# Neural Networks for Recognition of Semiographic Chants

Andrey Philippovich[1] and Boynov Maxim[2]

[1] University of mechanical engineering, Moscow,Russia,
aphilippovich@it-claim.ru,
[2] Bauman Moscow State Technical University, Moscow, Russia,
mboynov@it-claim.ru

**Abstract.** The paper presents research on the problem of recognition of signs applied to the analysis of basic units of ancient Russian chants. For testing, we take two types of deep neural networks: Back Propagation Neural Network (BPNN) and Convolution Neural Network (CNN). We investigate main features of the chant units and the properties of the networks to choose the best structure and algorithm. The results provide an analysis of accuracy for both approaches used in solving this particular task.

**Keywords:**Image Recognition, Neural Networks, Deep Learning, Semiography, BPNN, CNN, Ancient Russian Musical Manuscripts, Znamenny Notation.

## 1 Introduction

One of the most important and prominent tasks in the study of ancient Russian culture is the exploration of melodic content in vocal music manuscripts from the XII-XVII centuries. The melodies in these books are written using special musical structures that evolved in Russia over the centuries. The concept «semiography» is understood to mean the conventionally accepted methods of musical writing and expression of certain musical sounds and how they are related. Fig. 1 shows a fragment of a semiographic chant.

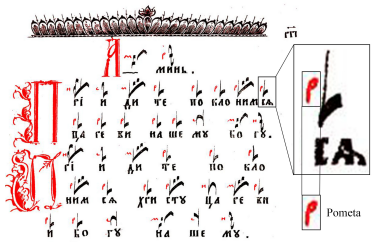


Fig. 1. The special type of marks “Pometa”

Music chants written by Znamenny Notation contain rows of semiographic signs (znamyas, flags) and rows of words divided by syllables. Every semiographic sign has special type of marks "Pometa". These marks indicate the duration and amplitude of the music [1]. The paper presents research on the problem of recognition of these marks with artificial deep neural networks. The novelty of the article is to present processing algorithms and train networks on obtained images after processing algorithms.

## 2 Materials and methods

### 2.1 Dataset

There are seven most common types of pometas (see Table 1). Dataset is generated by our program and includes 488 images. It was randomly divided into 2 separate sets: training and test. The training set had 70 percents of the examples and the remaining 30 persents formed test set. Program automatically cuts pometas from full image. Fig. 2 shows how program handles fragment of image. At first the program cuts the characters using an algorithm developed by us. The essence of the algorithm is to extract the image area and calculate standard deviation[3]in this area. If the value is greater than the value specified by us on the next layer of all the pixel values fall, if not then the next layer gets nothing. After this treatment, there is a noise that is removed by another algorithm is based on identifying the character contour. If the character contour size more than a certain value, pixels inside the contour fall to the next layer.

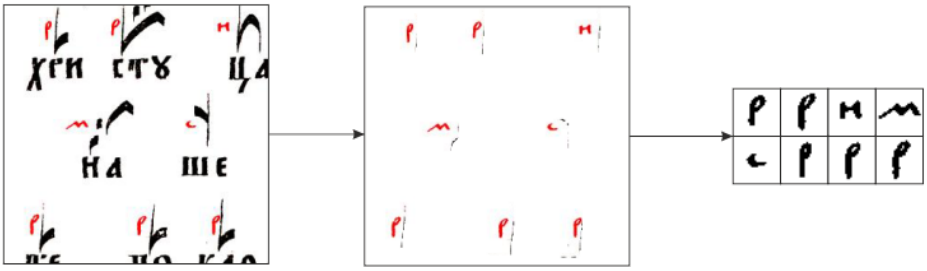

Fig. 2. A semiographic chant

The table below presents the number of examples of each type in the training and test sets.

For our neural networks, we create program to convert images from JPG format to "MNIST"[4] one. MNIST is a simple computer vision dataset on which are

[3] http://www.odelama.com/data-analysis/How-to-Compute-RGB-Image-Standard-Deviation-from-Channels-Statistics/

[4] http://yann.lecun.com/exdb/mnist/

Table 1. The list of pometas

| Pometa | С | Р | Н | М | П | Г | В |
|---|---|---|---|---|---|---|---|
| Training set | 49 | 104 | 59 | 84 | 49 | 11 | 22 |
| Test set | 14 | 31 | 17 | 25 | 14 | 3 | 6 |

based the recognition problem. The current best error rate on the MNIST digit-numberrecognition task (<0.3) approaches human performance [6]. The data is stored in a very simple file format designed for storing vectors and multidimensional matrices. All the integers in the files are stored in the "MSB first (high endian) format"[5] used by most non-Intel processors. Users of Intel processors and other low-endian machines must flip the bytes of the header. There are four files: $trainingsetimages, trainingsetlabels, testsetimages, testsetlabels$. Table 2 and Table 3 shows structure of these files.

