Training competition connections between local receptive fields in spiking neural networks

Daniil Gafni, Dmitry Nekhaev, Vyacheslav Demin

Abstract

Standard learning methods (based on the error backpropagation) used in formal 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 — impulses, or spikes. Although such projects exist, local learning algorithms, which are more biologically correct, capable of performing unsupervised learning (without a teacher) and more energy efficient (if implemented on specialized hardware), are of greater interest.

One of the main disadvantages of the SNNs in comparison with the ANNs is the current understudy of local learning rules, and because of this, lower quality metrics (on inference) of algorithms based on SNNs. Thus, the study of SNN learning algorithms, like spike-timing-dependent-plasticity (STDP), for use in various SNN architectures is an important task.

The introduction of competition connections (having negative weights) allows to achieve a better distinction of features learned by the neurons. The most common way to introduce competition between SNN neurons is to use a group of winner-take-all methods, where the first neuron in the layer that generates an impulse either zeroes the membrane variables of other neurons or prohibits other neurons from generating spikes for a given time. The use of competition connections in the form of negative synaptic weights has not been studied either, and the influence of the possibility of updating them according to local rules, along with stimulating projections, has been hardly studied too. In this work, it is shown that

learning competition relationships in the architecture of a locally connected SNN potentially allows to achieve better results in the accuracy of pattern recognition (on MNIST digits) due to the organization of competitive and at the same time cooperative work of groups of neurons.

Introduction

Modern formal neural networks do an excellent job at solving many machine learning problems [1; 2]. 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 [3]. 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 [2; 4] and even hundreds of billions (the latest OpenAI language model GPT-3 [5]).

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 [6]. 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 [7—9], 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.

!!!!!!!!!!!!!!!!!!! TODO: find references below

- SNNs can be implemented on specialized neuromorphic processing units (both based on digital elements [10—12] and using hybrid digital-analog circuits (BrainScales, [13]), including those based on memristors [links to others (Ielmini, Querlioz) and our works]), 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. [14; 15].

- 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 [2] architectures. Moreover, the introduction of additional recurrent competition connections contributes to a better separation of features in the learning process [16; 17]. 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 competition links, and as basic models, convolutional and fully connected networks with competition connections:

i we study the effect of training the competition connections [16—18] between neurons on the accuracy of the image classification problem of handwritten digits from the MNIST [19] dataset for the architecture of a locally connected network (Locally Connected Spiking Neural Network, LCSNN) [20];

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

1 Training of the SNN with fixed competition connections of local receptive fields

1.1 Problem description

For the 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.

1.2 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) [21] — receptive field, or patch. The architecture of the locally connected layer is the same, but each neuron has its own unique weights [20].

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. We connect neurons with common receptive fields by competition connections.

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. Competition connections are introduced to improve the separation

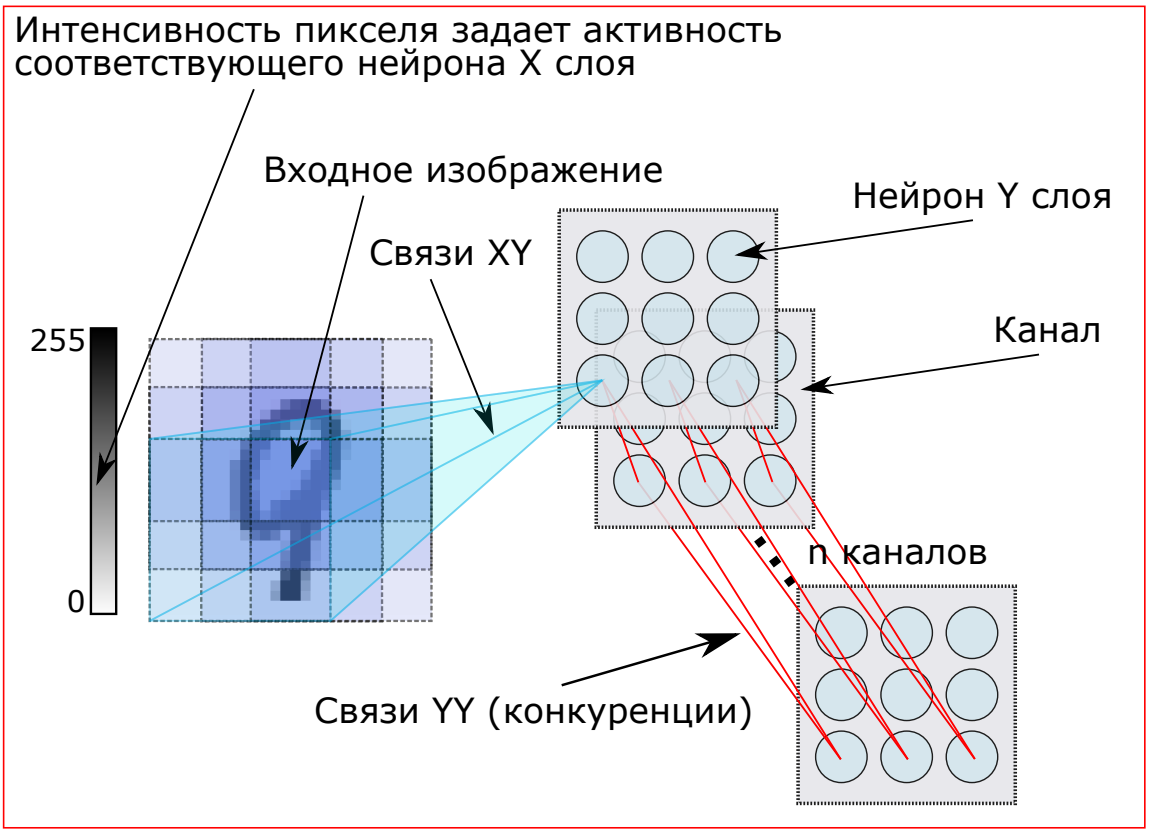


Рис. 1. LCSNN architecture

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 $\approx 100 - 900$ 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 $\times$20, the number of neurons per the channel was 3 or 4.

This work uses the Adaptive Integrate-And-Fire (ALIF) model of neurons. The dynamics of the neuron potential in this model is given by the equation

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

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, but 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.

1.3 Forward connections training

To reduce the number of model parameters, images are cropped to 20 $\times$ 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 [7] 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

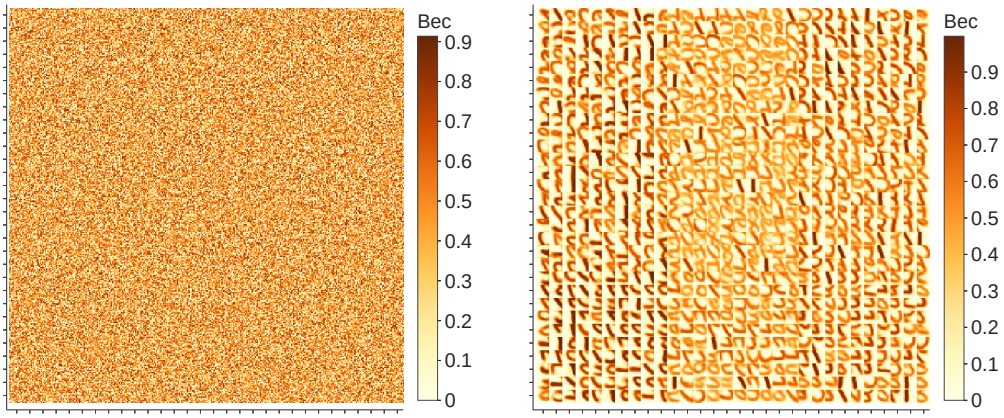
$$\Delta w = \begin{cases} A_+ \cdot e^{-\frac{t_{pre}-t_{post}}{\tau_+}}, t_{pre} - t_{post} > 0 \\ A_- \cdot e^{-\frac{t_{pre}-t_{post}}{\tau_-}}, t_{pre} - t_{post} < 0 \end{cases} \tag{4}$$

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. 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 [22] rule. In this paper it is used to train competition conenctions.

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) Before training (b) After 1000 training iterations

Рис. 2. Visualization of XY connection weights of a 900 Y neurons (100 neurons for each of the 9 patches) network — a 360 $\times$ 360 matrix. The weights of one neuron correspond to a 12 $\times$ 12 square. The weights of the Y neurons are grouped by patch - the lower left 120 $\times$ 120 square (10 $\times$ 10 neurons) corresponds to the lower left patch, the center square corresponds to the center patch, and so on. It can be seen that after training, the neurons learn features where digits elements can be observed.

1.4 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) —

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

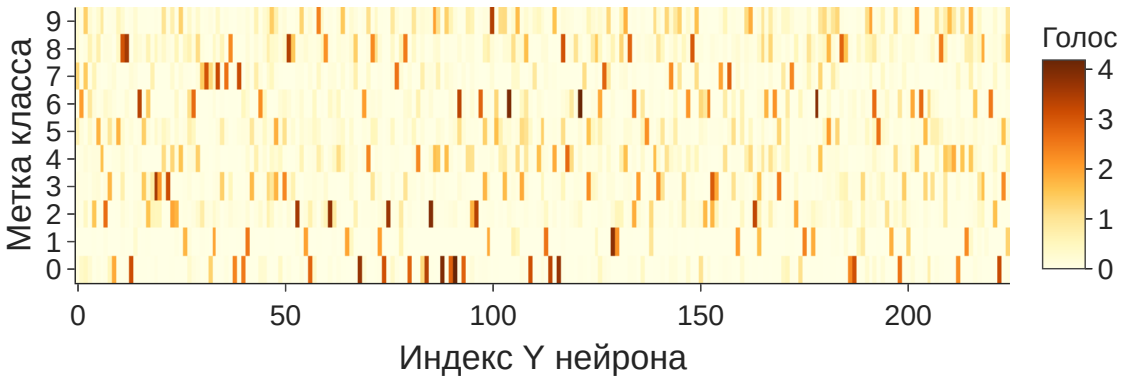
where s_t takes the value 0 or 1 depending on the presence of a spike at a given time, n_i is the size of the calibration sample for the digit i (we used $n_i = 1000$).

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.

