Training inhibitory connections between local receptive fields in spiking neural networks

D. Gafni [a*] D.V. Nekhaev [b] V.A. Demin [b]
[(a)] Lomonosov Moscow State University, Moscow, Russia
[(b)] National Research Center «Kurchatov Institute», Moscow, Russia
[(*)] Corresponding author. Email adress: *danielgafni16@gmail.com*

Acknowledgments

This work was partially supported by the Russian Science Foundation (project № 20-79-10185) in the studying of inhibitory connections training and by the Russian Foundation for Basic Research (Grant No. 18-29-23041) in the part of comparison of convolrtional and locally connected achitectures.

Abstract

Standard learning methods (based on the error backpropagation) used in artificial neural networks (ANNs) are difficult to apply to spiking neural networks (SNNs) due to discrete and time-distributed nature of the events they generate. Although such projects exist, local learning algorithms, which are more biologically correct, capable of performing unsupervised learning and more energy efficient (if implemented on specialized hardware), are of greater interest.

SNN's local learning rules are significantly understudied in comparison to ANNs, and because of this, SNN-based alrorithms produce lower inference quality metrics. Thus, the study of SNN learning algorithms, like spike-timing-dependent-plasticity (STDP), for use in various SNN architectures is an important problem.

The introduction of inhibitory connections (with negative weights) allows to achieve a better distinction of features learned by the neurons. In this work it is shown that learning competition relationships in a locally connected (LC) SNN potentially leads to higher pattern recognition accuracy due to the organization of competition and cooperation between groups of neurons. It is also shown that the LC SNN produces better results than a convolutional SNN.

Keywords

Spiking Neural Networks • Unsupervised Learning • Inhibitory Connections • Competition • Local Connection • Neuroscience

Declarations

Funding	This work was partially supported by the Russian Science Foundation (project № 20-79-10185) in the studying of inhibitory connections training and by the Russian Foundation for Basic Research (Grant No. 18-29-23041) in the part of comparison of convolrtional and locally connected achitectures.
Data, Materials, Code	https://github.com/danielgafni/bachelor
Conflicts of Interest	The authors declare no conflict of interest.

Introduction

Modern abstract neural networks (represented by combinations of linear layer transformations and non-linear activation functions) do an excellent job at solving many machine learning problems (Wan et al. 2013; Khan et al. 2020). However, training them is a laborious process that requires large computational resources. Typically, training is conducted using hundreds or thousands of examples and can take months. Both training and inference of formal neural network algorithms are far from being effective (Edwards 2015). This is due both to the physically separate storage of connection weights and neuron activations, and to the calculations themselves, which are tensor in nature, associated with vector-matrix and matrix-matrix multiplication of floating point numbers. Modern CPUs are not optimized for this kind of computing. GPUs — architectures originally created for working with computer graphics, and therefore more suitable for tensor calculations — are much better at these tasks than traditional CPUs, but they do not perform with the desired efficiency either. The number of parameters in modern formal neural networks can reach tens (Alom et al. 2019; Khan et al. 2020) and even hundreds of billions (the latest OpenAI language model GPT-3 (Brown et al. 2020)).

Spiking neural networks (SNNs) are an alternative promising neuromorphic algorithm and have several advantages over formal neural networks.

- SNNs allow a richer dynamic coding of patterns in continuous time, which is usually not available for formal neural networks (Ismail Fawaz et al. 2019). Thus, SNNs are of greatest interest for solving time series problems (video and sound streams processing, speech recognition, decision making).
- Local algorithms, which use only information from the neurons connected by this connection to update the weight of each connection (Markram, Gerstner, and Sjöström 2011; Pehlevan 2019; Baldi and Sadowski 2016), can be used to train SNNs. On the contrary, when training formal neural networks, information about all neural connections between this connection and the output of the network is used to update the weight of each connection. Thus, SNNs learning algorithms on their own are far more efficient than formal neural network learning algorithms.
- Moreover, these algorithms can use unsupervised learning, which does not require manual data labeling, as in the case of backpropagation methods.
- SNNs can be implemented on specialized neuromorphic processing units (both based on digital elements (Merolla et al. 2014; Davies et al. 2018; "Akida Neural Processor System-on-Chip" 2020) and using hybrid digital-analog circuits (BrainScales, (Painkras et al. 2013)), including those based on memristors (Wang et al. 2020), which have ultra-low power consumption (achieved by reducing the number of acts and the average length of signal transmission, as well as the spatial and temporal sparseness of spikes in the network). This implementation, together with the algorithmic advantage of the SNNs, also provides significant performance gains. (Ma et al. 2017; Tang, Shah, and Michmizos 2019).
- SNNs, to a greater extent than formal neural networks, biologically correctly model the interactions of neurons, which can be used for biological simulations of the nervous system for the purposes of studying not only bioinformatics, but also biophysical and medical aspects of its functioning.

It is of interest to study well-known ANN architectures in application to SNN, such as convolutional, locally connected and fully connected (Khan et al. 2020) architectures. Moreover, the introduction of additional recurrent inhibitory connections contributes to a better separation of features in the learning process (Demin and Nekhaev 2018; Nekhaev and Demin 2020a). A fully connected network is the simplest model that can be used as a benchmark. Convolutional and locally connected architectures use convolution — an operation that allows to extract important features more efficiently (comparing to a fully connected network). The locally connected architecture is especially interesting, because it can be easily implemented in hardware, since, unlike a convolutional network, it does not use shared synaptic weights inside a separate feature map. A locally connected architecture has significantly fewer parameters than a fully connected architecture and allows to train a unique set of features for each receptive field, in contrast to convolutional networks. At the same time, convolutional networks have translational invariance, a useful property for image processing (spatially correlated data). The presence of competition in the SNN makes it possible to make their features more independent. Therefore, for experiments in this work, we chose locally connected SNNs with inhibitory connections, and as basic models, convolutional and fully connected networks with inhibitory connections. In this paper we:

i compare this architecture with a Convolution Spiking Neural Network (CSNN) and a Fully Connected Spiking Neural Network (FCSNN).

ii study the effect of training the inhibitory connections (Demin and Nekhaev 2018; Nekhaev and Demin 2020a; Bouvier et al. 2019) between neurons on the accuracy of the image classification problem of handwritten digits from the MNIST (Lecun et al. 1998) dataset for the architecture of a locally connected network (Locally Connected Spiking Neural Network, LCSNN) (Saunders et al. 2019);

1 SNN with fixed inhibitory connections between local receptive fields

For different networks architectures comparative analysis, a classical machine learning problem was chosen — the problem of classifying images of handwritten digits from the MNIST dataset. MNIST consists of labeled training and test subsets of 60,000 and 10,000 images. The images have 28 × 28 resolution and are black and white. Due to the need to calibrate the networks (see below), the original training subset was split into 50,000 images for training (training dataset) and 10,000 images for calibration (calibration dataset).

The images are also cropped so that only the center 20 × 20 pixels are used.

1.1 Competition between local receptive fields in SNN

In this paper, networks are studied using locally connected, fully connected and convolutional layers. In a fully connected layer, each neuron is connected to each neuron from the previous layer. In the convolutional layer, neurons are divided into channels. The neurons in one channel have shared weights, but each neuron is only connected to a certain region (rectangular in the case of a two-dimensional layer) (Z. Li et al. 2020) — receptive field, or patch. The architecture of the locally connected layer is the same, but each neuron has its own unique weights (Saunders et al. 2019).

The neural network architecture of LCSNN is inspired by the structure of the visual cortex of the brain. The Y network layer consists of n channels, each of which is locally connected to the X layer of neurons, and neurons with common receptive fields are connected by inhibitory connections (Fig. **1**).

