
β	1		
Network architecture			
filter size	5×5	stride	1
padding	0		

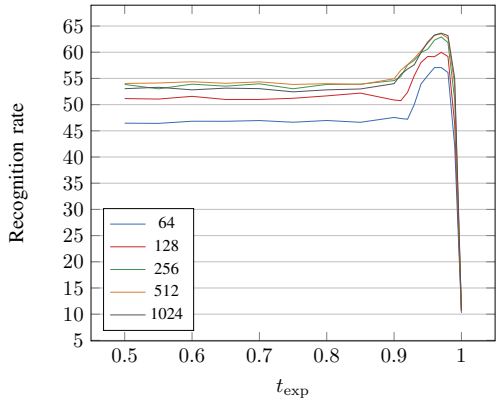
B. Standard Whitening vs Whitening Kernels

In this section, we compare standard whitening and whitening kernels in terms of the classification performance of our system (see Figure 4); we vary the major parameters: t_{exp} and the number of filters used in the convolution layer. Figure 5 shows that the behavior of both whitening processes is fairly similar. The reported performances of whitening kernels were obtained using 9×9 patches, $\epsilon = 10^{-2}$ and $\rho = 1.0$. The performances achieved for each t_{exp} and for each number of filters considered are similar. This shows that the approximation of whitening by convolution kernels performs as well as the original whitening transformation. For both methods, t_{exp} seems optimal around 0.96.

An in-depth exploration of the parameters of whitening kernels (patch size, whitening coefficient ϵ, and ratio of eigenvectors ρ) was conducted. Table II shows the results obtained with various kernel sizes, ϵ, and numbers of learned filters (l_{output}). Several observations can be drawn from these results. First, with 64 filters, the performances are overall lower than with 256. However, no strong improvement is observed when using 1024 learned filters. Second, for each configuration using a fixed number of filters, the performances are quite stable regardless of patch size. However, overall,



(a) Standard whitening.



(b) Whitening kernels.

Fig. 5: Classification performance on CIFAR-10 with standard whitening (a) vs whitening kernels (b).

slightly better average performances are obtained when the patch size increases (see 9×9 and 11×11 configurations). A value of 10^{-2} for the whitening coefficient ϵ seems more adequate when more filters are used.

Table III reports the performances with varying patch sizes, ρ, and numbers of learned filters. Among all configurations having a given number of filters, the performances are similar. This shows once more the stability and the genericity of the

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE II: Recognition rate (%) w.r.t. patch size and ϵ (ρ = 1.0).

$\lvert l_{\text{output}}\rvert$	$p_{\text{w}} \times p_{\text{h}}$	ϵ 10^{-1}	10^{-2}	10^{-3}	10^{-4}
64	5×5	51.36±0.48	56.3±0.09	53.49±0.09	49.89±0.77
	7×7	53.04±0.31	56.74±0.21	54.93±0.29	49.74±0.88
	9×9	53.07±0.27	57.04±0.10	**60.13±0.23**	49.96±1.12
	11×11	53.65±0.14	57.04±0.05	55.2±0.29	50.29±0.44
256	5×5	59.32±0.02	62.02±0.16	58.62±0.3	54.46±0.27
	7×7	60.05±0.26	62.55±0.19	58.76±0.61	54.56±0.27
	9×9	59.81±0.39	**63.72±0.39**	59.12±0.34	54.62±0.18
	11×11	60.13±0.29	62.83±0.63	58.99±0.19	55.19±0.56
1024	5×5	60.13±0.37	62.91±0.34	57.98±0.2	53.94±0.13
	7×7	60.17±0.24	63.63±0.51	58.47±0.67	54.29±0.29
	9×9	60.72±0.37	**63.78±0.13**	58.71±0.40	53.54±0.42
	11×11	61.18±0.07	62.91±0.34	58.87±0.55	53.57±0.64

TABLE III: Recognition rate (%) w.r.t. patch size and ρ (ϵ = 10^{-2}).

$\lvert l_{\text{output}}\rvert$	$p_{\text{w}} \times p_{\text{h}}$	ρ 0.25	0.50	0.75	1.00
64	5×5	50.44±0.28	55.99±0.11	55.85±0.17	56.17±0.44
	7×7	53.32±0.45	56.83±0.04	56.72±0.17	56.60±0.23
	9×9	54.64±0.06	56.88±0.16	57.11±0.19	56.86±0.03
	11×11	55.76±0.07	57.28±0.23	57.12±0.28	**57.33±0.06**
256	5×5	57.75±0.03	61.60±0.04	61.53±0.20	61.95±0.37
	7×7	59.43±0.45	62.19±0.17	62.25±0.14	62.41±0.36
	9×9	60.00±0.07	62.37±0.39	62.57±0.10	62.41±0.21
	11×11	61.35±0.29	62.56±0.91	**62.97±0.28**	62.70±0.63
1024	5×5	57.9±0.30	62.87±0.32	62.80±0.35	62.88±0.45
	7×7	59.24±0.44	63.4±0.19	63.49±0.44	63.63±0.28
	9×9	59.11±0.27	63.70±0.09	63.39±0.46	63.61±0.38
	11×11	60.71±0.46	62.87±0.32	63.72±0.42	**63.82±0.45**

whitening kernels. We can also see that the steep increase in performance from 64 to 256 learned filters is not present when increasing the learning capacity from 256 to 1024 filters. Values of 0.75 and 1.00 for ρ provide the best results.