Table 2. Training set label file / test set label file

| Offset | Type | Value | Description |
|---|---|---|---|
| 0000 | 32 bit integer | 0x00000801(2049) | magic number(MSB first) |
| 0004 | 32 bit integer | 488/110 | number of items |
| 0008 | unsigned byte | 0...6 | label |
| 0009 | unsigned byte | 0...6 | label |
| ... | ... | ... | ... |

Table 3. Training set image file / test set image file

| Offset | Type | Value | Description |
|---|---|---|---|
| 0000 | 32 bit integer | 0x00000803(2051) | magic number(MSB first) |
| 0004 | 32 bit integer | 488/110 | number of images |
| 0008 | 32 bit integer | 28 | number of rows |
| 0012 | 32 bit integer | 28 | number of columns |
| 0016 | unsigned byte | 0...255 | pixel |
| 0017 | unsigned byte | 0...255 | pixel |
| ... | ... | ... | ... |

Pixels are organized row-wise. For our experiments we have used pixels with values from 0 to 255. 0 means background (white), 255 means foreground (black).

[5] `https://en.wikipedia.org/wiki/Most_significant_bit`

### 2.2 Hardware and Software Configurations

In this article used the computer with 8GB Ram DDR3 on which the OS Microsoft Windows 8.1 and OS Ubuntu 15.04 is installed. It has a processor Intel Core i7-970 and videocard Radeon R280x. Also, for a machine learning used library known as TensorFlow[6]. Using TensorFlow makes it easy to implement backpropagation for convolutional neural networks, since it automatically computes all the mappings involved. It is also quite a bit faster than other libraries, and this makes it practical to train more complex networks. In particular, one great feature of TensorFlow is that it can run code on either a CPU or, if available, a GPU. Running on a GPU provides a substantial speedup and, again, helps make it practical to train more complex networks. alculations are performed without using the GPU and computing time is less than 10 minutes.

## 3 Back Propagation Neural Network (BPNN)

The structure of our BPNN[7] is described in fig. 3. Our net consists of three layers: an input layer, a hidden layer and an output layer. The input layer has 784 units, or neurons. Hidden layer has 625 and the output 7 units.

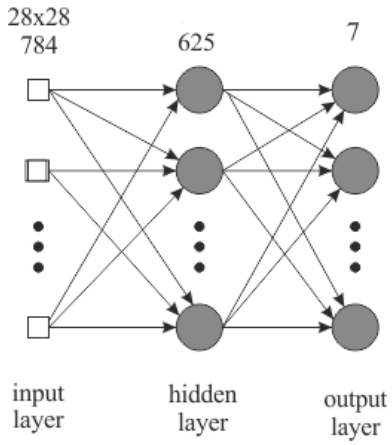


Fig. 3. BPNN structure

The standard way to model a neuron's output $f$ as a function of its input $x$ is with $f(x) = tanh(x)$. In terms of training time with gradient descent, these saturating nonlinearities are much slower than the non-saturating nonlinearity f(x) = max(0, x). Following Nair and Hinton [3], we refer to neurons with this nonlinearity as Rectified Linear Units (ReLUs). Deep convolutional neural networks

[6] `https://www.tensorflow.org`
[7] `https://en.wikipedia.org/wiki/Backpropagation`

with ReLUs train several times faster than their equivalents with tanh units. Two additional major benefits of ReLUs are sparsity and a reduced likelihood of vanishing gradient. One major benefit is the reduced likelihood of the gradient to vanish. This arises when a>0. In this mode, the gradient has a constant value. In contrast, the gradient of sigmoids becomes increasingly small as the absolute value of x increases. The constant gradient of ReLUs results in faster learning. The other benefit of ReLUs is sparsity. Sparsity arises when a<=0. The more such units that exist in a layer the more sparse the resulting representation. Sigmoids on the other hand are always likely to generate some non-zero value resulting in dense representations. Sparse representations seem to be more beneficial than dense representations. We are not the first to consider alternatives to traditional neuron models in CNNs. For example, Jarrett et al. [7] used the nonlinearity $f(x) = |tanh(x)|$. As the cost function, was chosen cross-entropy function. Learning rate of algorithm was 0.001. Convolutional Neural Networks are very similar to ordinary Neural Networks. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still express a single differentiable score function: From the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.[8] Structure for our network we can see in fig. 4. Input layer has size 28x28 pixels and value of each pixel can take value 0 or 1 after processing the initial image.

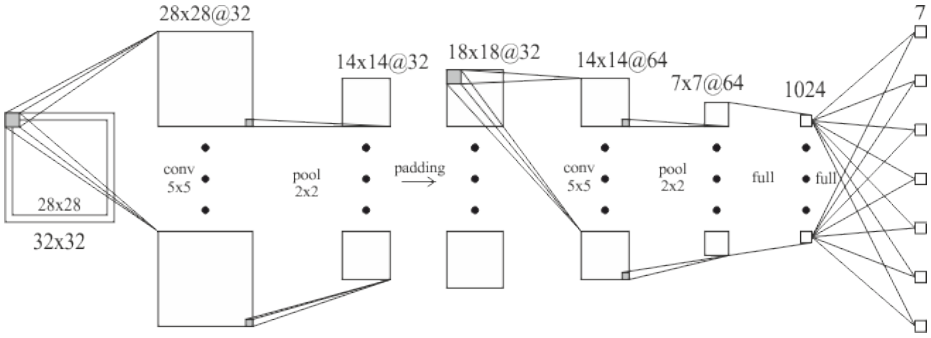


Fig. 4. Convolutional Neural Network structure

### 3.1 Parameters of convolution network

First parameter is the number of convolutional layers. The more convolutional layers the better (within reason, as each convolutional layer reduces the number

[8] http://cs231n.github.io/convolutional-networks/

of input features to the fully connected layers. After about two or three layers the accuracy gain becomes rather small so we need to decide what is more important: generalization accuracy or training time. All image recognition tasks are different so the best method is to simply try incrementing the number of convolutional layers one at a time until we are satisfied by the result.

Second parameter is the number of hidden layers. The number of required hidden layers depends on the intrinsic complexity of dataset, this can be understood by looking at what each layer achieves:

- Zero hidden layers allow the network to model only a linear function. This is unsuitable for most image recognition tasks.
- One hidden layer allows the network to model an arbitrarily complex function. This is suitable for nearly all image recognition tasks.
- Theoretically, two hidden layers offer little benefit over a single layer, however, in practice especially complex tasks may find an additional layer beneficial. This should be treated with caution, as a second layer can cause over-fitting. Using more than two hidden layers is almost never beneficial[9].

Third parameter is the number of nodes per hidden layer. There is no formula for deciding upon the number of nodes, it is different for each task. A rough guide to go by is to use a number of nodes 2/3 the size of the previous layer, with the first layer 2/3 the size of the final feature maps. This however is just a rough guide and depends again on the dataset. Another commonly used option is to start with an excessive number of nodes, then to remove the unnecessary nodes through pruning. For better recognize images input layer has padding.

Fourth parameter is the numbers of pooling layers. Pooling layers in CNNs summarize the outputs of neighboring groups of neurons in the same kernel map. Traditionally, the neighborhoods summarized by adjacent pooling units do not overlap (e.g.,[8, 7, 9]). To be more precise, a pooling layer can be thought of as consisting of a grid of pooling units spaced $s$ pixels apart, each summarizing a neighborhood of size $zz$ centered at the location of the pooling unit. If we set $s = z$, we obtain traditional local pooling as commonly employed in CNNs. If we set $s < z$, we obtain overlapping pooling.

### 3.2 Dropout

Combining the predictions of many different models is a very successful way to reduce test errors [10, 11], but it appears to be too expensive for big neural networks that already take several days to train. There is, however, a very efficient version of model combination that only costs about a factor of two during training. The recently-introduced technique, called "dropout" [13], consists of setting to zero the output of each hidden neuron with probability 0.5. The neurons which are "dropped out" in this way do not contribute to the forward pass

[9] `http://stackoverflow.com/questions/24509921/how-do-you-decide-the-parameters-of-a-convolutional-neural-network-for-image-cla`

and do not participate in backpropagation. So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights. This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons. It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons [12]. We don't use dropout because our training time is acceptable.

## 4 Results

Table 4 includes percentage of correct answers of test images of all types of networks. Both the network recognizes all images from the test sample correct. BPNN has 494375 weights to train and CNN has 6848, so in this case better to use CNN network. The best way to find a suitable network layout is literally to perform many trial and error tests. There is no one-size-fits-all network, and only you know the intrinsic complexity of your dataset. The most effective way of performing the number of necessary tests is through some form of cross validation [4]. Compare our result with the results of the article «Neural Models for Recognition of Basic Units of Semiographic Chants "[5].

Recall what result was obtained in this article. Initially, they manually cropped the pometas from chant manuscripts. In the next step removed noise and extra white space from the images. Then automatically extracted 10 geometrical features - number of intersections in the horizontal and vertical planes. Then they applied multidimensional scaling technique to project 10 dimensions on the 2D plot and added extra features to improve the clusterization rate. Finally, each pometa is described as a vector with 13 dimensions (5+5+3). For example, a vector representation of the pometa "M" might be $x_i = (1; 2; 2; 2; 1; 1; 1; 1; 1; 1; 1; 4; 0; 0)$.

Table 4. Results table

| MLP | PNN | CNN | BPNN |
|---|---|---|---|
| Multilayer Perceptron | Probabilistic Neural Network | Convolution Neural Network | Back Propagation Neural Network |
| 0.92 | 0.93 | 1 | 1 |

As we can see, CNN and BPNN have more answers that are correct but to obtain this result the network needs more time to train, because they have more parameters to train. Table 5 shows accuracy per each pometa.

So as we can see accuracy of pometas "H" and "П" only are different from one. This can be explained by the fact that they are very similar.

Table 5. Accuracy per each pometa

| | MLP | PNN | CNN | BPNN |
|---|---|---|---|---|
| С | 1 | 1 | 1 | 1 |
| Р | 1 | 1 | 1 | 1 |
| Н | 0.88 | 0.87 | 1 | 1 |
| М | 1 | 1 | 1 | 1 |
| П | 0.96 | 0.97 | 1 | 1 |
| Г | 1 | 1 | 1 | 1 |
| В | 1 | 1 | 1 | 1 |

## 5 Conclusion

During our research, we have developed two algorithms to processing image and compared four types of neural networks. We have achieved excellent accuracy image recognition. So our program itself can define pometas and translate them into a digital format. Further it is possible to improve and to determine not only the pometas but also others symbols. This program is easy to understand because there are not so many parameters to change.

# Bibliography


[1] *Philippovich, A.Yu., Danshina, M.V., Golubeva, I.V.* Editing and representation of ancient Russian semiographic chants on the web. In: Analysis of Images, Social Networks and Texts / Ed. by Dmitry I. Ignatov, Mikhail Yu. Khachay, Alexander Panchenko et al. — Springer International Publishing, 2014. — Vol. 436 of Communications in Computer and Information Science. — pp. 66-77.

[2] *Philippovich A.Yu, Golubeva I.V., Danshina M.V.* Semiotic system of musical texts In: Supplementary Proceedings of the 3rd International Conference on Analysis of Images, Social Networks and Texts (AIST'2014). — Vol. 1197. — Yekaterinburg, Russia : CEUR-WS.org, 2014. — pp. 28–34.

[3] *Vinod Nair and Geoffrey Hinton* Rectified linear units improve restricted Boltzmann machines. ICML.2010.

[4] *Devijver, Pierre A.; Kittler, Josef* Pattern Recognition: A Statistical Approach. London, GB: Prentice-Hall.1982

[5] *Vylomova, E.A., Philippovich, A.Yu., Danshina, M.V., Golubeva, I.V., Philippovich, Yu.N.* Neural models for recognition of basic units of semiographic chants.In: Analysis of Images, Social Networks and Texts / Ed. by Dmitry I. Ignatov, Mikhail Yu. Khachay, Alexander Panchenko et al. — Springer International Publishing, 2014. — Vol. 436 of Communications in Computer and Information Science. — pp. 249-254.

[6] *D. Cire¸san, U. Meier, and J. Schmidhuber.* Multi-column deep neural networks for image classification. Arxiv preprint arXiv:1202.2745, 2012.

[7] *K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun.* What is the best multi-stage architecture for object recognition? In International Conference on Computer Vision, pages 2146–2153. IEEE, 2009.

[8] *Y. LeCun, K. Kavukcuoglu, and C. Farabet.* Convolutional networks and applications in vision. In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, pages 253–256. IEEE, 2010.

[9] *D. Cire¸san, U. Meier, and J. Schmidhuber.* Multi-column deep neural networks for image classification. Arxiv preprint arXiv:1202.2745, 2012.

[10] *R.M. Bell and Y. Koren.* Lessons from the netflix prize challenge. ACM SIGKDD Explorations Newsletter, 9(2):75–79, 2007.

[11] *L. Breiman.* Random forests. Machine learning, 45(1):5–32, 2001.

[12] *A.Krizhevsky, I.Sutskever, Geoffrey E. Hinton* ImageNet Classification with Deep Convolutional Neural Networks

[13] *G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov* Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.