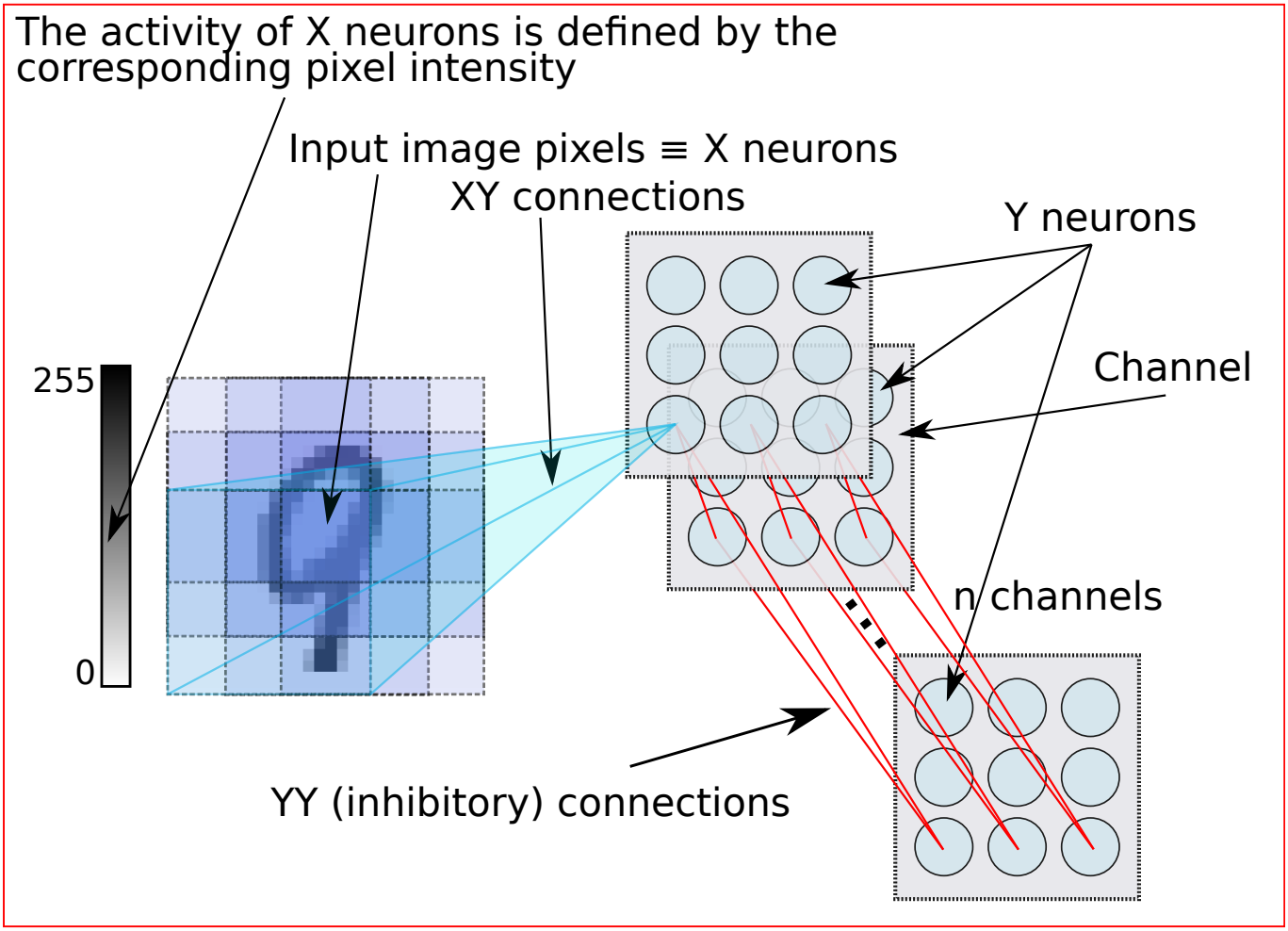


Figure 1. LCSNN architecture

Such connections have negative weights, which means they negatively affect neuron activity. Neurons that do not have a common patch (and therefore respond to different areas of the image) do not compete with each other. Inhibitory connections are introduced to improve the separation of neurons by learned features. A similar architecture can be built for networks with convolution or fully connected layers. We studied networks with the number of Y neurons ≈ 100 — 900 and folowwing convolutions parameters: kernel size of 8 or 12 and stride of 4. Since the MNIST images were cropped to a size of 20 ×20, the number of neurons per the channel was 3 or 4.

The Adaptive Integrate-And-Fire (ALIF) neuron model has been chosed for this study. The dynamics of the neuron potential in this model is given by the equationof

$$\tau_v \frac{\mathrm{d}v(t)}{\mathrm{d}t} = -v(t) + v_{rest} + I(t) \cdot R, \tag{1}$$

where $I(t)$ is the current, accumulated by time moment t, v_{rest} — relaxation level, τ_v — simulation time constant, a R — unit coefficient, numerically equal to one.

Activation threshold v_{thresh} of an ALIF neurone is not static. It slightly increases with every spike, then relaxing to its initial value θ_o. The dynamics of the activation threshold is given by:

$$v_{thresh} = \theta_0 + \theta(t), \tag{2}$$

where θ_0 is the initial activation threshold, $\theta(t)$ — an adaptive additive to the activation threshold after each spike generation, which is calculated from the condition

$$\tau_v \frac{\mathrm{d}\theta(t)}{\mathrm{d}t} = -\theta(t) \tag{3}$$

After the generation of each spike, a short refractory period occurs, when for time $t_{refract}$ the potential of the neuron remains at the level of v_{reset}. The initial connection weights are generated randomly from a uniform distribution.

1.2 Forward connections training

To reduce the number of model parameters, images are cropped to 20 × 20 pixels. The edges of the images are often almost empty, so this operation has little or no effect on the amount of information, available to the network. For each image, X spikes are generated using a Poisson distribution with a mathematical expectation value proportional to the corresponding pixel intensiveness. Connections XY are trained according to the STDP (Markram, Gerstner, and Sjöström 2011) rule. This is a biologically inspired rule of unsupervised learning. When the presynaptic spike (pre-spike) is received and the post-spike is emitted, the weight w of the corresponding connection is increased by Δw, where

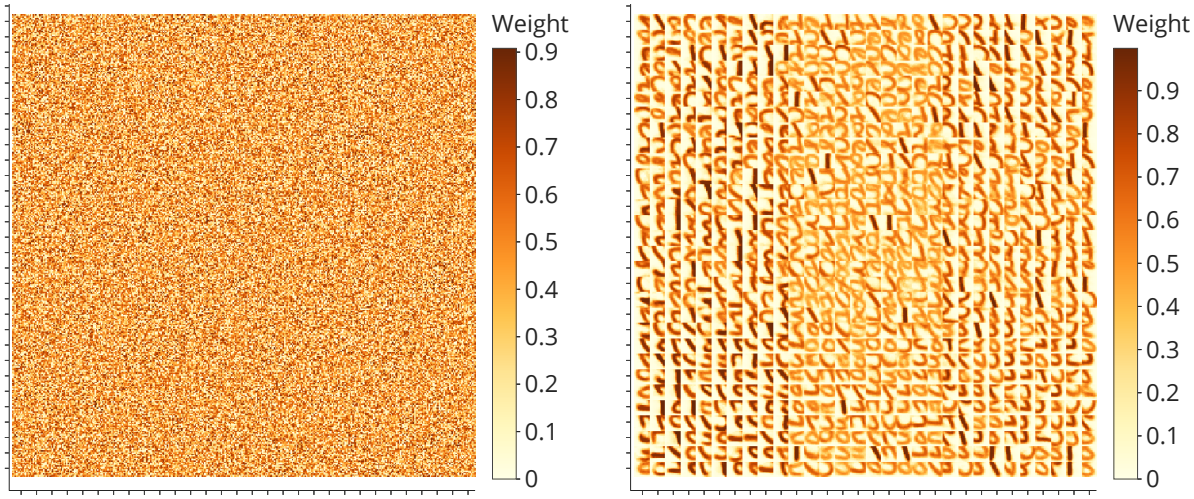
$$\Delta w = \begin{cases} A_+ \cdot e^{-\frac{t_{pre}-t_{post}}{\tau_+}}, t_{pre} - t_{post} > 0 \\ A_- \cdot e^{-\frac{t_{pre}-t_{post}}{\tau_-}}, t_{pre} - t_{post} < 0 \end{cases} \tag{4}$$

Let's notice, that

$$\begin{cases} A_+ > 0 \\ A_- < 0 \end{cases}$$

Thus, in the process of learning, the weights of connections where pre-spikes are systematicly recieved right before the emmision of post-spikes are increased, and decreased if vice versa. After such training, the neurons begins to actively react to pre-spikes received from neurons with larger connection weights. This leads to the neuron weights learning certain features (Fig. **2**). If these pre-spikes are recieved in a short enough time period, they might cause a new spike. The opposite rule (with A_+ and A_- having the opposite signs) is called the Anti-STDP (**Anti-STDP**) rule. It is used to train inhibitory connenctions in section **2** of this paper.

After each training iteration, the weights are normalized to keep their sum equal to a certain constant. This is done to forbid the weights to grow too large. The value of the normalization constant is an important model hyperparameter that has to be carefully selected for each specific network architecture.



a **b**

Figure 2. Visualization of XY connection weights of a 900 Y neurons (100 neurons for each of the 9 patches) network — a 360 × 360 matrix. The weights of one neuron correspond to a 12 × 12 square. The weights of the Y neurons are grouped by patch — the lower left 120 × 120 square (10 × 10 neurons) corresponds to the lower left patch, the center square corresponds to the center patch, and so on. Weights are initialized randomly (**a**), and after 1000 training iterations (**b**) the neurons learn different features, resembling digits elements

1.3 Spiking neural network activity interpretaion

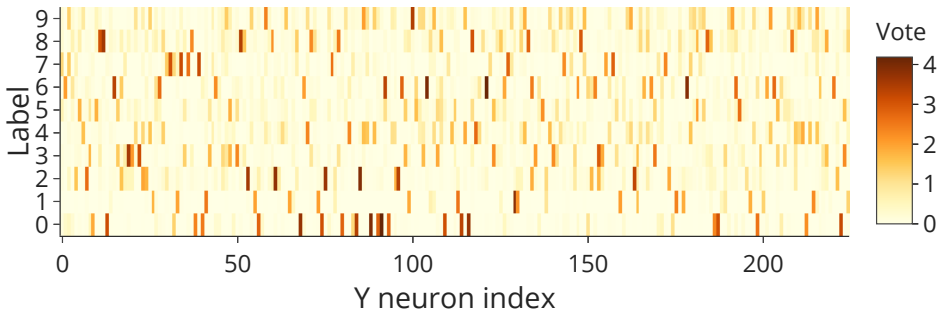
Several methods were used to interpret the activity of neurons in the Y layer (that is, to correlate their activity with a certain class of recognizable digits): **patch voting**, **global voting**, **preselection by spikes voting** or a **linear classifier**. The first three methods have the advantage of being simple. However, the linear classifier significantly outperforms them in performance.

For the first three methods, it is necessary to calibrate the «votes» of neurons. Each neuron of the Y layer is assigned 10 numbers (votes) for each digits class (from 0 to 9). The neuron-digit vote is calculated as the average number of spikes produced by the neuron in responce to the observation of this digit images by the network (throughout the entire time of the simulation) —

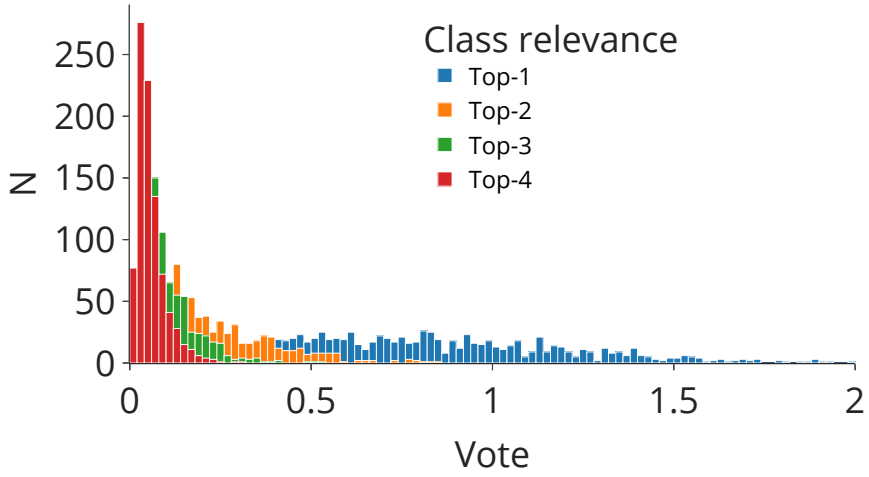
$$vote_i = \frac{\sum_1^{n_i} \sum_0^{t_{max}} s_t}{n_i},$$

where s_t takes the value 0 or 1 depending on the presence of a spike at a given time, n_i is the size of the calibration sample for the digit i (we used $n_i = 1000$).

For all networks, this calibration process was used for 10,000 examples from a calibration sample especially selected from the test part of the MNIST dataset, as noted above. Note that calibration is nota part of the network training, since it is only included in the algorithm for interpreting the behavior of the SNS. The votes are used as a measure of confidence of the neuron in each of the digits classes. The example votes are shown in Fig. **3**.



a



b

Figure 3. Votes visualization for a network with 225 Y neurons. All Y neurons votes are shown in (**a**). High values correspond to a large specialization on the corresponding class. It can be seen that almost all neurons have a nonzero vote value, therefore, almost all neurons in the network are active. Votes distrivution for the 4 most relevant classes (for each neuron) are shown in (**b**). It can be seen that the vote value (average number of spikes per class) drops significantly with relevance decreasing

Further, the product of the number of neuron spikes by its vote will be called *score*.

In **global voting**, the network's response is considered to be the class with the highest *score* among all neurons.

In **patch voting** the network response is considered to be the class with the highest *score* among neurons with the highest *score* per patch — similiar to a maxpool operation in ANNs.

Preselection by spikes voting is done in a similiar fassion, but comparing the number of spikes, not *scores*.

When using a **linear classifier**, the sums of spikes of individual Y layer neurons are fed as features to te classifier, which is trained to predict the true image labels. As in other methods of interpreting the network activity, classifier training is performed on a separate calibration sample.

To evaluate the interpretation algorithm, accuracy is used — the ratio of the number of correctly recognized objects from a test sample to the size of the test sample. In this work, the test sample size (when measuring the accuracy for individual networks) is 10,000.

Accuracy learning curves (Fig. **4**) were built for various interpretation algorithms. The accuracy was measured every 250 training iterations. To calibrate the interpretation algorithm at each step a calibration sample of 5000 was used, while to measure the accuracy, a test sample of 1000 was used.

After several thousand training iterations, the accuracy reaches a plateau. It is also noticeable that the three voting methods have very similar accuracy, while the linear classifier performes significantly better. The accuracy of even an untrained network can reach 70% due to the fact that even with random weights initialization, different feature maps are formed, with some being a little better at recognizing certain patters, which can be amplified by the interpretaion alrorithm. It is known from (Saunders et al. 2019) that for a sufficiently large number of parameters, a locally connected network outperforms a convolutional and fully connected network in learning speed, since more parameters are updated with each training iteration (at least one neuron per patch is active). In the present study, no such effect is observed, since the presented learning curves are constructed for insufficiently large networks. Interestingly, the use of a linear classifier does not improve the recognition accuracy for a fully connected network — most likely due to the

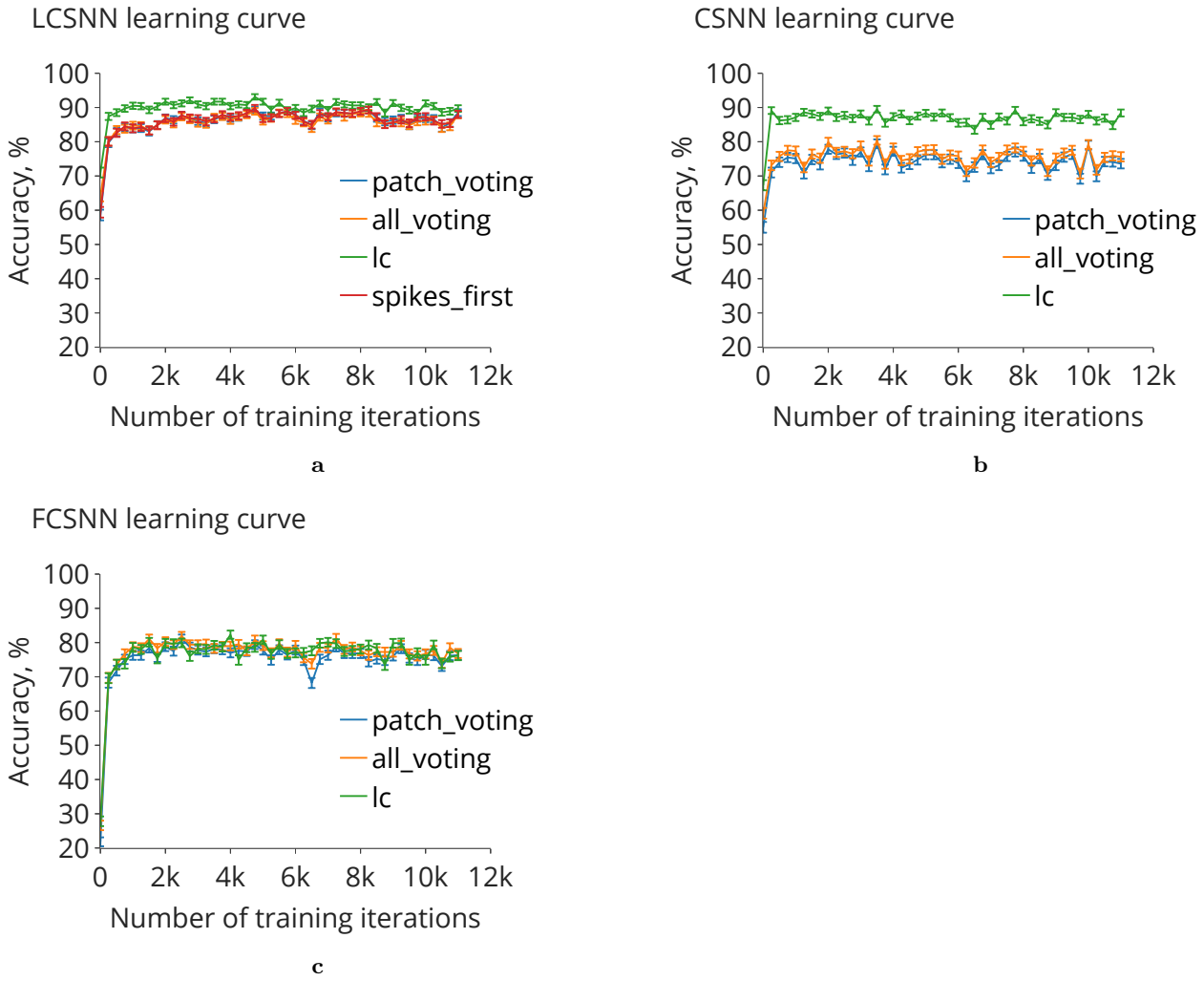


Figure 4. Comparison of learning curves of different architectures of SNNs with competition: (**a**) LCSNN, (**b**) CSNN, (**c**) FCSNN. The results are evaluated with different algorithms for the interpretation of activity. The **preselections by spikes** algorithm was only checked for LCSNN, because it achieves practically same results as **patch voting**. Standard deviations are plotted as errors. The networks have 100 channels (for a FCSNN the channel consists of a single neuron), the convolution kernel size (k) for the LCSNN is 12, and 8 for the CSNN

selection of voting features throughout the image, in contrast to convolutional and locally connected architectures.

The linear classifier surpasses the LCSNN- and CSNN-based voting algorithms in accuracy, since it is a generalization of voting in the sense that it also uses the sum of the products of weights/votes by neurons activity. However, when training a linear classifier effective optimization algorithms (including gradient) are being used, while when voting we achieve practically the same goal (calculating satisfactory votes/weights) with a heuristic — the averaged activity of neurons by classes.

1.4 Comparison of convolution and local connections effectiveness

Experiments were carried out to measure the accuracy of networks with different architectures. For convolutional and fully connected networks, inhibitory connections similar to LCSNN were introduced. Note that using a linear classifier as an interpretation algorithm, it was possible to achieve 95% accuracy for a locally connected network of 1000 channels with kernel size of 12.

Table 1. Comparison of different architectures. For each configuration, the accuracy was measured $N = 5$ times. The table shows the arithmetic mean of the accuracy and its standard deviation. The «kernel» corresponds to the number k in the size of the convolution $k \times k$

	Conviguration					Accuracy, %	
N	Architecture	Channels	Kernel	Parameters	Y neurons	Method [1]	Method [2]
1	**LCSNN**	**1000**	**12**	**10287000**	**9000**	$\mathbf{92.3 \pm 0.7}$	$\mathbf{95.1 \pm 0.5}$
2	LCSNN	100	12	218700	900	87.5 ± 0.9	91.5 ± 0.6
3	LCSNN	100	8	260800	1600	82.9 ± 0.6	88.1 ± 0.7
4	**LCSNN**[3]	**25**	**12**	**37800**	**225**	$\mathbf{82.3 \pm 1.0}$	$\mathbf{88.2 \pm 0.6}$
5	LCSNN	25	12	37800	225	80.1 ± 1.0	85.5 ± 0.8
6	LCSNN	25	8	35200	400	73.6 ± 1.0	80.3 ± 0.7
7	CSNN	169	12	279864	1521	79.2 ± 1.6	85.7 ± 1.4
8	CSNN	81	12	69984	729	77.2 ± 1.7	83.1 ± 1.2
9	CSNN	100	8	164800	1600	77.4 ± 1.9	82.1 ± 1.3
10	CSNN	25	12	9000	225	65.8 ± 0.7	77.1 ± 0.6
11	CSNN	25	8	11200	400	63.1 ± 1.2	75.8 ± 0.5
12	FCSNN	100	20	49900	100	81.4 ± 0.9	82.1 ± 0.8

It can be seen (№2 and №7 in the **1** table) that the locally connected network surpasses the convolutional network in accuracy even if the number of parameters of the latter is slightly larger. Also, having too small convolution kernel size leads to lower network accuracy, because it leads to learning of less essential features. Indeed, in traditional deep learning based, either shallow networks with large convolution kernels are used, or deep networks with small convolution kernels.

It should be noted that SNNs can achieve significantly higher recognition accuracy on MNIST. It can be achieved by: (i) networks with a significantly larger number of parameters (for example, with an increased number of channels); (ii) deeper networks; (iii) more efficient learning algorithms (for example, supervised learning or using the contrastive loss function (Hadsell, Chopra, and LeCun 2006)). At the same time, achieving the highest possible accuracy was not the goal of this study, which is primarily aimed at comparing various SNN architectures.

Table 2. Other studies on spiking neural networks with MNIST results

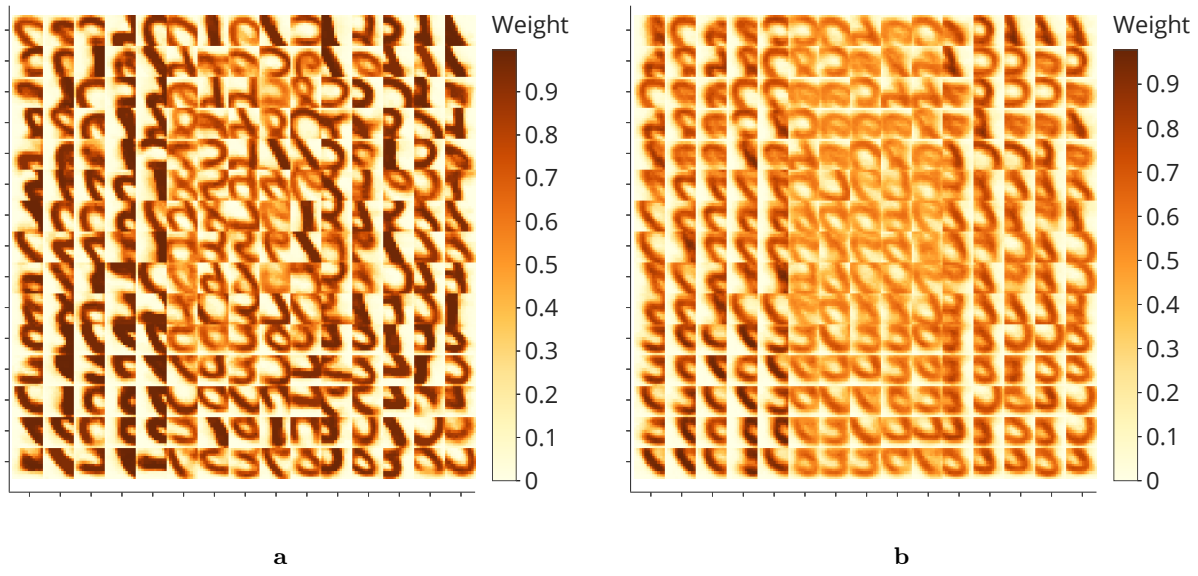
Paper	Architecture	Training	Accuracy, %
This study	Local + competition	Unsupervised	95.1 ± 0.5
(Saunders et al. 2019)	Local + competition	Unsupervised	95.07 ± 0.63
(Diehl and Cook 2015)	Fully connected + competition	Unsupervised	95
(Demin and Nekhaev 2018)	Fully connected + competition	Supervised / Semi-supervised	95.4 / 72.1
(Tavanaei and Maida 2017)	Convolutional	Semi-supervised	96.95 ± 0.08
(Lee et al. 2018)	Convolutional	Semi-supervised	99.28 ± 0.10
(Tavanaei, Kirby, and Maida 2018)	Convolutional	Semi-supervised	97.20 ± 0.07

The result obtained in this work are comparable to the result of (Saunders et al. 2019). Note that the results achieved with unsupervised training are only slightly inferior to the results achived by supervised training. Note, that our best locally connected network with trainable competitions had 10^7 parameters (of which $1 \cdot 10^6$ are forward connections and $9 \cdot 10^6$ inhibitory connections), which is less than $4.6 \cdot 10^7$ (of which $0.5 \cdot 10^7$ forward conenctions and $4.1 \cdot 10^7$ compettition connections) for a fully connected network with competition from (Diehl and Cook 2015). In addition, the network from (Diehl and Cook 2015) was trained for $1 \cdot 10^6$ iterations, while in this study the number of training iterations did not exceed 5000.

[1] Best voting algorithm
[2] Linear classifier
[3] Network with trainable inhibitory connections

2 Inhibitory connections training

Inhibitory connections YY greatly affect the training of forward connections XY. Large absolute values of inhibitory weights contibute to greater variability and specialization in training Y neurons, as for each receptive field similar weights XY neoruns are simultaneously active (Fig. **5a**). On the contrary, inhibitory weights of small absolute values do not allow neurons to specialize effectively (Fig. **5b**).



a **b**

Figure 5. The effect of competition on XY connections training. (**b**) shows poorly specialized neuros with similar features — the result of training with competiiton weights set to -10, while (**a**) features highly specialized neurons with diverse features, trained with inhibitory weights set to -100

All SNNs discussed in this paper up to this point had fixed competition. In this regard, the question arised of how the trianing of inhibitory weights affects the accuracy of the network.

To answer this question, in order to update the YY connections, the Anti-STDP rule was chosen (the rule opposite in the signs of A_+ and A_- to standard STDP). For different values of the parameters of this rule, different distributions of inhibitory weights were obtained. The initial values of the weights were set from a uniform distibution from 0 to a certain negative values. Experiments were carried out with LCSNN networks.

Table 3. Anti-STDP parameters

Figure	τ_+, ms	τ_-, ms	A_+	A_-
6a, **7a**	14.7	14.2	-0.5	1.5
6b, **7b**	5.4	15.1	-1.2	0.6
6c, **7c**	17.6	24.5	-1.9	1.6
6d, **7d**	17.7	16.5	-0.1	1.6

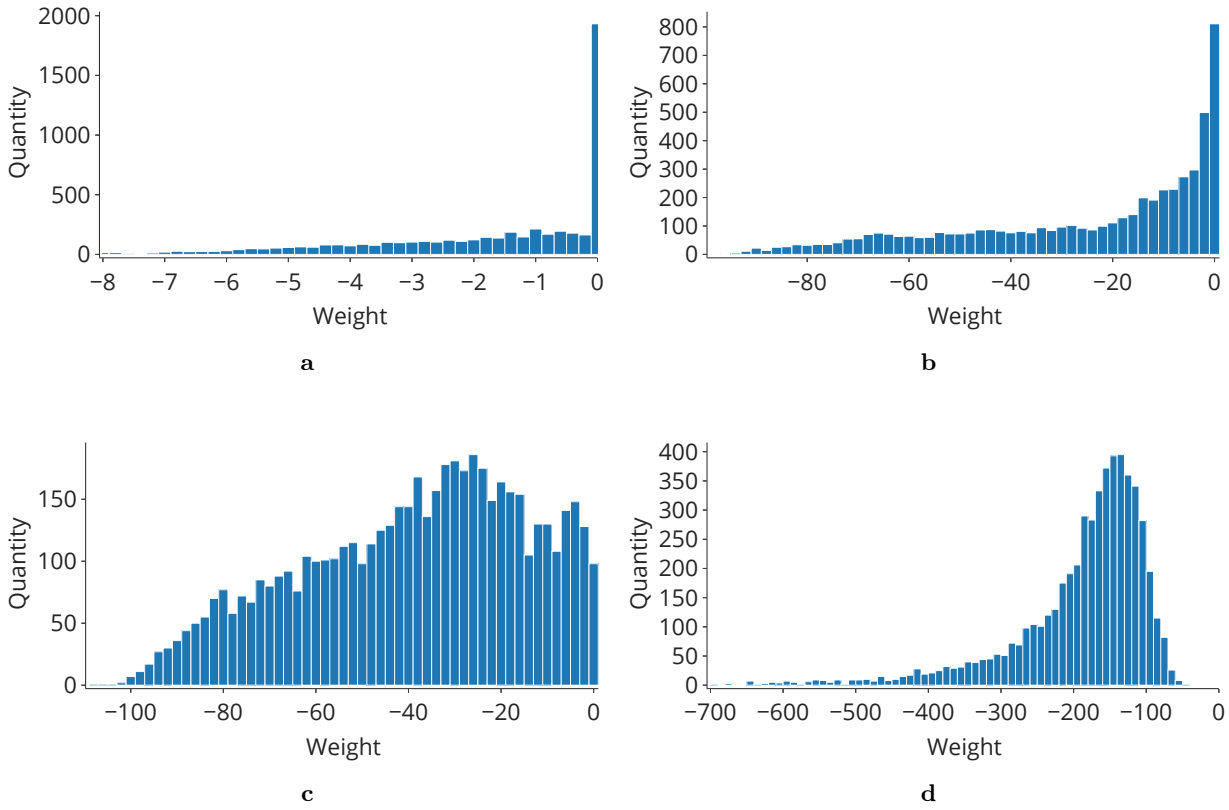


Figure 6. Different inhibitory (YY) weights distributions: weakest competition (**a**), weak competition (**b**), medium competition (**c**) and strong competition (**d**)

It can be seen that the accuracy of the network increases when the distribution of inhibitory weights is shifted towards larger negative values. Note again that the goal was not to find the parameters that provide maximum accuracy, but to study the influence of competition training on the accuracy of the network with a given configuration of the remaining parameters.

It is noteworthy that not all YY connections end up with a large value (Fig. **6d**). This is due to the fact that neurons specializing in substantially different features do not need competition, since they do not exhibit high activity at the same time.

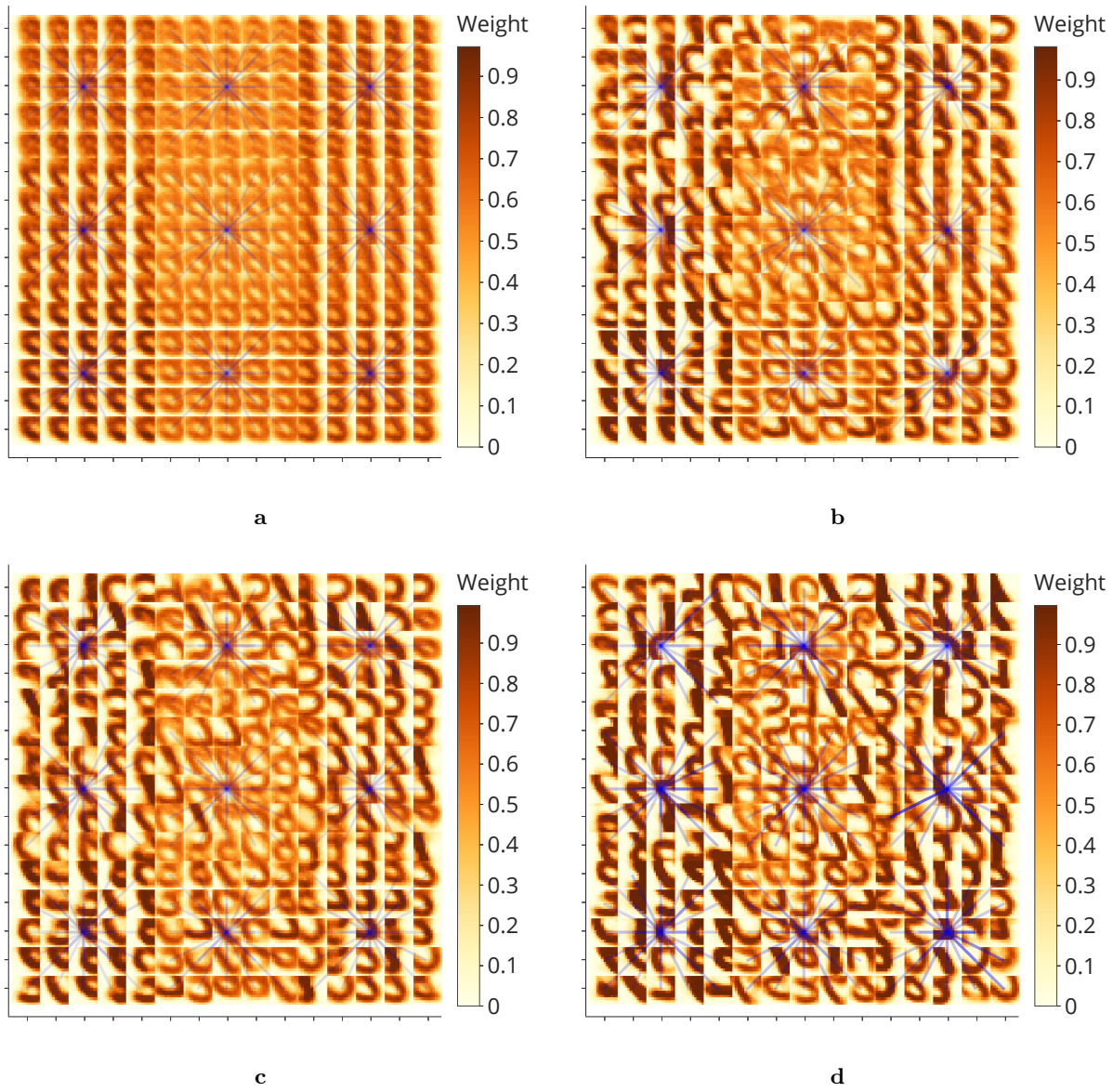


Figure 7. Visualization of inhibitory weights on top of XY weights for the networks from Fig. (**6**). Inhibitory weights are shown only for one center neuron in each receptive field to avoid cluttering the visualization. The saturated blue color corresponds to the inhibitory weights of large absolute value (the arithmetic mean between the weights W_{ij} and W_{ji} is used). Figures **a-d** correspond to the same networks as on Figures **6a-d** (from weakest to strongest competition). It can be seen that similar features have higher inhibitory connections weights than different

Competition training was carried out only on networks of 25 channels (225 neurons), since its simulation requires large computational resources.

During the optimization of the models, the accuracy of a large number of LCSNNs was measured with various hyperparameter configurations (Fig. **8**). It can be seen that configurations with high accuracy (dark lines) are localized in a manner that encourages negative weight accumulation for larger time differencies $t_{post} - t_{pre}$ between -post and -pre spikes (for the Anti-STDP weight update rule).

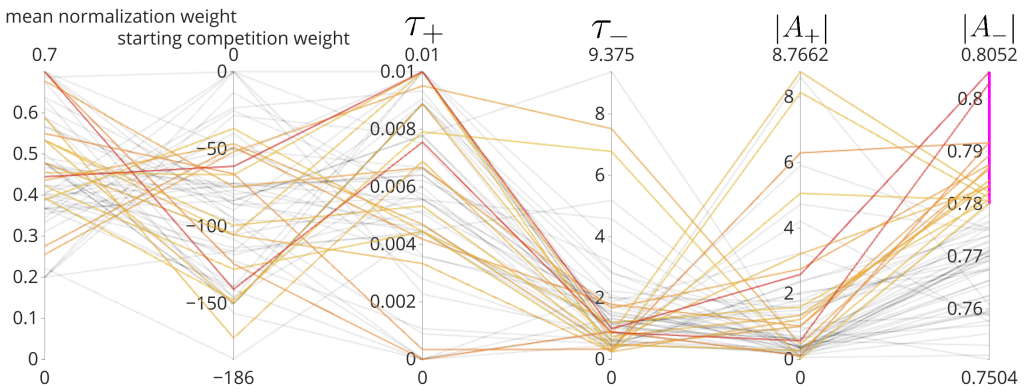


Figure 8. Visualization of the achieved LCSNN recognition accuracy when using various combinations in the hyperparameters space. The parameters «normalization weight», «initial inhibitory weight» and «weight decay» are responsible for the weights values, and the next two parameters are combinations of Anti-STDP parameters

2.1 Trained inhibitory weights analysis

In addition, inhibitory weights clamping (during training) experiments were carried out. It turned out that the inhibitory weights in the entire range of their variation are important for the effective LCSNN functioning, since both the upper and lower clamps negatively affect the classification accuracy (Fig. **9**). This is due to the fact that high competition contributes to greater specialization of neurons and therefore is useful, and low competition allows neurons to cooperate and recognize digits together (including ensuring the accumulation of more statistics for calibrating the voices of neurons).

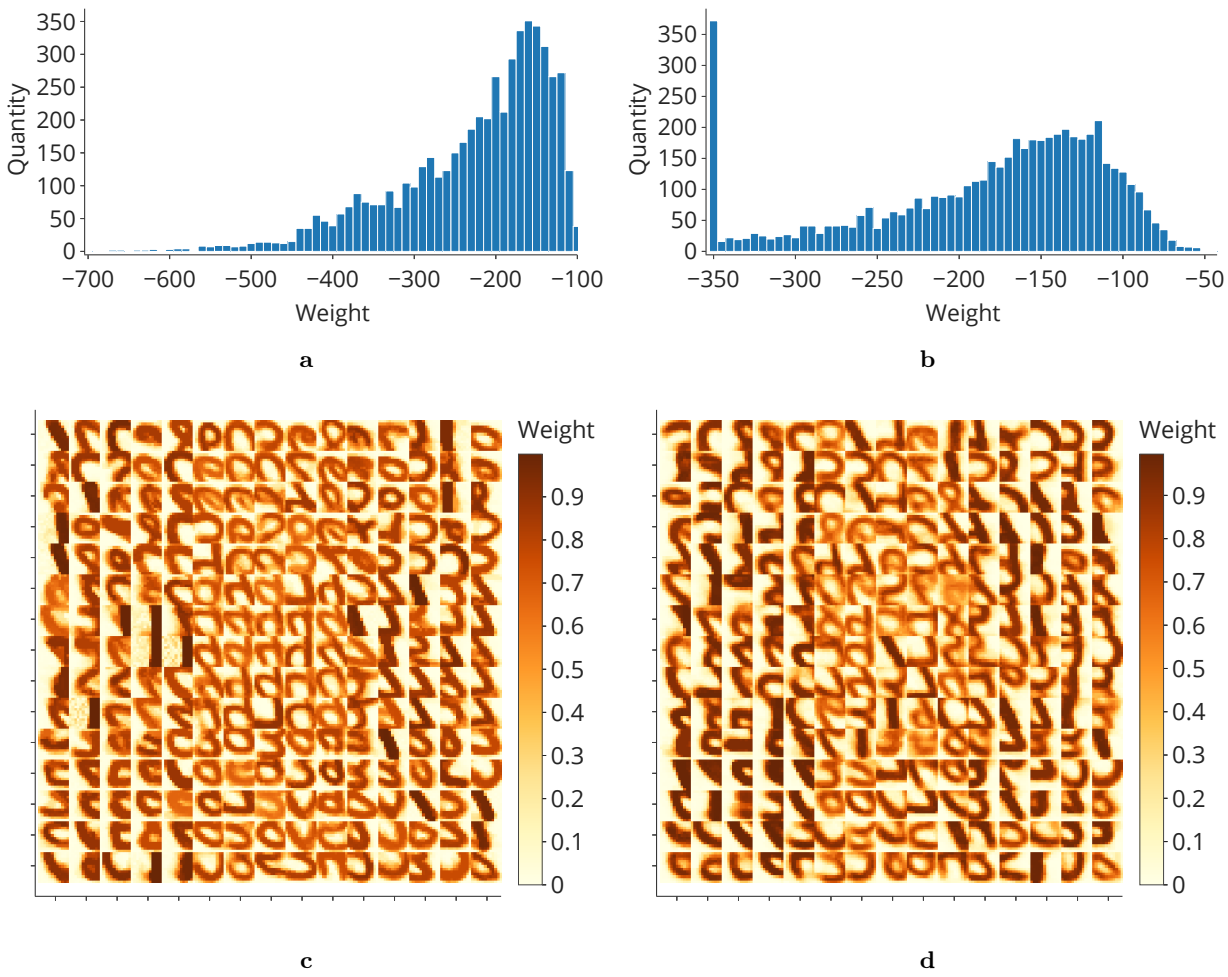


Figure 9. Impact of limiting the values of inhibitory weights on accuracy. Figures (**c**, **d**) show XY weights and Figures (**a**, **b**) show the inhibitory (YY) weights distributions. The inhibitory weights were clamped (during training) to $[-350, 0]$ (**a**,**c**) and to $(-\infty, -100]$ (**b**,**d**). The accuracy of a network with the same hyperparameters, but without the competition limitation is 82.3%, which is significantly higher than the accuracies of the networks presented in this Figure. Its weights XY are shown in Fig. **7d**, and the distribution of its inhibitory weights is shown in Fig. **6d**. In both cases the networks develop more blurred features comparing to the network with unrestricted competition in Fig. **6d**

Competition training made it possible to achieve an accuracy slightly (by 2 %) higher than the accuracy of a network of the same configuration, but with a fixed competition with inhibitory weights equal to -50 (Table **1**, №4 and №5).

3 Discussion

This study demonstrates that a locally connected network is a promising spiked neural network architecture suitable for efficient implementation on a specialized neural hardware. Indeed, the ability to quickly plateau the learning curve allows the LCSNN to be used for training on relatively small training samples (3000-5000). The use of an additional activity interpretation algorithm, such as a linear classifier, can improve the efficiency of the network. Note that in this case the very features of the objects being recognized are learned by the network in an unsupervised fashion. The question remains about the possibility of obtaining similar results without using supervised learning to interpret network activity.

The main advantage of LCSNN is, as has been shown, that the locally connected network outperforms the convolutional in accuracy with approximately the same or slightly larger number of parameters. Presumably, this is due to the richer statistics of patches (receptive fields) that neurons within each channel respond to, compared to one receptive field per channel in the convolutional architecture. The price for this is the increased number of network parameters, but is not a critical factor due to the local (as opposed to fully connected) topology of interneural contacts. The local architecture also allows to save information about the localization of the selected feature in space and therefore is a reasonable choice for the implementation of unsupervised trained SNNs. This choice also determines its overall efficiency (in terms of performance, accuracy and power consumption at a certain density of elements) in

a hardware implementation, in which each weight is usually represented by one or more physical elements (mainly SRAM cells (Merolla et al. 2014; Davies et al. 2018) or memristors (Y. Li et al. 2018)). The training of inhibitory connections slightly increases the accuracy of the LCSNN-based algorithm, but it is not so critical as to justify the use of such an expensive operation as the introduction and training of a significant number of inhibitory connections with finite weights of different values. At the same time, this conclusion is probably only valid only for small networks, like the one studied in this paper. In our opinion, the selection of optimal algorithms and parameters for training competition for deep SNNs deserves a more thorough study, in which the balance of competition and cooperation of neurons within each layer can lead to the formation of clusters from classes of supplied images, which contributes to building a semantically-like hierarchy of learned features and their combinations, that is, the final images (Nekhaev and Demin 2020b). Potentially, such a hierarchy of images can lead to a significant increase in the quality metrics.

Conclusion

It was shown that for the problem of pattern recognition on the MNIST dataset, the locally connected architecture with competition is superior to the convolutional one with an equal number of parameters. It was also shown that training inhibitory weights can slightly improve the quality of the model, and in the resulting distribution of inhibitory weights, both with high values (strong competition) and low (cooperation) values of weights are important.

References

[1] "Akida Neural Processor System-on-Chip" (Apr. 2020). URL: https://brainchipinc.com/akida-neuromorphic-system-on-chip/.

[2] M. Z. Alom et al. "A State-of-the-Art Survey on Deep Learning Theory and Architectures". *Electronics* 8 (Mar. 2019), p. 292. DOI: 10.3390/electronics8030292.

[3] P. Baldi and P. Sadowski. "A theory of local learning, the learning channel, and the optimality of backpropagation". *Neural Networks* 83 (Nov. 2016), pp. 51–74. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2016.07.006. URL: http://dx.doi.org/10.1016/j.neunet.2016.07.006.

[4] M. Bouvier et al. "Spiking Neural Networks Hardware Implementations and Challenges". *ACM Journal on Emerging Technologies in Computing Systems* 15.2 (June 2019), pp. 1–35. ISSN: 1550-4840. DOI: 10.1145/3304103. URL: http://dx.doi.org/10.1145/3304103.

[5] T. B. Brown et al. "Language Models are Few-Shot Learners" (2020). arXiv: 2005.14165 [cs.CL].

[6] M. Davies et al. "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning". *IEEE Micro* PP (Jan. 2018), pp. 1–1. DOI: 10.1109/MM.2018.112130359.

[7] V. Demin and D. Nekhaev. "Recurrent Spiking Neural Network Learning Based on a Competitive Maximization of Neuronal Activity". *Frontiers in Neuroinformatics* 12 (Nov. 2018), p. 79. DOI: 10.3389/fninf.2018.00079.

[8] P. Diehl and M. Cook. "Unsupervised learning of digit recognition using spike-timing-dependent plasticity". *Frontiers in Computational Neuroscience* 9 (2015), p. 99. ISSN: 1662-5188. DOI: 10.3389/fncom.2015.00099. URL: https://www.frontiersin.org/article/10.3389/fncom.2015.00099.

[9] C. Edwards. "Growing pains for deep learning". *Commun. ACM* 58 (2015), pp. 14–16.

[10] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality Reduction by Learning an Invariant Mapping". 2 (2006), pp. 1735–1742. DOI: 10.1109/CVPR.2006.100.

[11] H. Ismail Fawaz et al. "Deep learning for time series classification: a review". *Data Mining and Knowledge Discovery* 33.4 (Mar. 2019), pp. 917–963. ISSN: 1573-756X. DOI: 10.1007/s10618-019-00619-1. URL: http://dx.doi.org/10.1007/s10618-019-00619-1.

[12] A. Khan et al. "A survey of the recent architectures of deep convolutional neural networks". *Artificial Intelligence Review* (Apr. 2020). ISSN: 1573-7462. DOI: 10.1007/s10462-020-09825-6. URL: http://dx.doi.org/10.1007/s10462-020-09825-6.

[13] Y. Lecun et al. "Gradient-based learning applied to document recognition". *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[14] C. Lee et al. "Training Deep Spiking Convolutional Neural Networks With STDP-Based Unsupervised Pre-training Followed by Supervised Fine-Tuning". *Frontiers in Neuroscience* 12 (2018), p. 435. ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00435. URL: https://www.frontiersin.org/article/10.3389/fnins.2018.00435.

[15] Y. Li et al. "Review of memristor devices in neuromorphic computing: materials sciences and device challenges". *Journal of Physics D: Applied Physics* 51.50 (Sept. 2018), p. 503002. DOI: 10.1088/1361-6463/aade3f. URL: https://doi.org/10.1088%2F1361-6463%2Faade3f.

[17] D. Ma et al. "Darwin: A neuromorphic hardware co-processor based on spiking neural networks". *Journal of Systems Architecture* 77 (2017), pp. 43–51. ISSN: 1383-7621. DOI: 10.1016/j.sysarc.2017.01.003. URL: http://www.sciencedirect.com/science/article/pii/S1383762117300231.

[18] H. Markram, W. Gerstner, and P. J. Sjöström. "A History of Spike-Timing-Dependent Plasticity". *Frontiers in Synaptic Neuroscience* 3 (2011), p. 4. ISSN: 1663-3563. DOI: 10.3389/fnsyn.2011.00004. URL: https://www.frontiersin.org/article/10.3389/fnsyn.2011.00004.

[19] P. A. Merolla et al. "A million spiking-neuron integrated circuit with a scalable communication network and interface". *Science* 345.6197 (2014), pp. 668–673. ISSN: 0036-8075. DOI: 10.1126/science.1254642. eprint: https://science.sciencemag.org/content/345/6197/668.full.pdf. URL: https://science.sciencemag.org/content/345/6197/668.

[20] D. Nekhaev and V. Demin. "Competitive Maximization of Neuronal Activity in Convolutional Recurrent Spiking Neural Networks" (Jan. 2020), pp. 255–262. DOI: 10.1007/978-3-030-30425-6_30.

[21] D. Nekhaev and V. Demin. "Competitive Maximization of Neuronal Activity in Convolutional Recurrent Spiking Neural Networks" (2020). Ed. by B. Kryzhanovsky et al., pp. 255–262.

[22] E. Painkras et al. "SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation". *Solid-State Circuits, IEEE Journal of* 48 (Aug. 2013), pp. 1943–1953. DOI: 10.1109/JSSC.2013.2259038.

[23] C. Pehlevan. "A Spiking Neural Network with Local Learning Rules Derived From Nonnegative Similarity Matching" (2019). arXiv: 1902.01429 [cs.NE].

[24] D. J. Saunders et al. "Locally Connected Spiking Neural Networks for Unsupervised Feature Learning" (2019). arXiv: 1904.06269 [cs.NE].

[25] G. Tang, A. Shah, and K. P. Michmizos. "Spiking Neural Network on Neuromorphic Hardware for Energy-Efficient Unidimensional SLAM" (2019). arXiv: 1903.02504 [cs.RO].

[26] A. Tavanaei and A. S. Maida. "Multi-layer unsupervised learning in a spiking convolutional neural network" (2017), pp. 2023–2030.

[27] A. Tavanaei, Z. Kirby, and A. S. Maida. "Training Spiking ConvNets by STDP and Gradient Descent". *2018 International Joint Conference on Neural Networks (IJCNN)* (2018), pp. 1–8.

[28] L. Wan et al. "Regularization of Neural Networks using DropConnect". Proceedings of Machine Learning Research 28.3 (June 2013). Ed. by S. Dasgupta and D. McAllester, pp. 1058–1066. URL: http://proceedings.mlr.press/v28/wan13.html.

[29] W. Wang et al. "Integration and Co-design of Memristive Devices and Algorithms for Artificial Intelligence". *iScience* 23.12 (2020), p. 101809. ISSN: 2589-0042. DOI: https://doi.org/10.1016/j.isci.2020.101809. URL: https://www.sciencedirect.com/science/article/pii/S2589004220310063.