These results show that the benefits brought by whitening kernels are stable and can generalize to a wide set of settings.

C. On-center / Off-center Filtering vs Whitening

Table IV compares the performances on CIFAR-10 of whitening to the baseline OC/OC coding, for $|l_{\text{output}}| = 64$ and $|l_{\text{output}}| = 1024$. Whitening provides much better results than color OC/OC coding: +18% (+9 percentage points) with 64 filters and +11% (+6 pp.) with 1,024 filters. This may be due to its ability to retain color information and all spatial frequencies, whereas OC/OC coding only encodes edge information and a limited range of spatial frequencies.

Figure 6 shows samples of features learned on CIFAR-10 with STDP with the three pre-processing approaches, as well as features learned by an auto-encoder with standard whitening [10] and features learned by a regular DNN without whitening on ImageNet [1]. Whereas the filters learned with OC/OC coding (Figure 6(a)) are almost only oriented edges, the filters learned with whitening (Figures 6(b) and 6(c)) include both oriented edges and oriented color patterns. These filters are much closer to the ones that can be learned on whitened data by an auto-encoder (Figure 6(d)), but also the ones learned by DNNs on non-whitened data (Figure 6(e)).

TABLE IV: Performance of whitening versus on-center / off-center coding on CIFAR-10.

Method	64 filters	1,024 filters
On-center / off-center (grayscale) [5]	45.37%	52.77%
On-center / off-center (color) [5]	48.27%	56.93%
Standard whitening	**57.66%**	63.37%
Whitening kernels	57.07%	**63.64%**

TABLE V: Performances obtained in a cross-dataset configuration (first header row: dataset used for classification, second header row: dataset used to generate the whitening kernels).

$\lvert l_{\text{output}}\rvert$	CIFAR-10	CIFAR-10 STL-10	Δ	STL-10	STL-10 CIFAR-10	Δ
64	57.66±0.44	57.09±0.11	-0.57	57.08±0.44	56.97±0.34	-0.11
128	60.18±0.29	59.95±0.08	-0.23	58.93±0.25	58.74±0.48	-0.19
256	62.92±0.10	62.77±0.30	-0.15	59.86±0.32	59.74±0.43	-0.12
512	63.69±0.16	63.78±0.18	+0.09	60.73±0.46	60.72±0.42	-0.01
1024	63.37±0.21	63.80±0.56	+0.43	63.29±0.11	62.94±0.42	-0.35

The main difference with auto-encoder features is that they are more localized, which may be due to our filters being smaller in size (5×5 pixels vs 8×8 pixels in [10]).

D. Cross-dataset Experiments

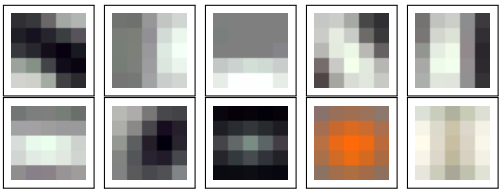
Since computing whitening transformations is computationally expensive and not suited to neuromorphic hardware, the learned transformations should fit different datasets, to avoid re-training. To test this, whitening kernels are computed independently from CIFAR-10 and STL-10, respectively. Then, the CIFAR-10 dataset is pre-processed using whitening kernels learned on STL-10, and reversely. We then measure the accuracy of our system on each whitened dataset.

Table V shows the results obtained using different configurations. We used several numbers of filters. We fixed the following parameters: $\rho = 1$, $\epsilon = 10^{-2}$, and patch size $= 9 \times 9$. We report the results obtained either using whitening kernels computed on the same dataset or whitening kernels computed on the other dataset. Regardless of the underlying configuration, the difference of the recognition rates between whitening kernels trained on same dataset and trained on a different datasets is negligible. In all cases, the difference between the two configurations is close to zero or statistically not significant (within twice the standard deviation). Thus, whitening kernels are dataset-independent and can be computed once, then re-used on multiple datasets.