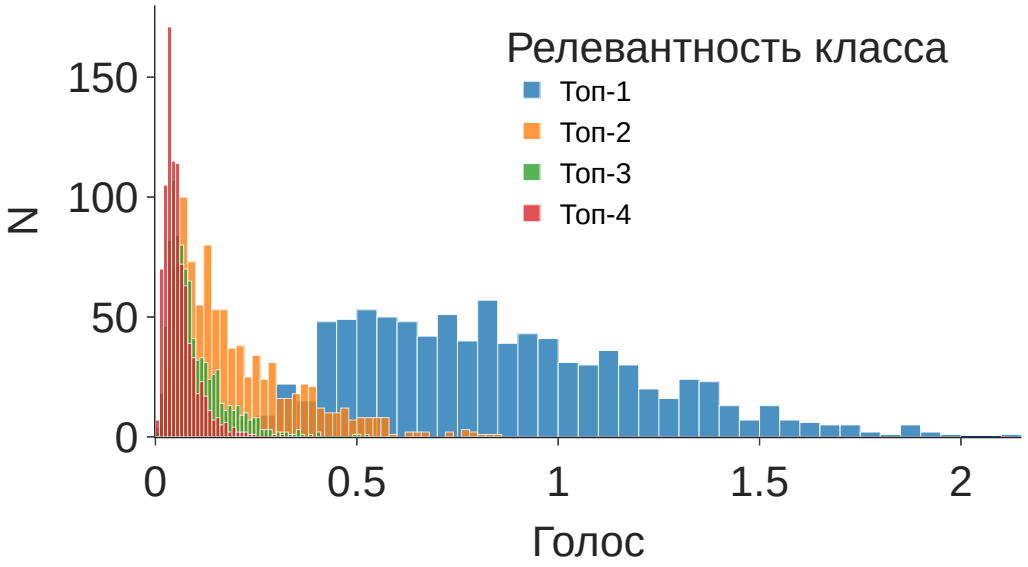
Further, the product of the number of neuron spikes by its vote will be called *score*.

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.



(a) Y neurons votes.



(b) Votes distrivution for the 4 most relevant classes (for each neuron). It can be seen that the vote value (average number of spikes per class) drops significantly with relevance decreasing.

Рис. 3. Visualization of the votes of the Y network layer with 225 Y neurons. High values correspond to a large specialization of the neuron in the digits class. It can be seen that almost all neurons have a nonzero vote value, therefore, almost all neurons in the network are active.

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

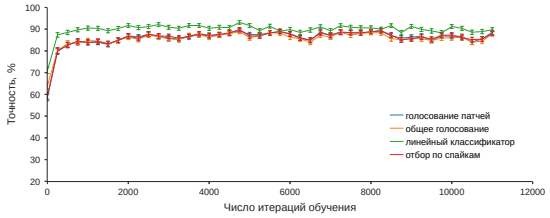
When using a **linear classifier**, the sums of spikes of individual Y layer neurons are fed as features to te classifier, which is trained to predict the true image labels. As in other methods of interpreting the network activity, classifier training is performed on a separate calibration sample.

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.

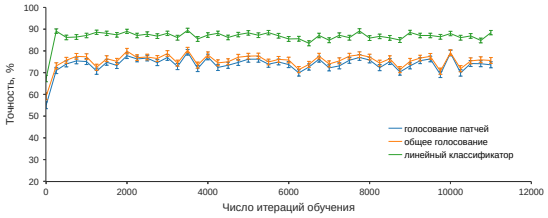
Accuracy learning curves were built for various interpretation algorithms. The accuracy was measured every 250 training iterations. To calibrate the interpretation algorithm at each step a calibration sample of 5000 was used, while to measure the accuracy, a test sample of 1000 was used.

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

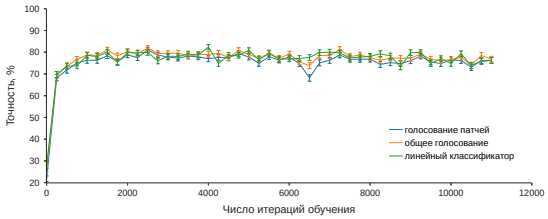
The linear classifier surpasses the LCSNN- and CSNN-based voting algorithms in accuracy, since it is a generalization of voting in the sense that it also uses the sum of the products of weights/votes by neurons activity. However, when training a linear classifier effective optimization algorithms (including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



(a) LCSNN learning curve.



(b) CSNN learning curve.



(c) FCSNN learning curve.

Рис. 4. Comparison of learning curves of different architectures of SNNs with competition: (4a) LCSNN, (4b) CSNN, (4c) 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 for LCSNN is 12, and for CSNN it is 8.

achieve practically the same goal (calculating satisfactory votes/weights) with a heuristic - the averaged activity of neurons by classes.

1.5 Сравнение эффективности операции свертки и локального рецептивного поля

Были проведены эксперименты по измерению точности сетей с различными архитектурами. Для сверточных и полносвязных сетей вводились аналогичные LCSNN связи конкуренции. Из-за высоких вычислительных нагрузок не ставилось задачи по нахождению параметров, обеспечивающих максимальную точность для каждой архи-

Таблица 1. Результаты сравнения различных архитектур спайковых нейронных сетей. Для каждой конфигурации точность измерялась $N = 5$ раз. В таблице указаны среднее арифметическое значение точности и его стандартное отклонение. «Ядро» соответсвует числу k в размере свертки $k \times k$.

	Конфигурация					Точность, %	
N	Архитектура	Каналы	Ядро	Параметры	Y нейроны	Метод [1]	Метод [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

тектуры. Эти параметры были подобраны приблизительно, однако, судя по результатам отдельных отладочных экспериментов, величина расхождения с оптимальными значениями не превышает 1-2%. Заметим, что при использовании линейного классификатора в качестве алгоритма интерпретации удалось достигнуть 95% точности для локально соединенной сети из 1000 каналов с размером ядра 12.

Видно (№2 и №7 в таблице 1), что локально соединенная сеть превосходит сверточную сеть по точности даже при чуть превышающем

[1] Лучший алгоритм голосования

[2] Линейный классификатор

[3] Сеть с обучением связей конкуренции

числе параметров последней. Также, можно заметить, что не следует использовать слишком малый размер ядра свертки. Это приводит к выучиванию менее существенных признаков. Действительно, в традиционном машинном обучении на основе ИНН используются либо неглубокие сети с большими ядрами свертки, либо глубокие модели, но, наоборот, с маленькими ядрами свертки.

Следует отметить, что SNN могут достигать значительно больших точностей распознавания на MNIST. Для этого можно использовать: (i) сети с существенно большим числом весовых параметров (например, с увеличенным числом каналов); (ii) более глубокие сети (с большим количеством слоев); (iii) более эффективные на сегодня алгоритмы обучения (например, обучение с учителем или с использованием контрастивной функции потерь [23]). В то же время, достижение максимально возможной точности не являлось целью настоящего исследования, адресованного, в первую очередь, сравнению различных архитектур SNN.

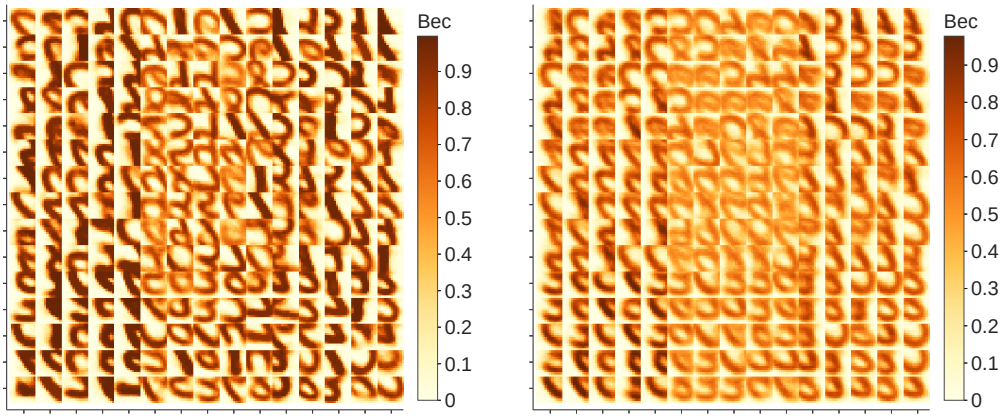
Результат, полученный в настоящей работе, практически соответствует результату из [20]. Заметим, что при помощи признаков, обученных без учителя достигаются результаты, лишь немногим уступающие результатам, полученным при помощи обучения с учителем. При этом лучшая локально соединенная сеть с конкуренцией локальных рецептивных полей из этой работы имеет 10^7 связей (из них связей прямого распространения — $1 \cdot 10^6$, связей конкуренции — $9 \cdot 10^6$), по сравнению с $4.6 \cdot 10^7$ (из них связей прямого распространения — $0.5 \cdot 10^7$, связей конкуренции — $4.1 \cdot 10^7$) у полносвязной сети с конкуренцией из [24]. К тому же, обучение сети из [24] велось в течение $1 \cdot 10^6$ итераций, тогда как в этой работе число итераций обучения не превосходит 5000.

2 Обучение связей конкуренции

Ингибирующие связи YY существенно влияют на обучение связей прямого распространения сигнала XY. Большие по модулю значения весов конкуренции способствуют большей вариативности и спе-

Таблица 2. Результаты других исследований спайковых нейронных сетей. Во всех используется датасет MNIST.

Статья	Архитектура	Обучение	Точность, %
Эта работа	Локальная + конкуренция	Без учителя	95.1 ± 0.5
[20]	Локальная + конкуренция	Без учителя	95.07 ± 0.63
[24]	Полносвязная + конкуренция	Без учителя	95
[16]	Полносвязная + конкуренция	С учителем / с частичным привлечением учителя	95.4 / 72.1
[25]	Сверточная	С частичным привлечением учителя	96.95 ± 0.08
[26]	Сверточная	С частичным привлечением учителя	99.28 ± 0.10
[27]	Сверточная	С частичным привлечением учителя	97.20 ± 0.07



(a) Высоко специализированные веса, вес конкуренции равен –100.

(b) Слабо специализированные веса, вес конкуренции равен –10.

Рис. 5. Влияние конкуренции на обучение связей XY.

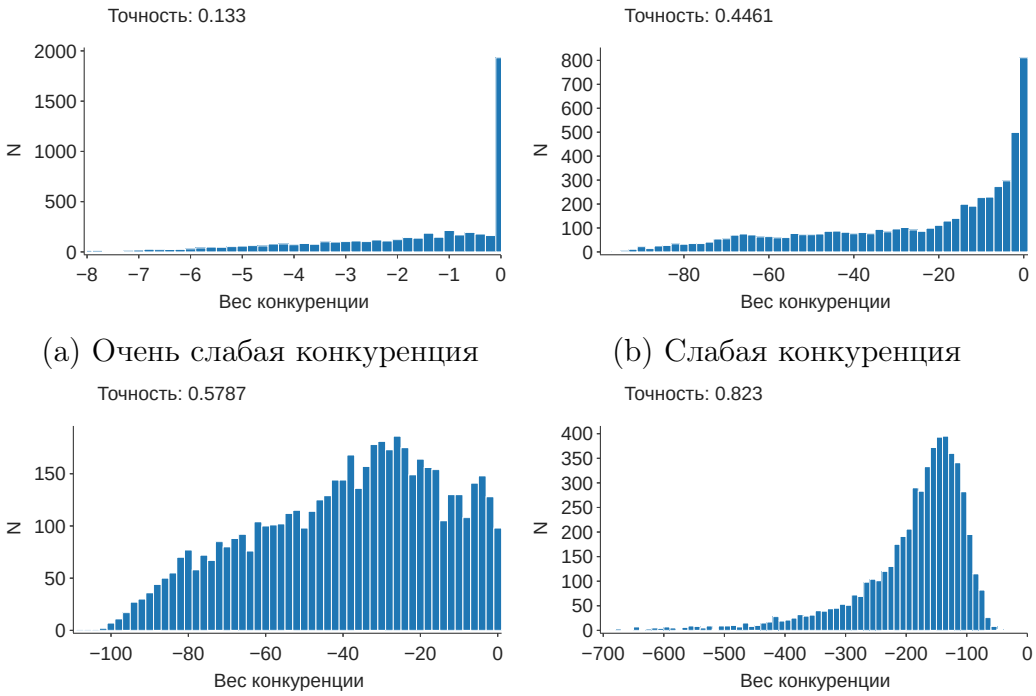
циализации в обучении Y нейронов, так как для каждого рецептивного поля одновременно активными не могут быть нейроны, имеющие схожие веса XY (Рис. 5a). Наоборот, малые по модулю веса конкуренции не позволяют нейронам эффективно специализироваться (Рис. 5a).

Все SNN, о которых шла речь в настоящей работе до этого момента, имели фиксированные веса конкуренции. В связи с этим, возникает вопрос, как влияет на точность сети обучение весов конкуренции. Для ответа на поставленный вопрос с целью обновления связей YY было выбрано правило anti-STDP (правило, противоположное по знакам A_+ и A_- в стандартном STDP). При варьировании значений параметров этого правила были получены различные распределения весов конкуренции. Начальные значения весов конкуренции задавались из равномерного распределения от 0 до некоторого числа, которое в дальнешем будем называть начальным весом конкуренции. Эксперименты проводились с LCSNN.

Видно, что точность сети повышается при смещении распределения весов конкуренции в сторону больших по модулю отрицательных значений. Заметим опять, что целью являлось не нахождение пара-

Таблица 3. Параметры anti-STDP

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



(a) Очень слабая конкуренция

(b) Слабая конкуренция

(c) Средняя конкуренция

(d) Сильная конкуренция

Рис. 6. Различные распределения весов конкуренции

метров сети, обеспечивающих максимальную точность, а исследование влияния способности обучения конкуренции на точность сети с заданной конфигурацией остальных параметров.

Примечательно, что не все связи YY получают большие по модулю значения (Рис. 6d). Это объясняется тем, что нейроны, специализирующиеся на существенно разных признаках, не нуждаются в конкуренции, так как они не проявляют высокую активность одновременно.

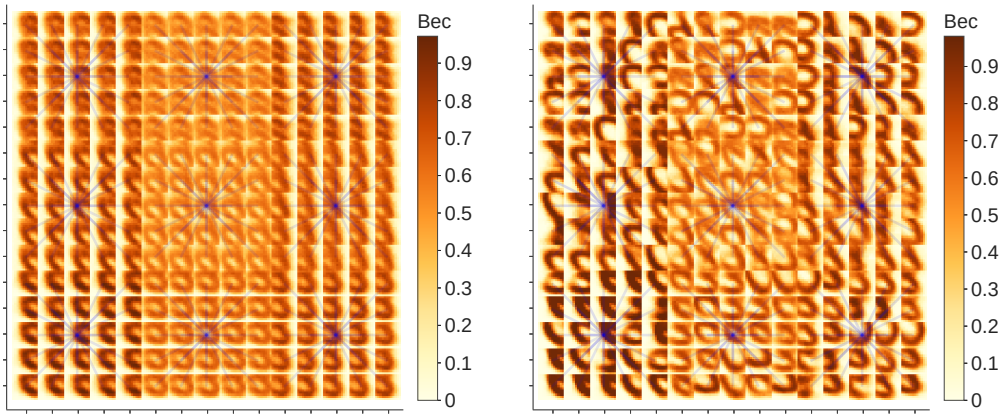
Обучение конкуренции производилось только на сетях из 25 каналов (225 нейронов), так как его моделирование требует больших вычислительных ресурсов.

В ходе оптимизации моделей была измерена точность большого количества LCSNN с различными конфигурациями гиперпараметров (Рис. 8). Видно, что конфигурации с высокими точностями (темные ломаные) локализуются определенным образом - дописать

2.1 Анализ обученных весов конкуренции

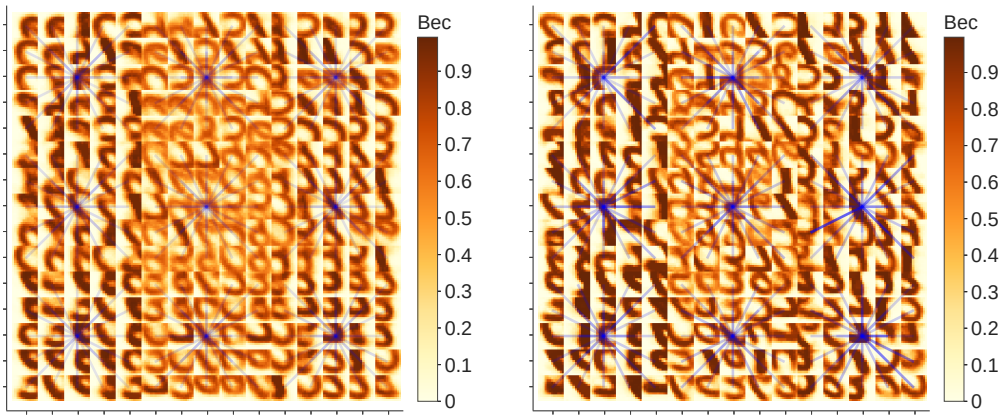
Дополнительно были проведены эксперименты по ограничению значений весов конкуренции по модулю сверху и снизу до проведения обучения сети. Оказалось, что веса конкуренции во всем диапазоне их изменения играют важную роль в работе LCSNN, поскольку ограничение как сверху, так и снизу негативно влияет на точность распознавания (Рис. 9). Это объясняется тем, что высокая конкуренция способствует большей специализации нейронов и потому полезна, а низкая конкуренция позволяет нейронам кооперироваться и распознавать классы совместно (в том числе обеспечивая накопление большей статистики для калибровки голосов нейронов).

Обучение конкуренции позволило достичь точности, незначительно (на 2%) превышающей точность сети такой же конфигурации, но с фиксированной конкуренцией с ингибирующими весами, равными -50 (Таб. 1, №4 и №5).



(a) Очень слабая конкуренция

(b) Слабая конкуренция



(c) Средняя конкуренция

(d) Сильная конкуренция

Рис. 7. Визуализация весов конкуренции поверх весов XY. Изображения соответствуют весам сетей с рисунка 6. Изображены только веса конкуренции для одного нейрона в каждом рецептивном поле для избегания загромождения визуализации. Насыщенный синий цвет соответствует большим по модулю весам конкуренции (используется среднее арифметическое между весами W_{ij} и W_{ji}). Видно, что похожие признаки сильнее конкурируют между собой, чем различные.

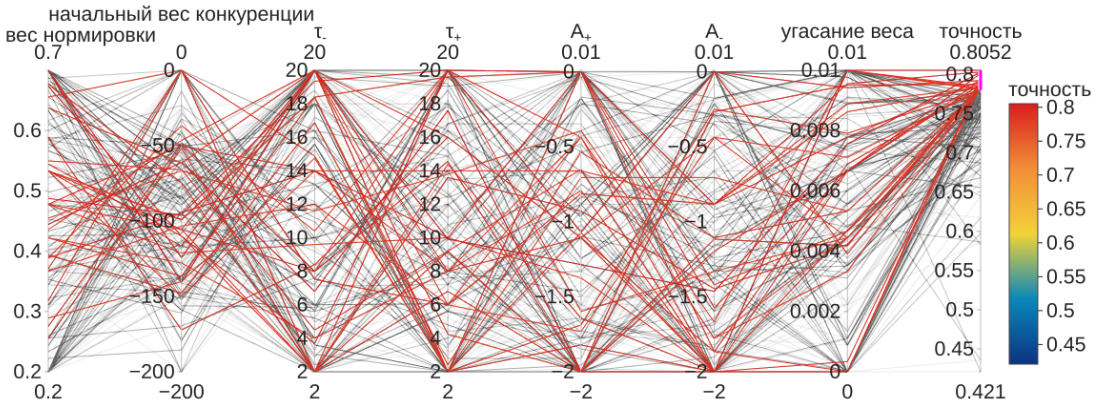
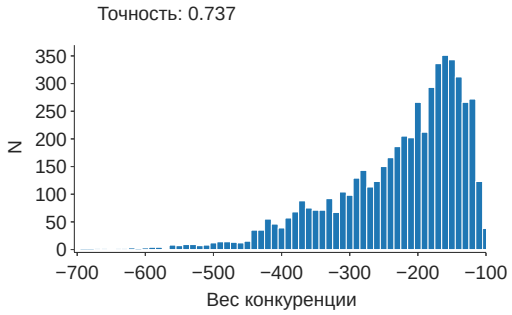
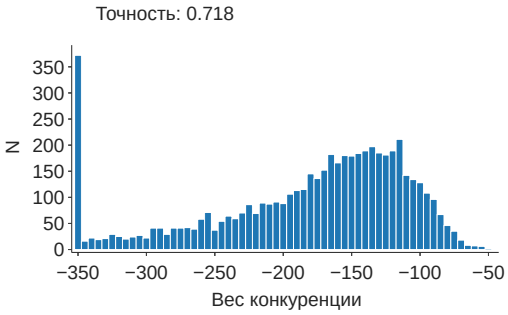


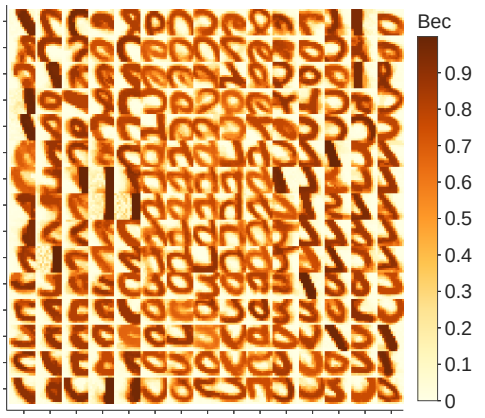
Рис. 8. Визуализация достигаемой точности распознавания LCSNN при использовании различных комбинаций параметров в пространстве гиперпараметров. Параметры «вес нормировки», «начальный вес конкренции» и «угасание веса» отвечают за значения весов, а следующие два являются комбинациями параметров anti-STDP. На последней оси указана точность сети с конфигурацией, соответствующих точкам пересечения ломаной осей гиперпараметров.



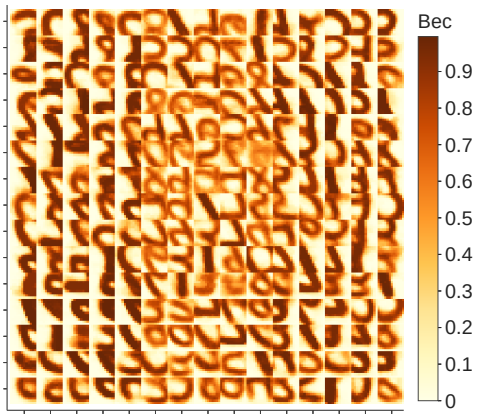
(a) Распределение весов конкуренции, ограничение сверху



(b) Распределение весов конкуренции, ограничение снизу



(c) Веса прямого распространения, ограничение сверху



(d) Веса прямого распространения, ограничение снизу

Рис. 9. Влияние ограничения значений весов конкуренции на точность. Веса были ограничены значениями -350 снизу и -100 сверху. Точность сети с аналогичными параметрами, но без ограничения конкуренции составляет 0.823, ее веса XY изображены на Рис. 7d, а распределение ее весов конкуренции — на Рис. 6d. Видно, что в обоих случаях обучаются менее четкие веса прямого распространения.

3 Обсуждение

Настоящее исследование демонстрирует, что локально соединенная сеть – перспективная спайковая нейросетевая архитектура, подходящая для эффективной реализации на специализированном нейрочипе. Действительно, способность быстро выходить на плато кривой обучения позволяет использовать LCSNN для обучения на выборках сравнительно небольшого объема (3000-5000). Использование дополнительного алгоритма интерпретации активности, такого как линейный классификатор, позволяет повысить эффективность сети. Заметим, что при этом сами признаки распознаваемых объектов выучиваются сетью без учителя. Остается открытым вопрос о возможности получения схожих результатов без использования обучения с учителем для интерпретации активности сети.

Главным преимуществом LCSNN является, как было показано, то, что локально соединенная сеть превосходит сверточную в точности при примерно одинаковом или чуть большем числе параметров. Предположительно, это происходит за счет более богатой статистики патчей (рецептивных полей), на которые реагируют нейроны внутри каждого канала, по сравнению с одним рецептивным полем на каждый канал в сверточной архитектуре. Ценой за это является увеличенное число параметров в LCSNN, но не является критическим фактором, ввиду локальной (в противоположность полносвязной) топологии межнейронных контактов. Локальная архитектура также позволяет сохранить информацию о локализации выделяемого признака в пространстве и потому является разумным выбором для реализации SNN, обучающихся преимущественно без учителя. Данный выбор обусловливает также его совокупную эффективность (по производимости, точности и энергопотреблению при определённой плотности элементов) при аппаратной реализации, при которой каждый вес обычно представлен одним или несколькими физическими элементами (главным образом, ячейками SRAM [10; 11] или мемристорами [28]). Обучение связей конкуренции слегка увеличивает точность работы алгоритма на базе LCSNN, но не настолько критично, чтобы оправдывать использование такой дорогой опера-

ции как введение и обучение значительного числа ингибирующих связей с различными по модулю конечными весами. В то же время, вероятно, этот вывод справедлив только для небольших сетей, вроде исследованной в настоящей работе. На наш взгляд, более тщательного изучения заслуживает подбор оптимальных алгоритмов и параметров обучения конкуренции для глубоких SNN, в которых баланс конкуренции и кооперации нейронов внутри каждого слоя может приводить к формированию кластеров из классов подаваемых изображений, что способствует выстраиванию семантически-подобной иерархии выученных признаков и их комбинаций, то есть конечных образов [29]. Потенциально такая иерархия образов может привести к существенному приросту качественных показателей работы интеллектуальных алгоритмов.

Выводы

Было показано, что для задачи распознавания образов на датасете MNIST локально соединенная архитектура с конкуренцией превосходит сверточную при равном числе параметров. Также было показано, что обучение весов конкуренции позволяет незначительно повысить качество модели, а в полученном распределении весов конкуренции важны как высокие по модулю (сильная конкуренция), так и низкие (кооперация) значения весов.

Литература

1. Regularization of Neural Networks using DropConnect / L. Wan [и др.]. — 2013. — Июнь. — URL: http://proceedings.mlr.press/v28/wan13.html.
2. A survey of the recent architectures of deep convolutional neural networks / A. Khan [и др.] // Artificial Intelligence Review. — 2020. — Апр. — ISSN 1573-7462. — DOI: 10.1007/s10462-020-09825-6. — URL: http://dx.doi.org/10.1007/s10462-020-09825-6.
3. *Edwards C.* Growing pains for deep learning // Commun. ACM. — 2015. — Т. 58. — С. 14—16.
4. A State-of-the-Art Survey on Deep Learning Theory and Architectures / M. Z. Alom [и др.] // Electronics. — 2019. — Март. — Т. 8. — С. 292. — DOI: 10.3390/electronics8030292.
5. Language Models are Few-Shot Learners / T. B. Brown [и др.]. — 2020. — arXiv: 2005.14165 [cs.CL].
6. Deep learning for time series classification: a review / H. Ismail Fawaz [и др.] // Data Mining and Knowledge Discovery. — 2019. — Март. — Т. 33, № 4. — С. 917—963. — ISSN 1573-756X. — DOI: 10.1007/s10618-019-00619-1. — URL: http://dx.doi.org/10.1007/s10618-019-00619-1.

7. *Markram H.*, *Gerstner W.*, *Sjöström P. J.* A History of Spike-Timing-Dependent Plasticity // Frontiers in Synaptic Neuroscience. — 2011. — Т. 3. — С. 4. — ISSN 1663-3563. — DOI: 10.3389/fnsyn.2011.00004. — URL: https://www.frontiersin.org/article/10.3389/fnsyn.2011.00004.

8. *Pehlevan C.* A Spiking Neural Network with Local Learning Rules Derived From Nonnegative Similarity Matching. — 2019. — arXiv: 1902.01429 [cs.NE].

9. *Baldi P.*, *Sadowski P.* A theory of local learning, the learning channel, and the optimality of backpropagation // Neural Networks. — 2016. — Нояб. — Т. 83. — С. 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.

10. A million spiking-neuron integrated circuit with a scalable communication network and interface / P. A. Merolla [и др.] // Science. — 2014. — Т. 345, № 6197. — С. 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.

11. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning / M. Davies [и др.] // IEEE Micro. — 2018. — Янв. — Т. PP. — С. 1—1. — DOI: 10.1109/MM.2018.112130359.

12. Akida Neural Processor System-on-Chip. — 2020. — Апр. — URL: https://brainchipinc.com/akida-neuromorphic-system-on-chip/.

13. SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation / E. Painkras [и др.] // Solid-State Circuits, IEEE Journal of. — 2013. — Авг. — Т. 48. — С. 1943—1953. — DOI: 10.1109/JSSC.2013.2259038.

14. Darwin: A neuromorphic hardware co-processor based on spiking neural networks / D. Ma [и др.] // Journal of Systems Architecture. — 2017. — Т. 77. — С. 43—51. — ISSN 1383-7621. — DOI: 10.1016/

j.sysarc.2017.01.003. — URL: http://www.sciencedirect.com/science/article/pii/S1383762117300231.

15. *Tang G.*, *Shah A.*, *Michmizos K. P.* Spiking Neural Network on Neuromorphic Hardware for Energy-Efficient Unidimensional SLAM. — 2019. — arXiv: 1903.02504 [cs.RO].

16. *Demin V.*, *Nekhaev D.* Recurrent Spiking Neural Network Learning Based on a Competitive Maximization of Neuronal Activity // Frontiers in Neuroinformatics. — 2018. — Нояб. — Т. 12. — С. 79. — DOI: 10.3389/fninf.2018.00079.

17. *Nekhaev D.*, *Demin V.* Competitive Maximization of Neuronal Activity in Convolutional Recurrent Spiking Neural Networks. — 2020. — Янв. — DOI: 10.1007/978-3-030-30425-6_30.

18. Spiking Neural Networks Hardware Implementations and Challenges / M. Bouvier [и др.] // ACM Journal on Emerging Technologies in Computing Systems. — 2019. — Июнь. — Т. 15, № 2. — С. 1—35. — ISSN 1550-4840. — DOI: 10.1145/3304103. — URL: http://dx.doi.org/10.1145/3304103.

19. Gradient-based learning applied to document recognition / Y. Lecun [и др.] // Proceedings of the IEEE. — 1998. — Т. 86, № 11. — С. 2278—2324.

20. Locally Connected Spiking Neural Networks for Unsupervised Feature Learning / D. J. Saunders [и др.]. — 2019. — arXiv: 1904.06269 [cs.NE].

22. *Roberts P.*, *Leen T.* Anti-Hebbian Spike-Timing-Dependent Plasticity and Adaptive Sensory Processing // Frontiers in computational neuroscience. — 2010. — Дек. — Т. 4. — С. 156. — DOI: 10.3389/fncom.2010.00156.

23. *Hadsell R.*, *Chopra S.*, *LeCun Y.* Dimensionality Reduction by Learning an Invariant Mapping. — 2006. — DOI: 10.1109/CVPR.2006.100.

24. *Diehl P.*, *Cook M.* Unsupervised learning of digit recognition using spike-timing-dependent plasticity // Frontiers in Computational Neuroscience. — 2015. — Т. 9. — С. 99. — ISSN 1662-5188. — DOI: 10.3389/fncom.2015.00099. — URL: https://www.frontiersin.org/article/10.3389/fncom.2015.00099.

25. *Tavanaei A.*, *Maida A. S.* Multi-layer unsupervised learning in a spiking convolutional neural network. — 2017.

26. Training Deep Spiking Convolutional Neural Networks With STDP-Based Unsupervised Pre-training Followed by Supervised Fine-Tuning / C. Lee [и др.] // Frontiers in Neuroscience. — 2018. — Т. 12. — С. 435. — ISSN 1662-453X. — DOI: 10.3389/fnins.2018.00435. — URL: https://www.frontiersin.org/article/10.3389/fnins.2018.00435.

27. *Tavanaei A.*, *Kirby Z.*, *Maida A. S.* Training Spiking ConvNets by STDP and Gradient Descent // 2018 International Joint Conference on Neural Networks (IJCNN). — 2018. — С. 1—8.

28. Review of memristor devices in neuromorphic computing: materials sciences and device challenges / Y. Li [и др.] // Journal of Physics D: Applied Physics. — 2018. — Сент. — Т. 51, № 50. — С. 503002. — DOI: 10.1088/1361-6463/aade3f. — URL: https://doi.org/10.1088%2F1361-6463%2Faade3f.

29. *Nekhaev D.*, *Demin V.* Competitive Maximization of Neuronal Activity in Convolutional Recurrent Spiking Neural Networks. — 2020.

Все материалы этой работы находятся в репозитории
https://github.com/danielgafni/bachelor