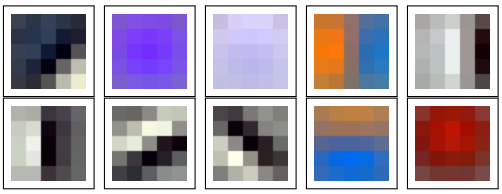
VI. Conclusion

SNNs trained with STDP are good candidates to produce ultra-low power neural networks. However, their performances are currently far behind DNNs. Notably, STDP cannot learn effective features on real-world color images. OC/OC coding, used to pre-process images in this context, is partially responsible for it, as it filters only a subrange of spatial frequencies from the images. In this paper, we showed that pre-processing images with whitening allows to learn more effective features, visually closer to the ones learned with

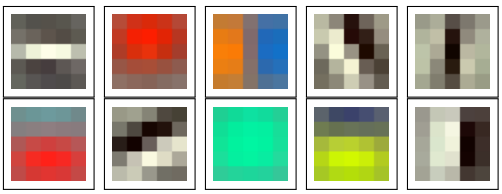
978-1-7281-6926-2/20/$31.00 ©2020 IEEE



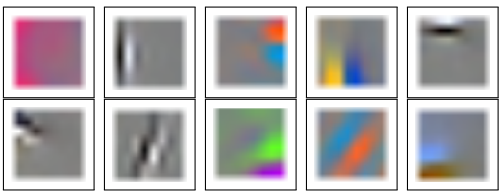
(a) OC/OC (color) + STDP ($\beta = 3.0$, $t_{\text{exp}} = 0.90$).



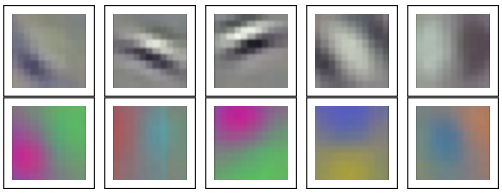
(b) Standard whitening + STDP ($\beta = 3.0$, $t_{\text{exp}} = 0.97$).



(c) Kernel whitening + STDP ($\beta = 3.0$, $t_{\text{exp}} = 0.97$).



(d) Standard whitening + autoencoders (images taken from [10]).



(e) No whitening + CNN (images taken from [1]).

Fig. 6: Samples of filters learned on CIFAR-10 (a-d) or ImageNet (e) with different pre-processing and learning methods.

standard neural networks. Implementing whitening on neuromorphic hardware may not be trivial, so we also propose to approximate whitening with convolution kernels to facilitate its implementation. It yields almost the same performance as traditional whitening. Cross-dataset experiments show stable performance of the whitening kernels over datasets, so it is possible to learn a single set of kernels to process different datasets, making it even more suitable in a low-power context.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2012, p. 1097–1105.

[2] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[3] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains," *PLoS One*, vol. 3, no. 1, Jan. 2008.

[4] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," *CoRR*, vol. abs/1705.06963, pp. 1–88, May 2017.

[5] P. Falez, P. Tirilly, I. M. Bilasco, P. Devienne, and P. Boulet, "Unsupervised visual feature learning with spike-timing-dependent plasticity: How far are we from traditional feature learning approaches?" *Pattern Recognition*, vol. 93, p. 418–429, 2019.

[6] A. Delorme, L. Perrinet, and S. J. Thorpe, "Networks of integrate-and-fire neurons using rank order coding B: Spike timing dependant plasticity and emergence of orientation selectivity," *Neurocomputing*, vol. 38, pp. 539–545, Jun. 2001.

[7] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, vol. 99, pp. 56–67, Mar. 2018.

[8] S. R. Kheradpisheh, M. Ganjtabesh, and T. Masquelier, "Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition," *Neurocomputing*, vol. 205, pp. 382–392, Sep. 2016.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[10] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 15. PMLR, Apr. 2011, pp. 215–223.

[11] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., Apr. 2009.

[12] A. J. Bell and T. J. Sejnowski, "The "independent components" of natural scenes are edge filters," *Vision Research*, vol. 37, no. 23, pp. 3327 – 3338, 1997.

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

[14] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a memristor-based spiking neural network immune to device variations," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, Jul. 2011, pp. 1775–1781.

[15] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, pp. 1–9, Aug. 2015.

[16] A. Tavanaei and A. S. Maida, "Bio-inspired spiking convolutional neural network using layer-wise sparse coding and STDP learning," *CoRR*, vol. abs/1611.03000, Jun. 2017.

[17] P. Falez, P. Tirilly, I. M. Bilasco, P. Devienne, and P. Boulet, "Multi-layered spiking neural network with target timestamp threshold adaptation and STDP," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, Jul. 2019, pp. 1–8.

[18] L. Huang, D. Yang, B. Lang, and J. Deng, "Decorrelated batch normalization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018.

[19] K. S. Burbank, "Mirrored STDP implements autoencoder learning in a network of spiking neurons," *PLoS computational biology*, vol. 11, no. 12, pp. 1–25, 2015.

[20] B. Olshausen and D. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–9, 07 1996.

[21] S. Thorpe, A. Delorme, and R. Van Rullen, "Spike-based strategies for rapid processing," *Neural Networks*, vol. 14, no. 6, pp. 715–725, Jul. 2001.

[22] A. N. Burkitt, "A review of the integrate-and-fire neuron model: I. homogeneous synaptic input," *Biological Cybernetics*, vol. 95, no. 1, pp. 1–19, Mar. 2006.

[23] P. Falez, "Improving spiking neural networks trained with spike timing dependent plasticity for image recognition," Ph.D. dissertation, Université de Lille, Oct. 2019.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE