

A Shallow Convolutional Neural Network for Accurate Handwritten Digits Classification

Vladimir Golovko^{1(✉)}, Mikhno Egor¹, Aliaksandr Brich¹, and Anatoliy Sachenko²

¹ Brest State Technical University, Moskovskaja 267, 224017 Brest, Belarus
gva@bstu.by

² Research Institute for Intelligent Computer Systems,
Ternopil National Economic University, 3 Peremoga Square, Ternopil, 46020, Ukraine
as@tneu.edu.ua

Abstract. At present the deep neural network is the hottest topic in the domain of machine learning and can accomplish a deep hierarchical representation of the input data. Due to deep architecture the large convolutional neural networks can reach very small test error rates below 0.4% using the MNIST database. In this work we have shown, that high accuracy can be achieved using reduced shallow convolutional neural network without adding distortions for digits. The main contribution of this paper is to point out how using simplified convolutional neural network is to obtain test error rate 0.71% on the MNIST handwritten digit benchmark. It permits to reduce computational resources in order to model convolutional neural network.

Keywords: Convolutional neural networks · Handwritten digits · Data classification

1 Introduction

An artificial neural network is powerful tool in different domains [1–5]. Over the last decade the machine learning techniques has the leading role in domain of artificial intelligence [1]. This is confirmed by recent qualitative achievements in images, video, speech recognition, natural language processing, big data processing and visualization, etc. [1–18]. These achievements are primarily associated with new paradigm in machine learning, namely deep neural networks and deep learning [2, 6–18]. However in many real world applications the important problem is limited computational resources, which doesn't permit to use deep neural networks. Therefore the further development of shallow architecture is an important task. It should be noted especially that for many real applications, the shallow architecture can show the comparable accuracy in comparison with deep neural networks.

This paper deals with a convolutional neural network for handwritten digits classification. We propose a simplified architecture of convolutional neural networks, which permits to classify handwritten digits with more precision than a conventional convolution neural network LeNet-5. We have shown that by using a simplest convolutional neural network can be obtained the better classification results.

The rest of the paper is organized as follows. Section 2 introduces the standard convolutional neural networks. In Sect. 3 we propose a simplified convolutional network. Section 4 demonstrates the results of experiments and finally Sect. 5 gives conclusion.

2 Related Works

Convolutional neural network is a further development of a multilayer perceptron and neocognitron and is widely used for image processing [19–22]. This kind of neural network is invariant to shifts and distortions of the input. Convolutional neural network integrates three approaches, namely local receptive field, shared weights and spatial subsampling [20–22]. Using local receptive areas the neural units of the first convolutional layer can extract primitive features such as edges, corners etc. The general structure of convolutional neural network is shown in the Fig. 1.

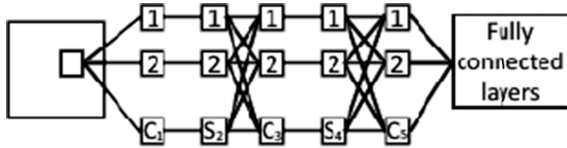


Fig. 1. General structure of convolutional neural network

A convolutional layer consists of set of a feature maps and the neural units of each map contain the same set of weights and thresholds. As a result each neuron in a feature map performs the same operations on different parts of image. The sliding windows technique is used for image scanning. Therefore if size of window is $p \times p$ (receptive field) then each unit in a convolutional layer is connected with p^2 units of the corresponding receptive field. Each receptive field in input space is mapped into special neuron in each feature map. Then if the stride of sliding window is one that the numbers of neurons in each feature map is given by

$$D(C_1) = (n - p + 1)(n - p + 1) \tag{1}$$

where $n \times n$ is size of image. If the stride of sliding window is S that the numbers of neurons in each feature map is defined by the following way:

$$D(C_1) = \left(\frac{n - p}{s} + 1\right) \left(\frac{n - p}{s} + 1\right) \tag{2}$$

Accordingly, the common number of synaptic weights in convolutional layer is defined by

$$V(C_1) = M(p^2 + 1) \tag{3}$$

where M – the number of feature maps in convolutional layer. Let's represent the pixels of the input image in one-dimensional space. Then the ij -th output unit for k -th feature map in convolutional layer is given by

$$y_{ij}^k = F(S_{ij}^k) \tag{4}$$

$$S_{ij}^k = \sum_c w_{cij}^k x_c - T_{ij}^k \tag{5}$$

where $c = 1, p^2$, F – the activation function, S_{ij}^k — the weighted sum of the ij -th unit in k -th feature map, w_{ij}^k — the weight from the c -th unit of the input layer to the ij -th unit of the k -th feature map, T_{ij}^k – the threshold of the ij -th unit of the k -th feature map.

As already said the neural units of each feature map contain the same set of weights and thresholds. As a result the multiple features can be extracted at the same location. These features are then combined by the higher layer using pooling in order to reduce the resolution of feature maps [22]. This layer is called subsampling or pooling layer and performs a local averaging or maximization different regions of image. To this end, each map of convolutional layer is divided into non-overlapping areas with size of $k \times k$ and each area is mapped into one unit of corresponding map in pooling layer. It should be noted that each map of convolutional layer is connected only with corresponding map in pooling layer. Each unit of pooling layer computes the average or maximum of k^2 neurons in a convolutional layer:

$$z_j = \frac{1}{k \times k} \sum_{j=1}^{k \times k} y_j$$

$$z_j = \max(y_j) \tag{6}$$

The number the of neurons in each pooling map is given by

$$D(S_2) = \frac{D(C_1)}{k^2} \tag{7}$$

The number of feature maps in pooling layer will be the same like in convolutional layer and equal M . Thus convolutional neural network represents combination of convolutional and pooling layers, which perform nonlinear hierarchical transformation of input data space. The last block of the convolutional neural network is a multilayer perceptron, SVM or other classifier (Fig. 2).

Lets consider the conventional convolutional neural network (LeNet-5) for handwritten digits classification (Fig. 3) [22]. The input image has size 32×32 . The sliding window with size 5×5 scans the image and the segments of images enter to the layer C1 of neural network. Layer C1 is a convolution layer with 6 feature maps and each feature map contains 28×28 neurons. Layer S2 is a subsampling layer with 6 feature maps and a 2×2 kernel for each feature map. As a result each feature map of this layer contains 14×14 units. Layer C3 is a convolution layer with 16 feature maps and a 5×5

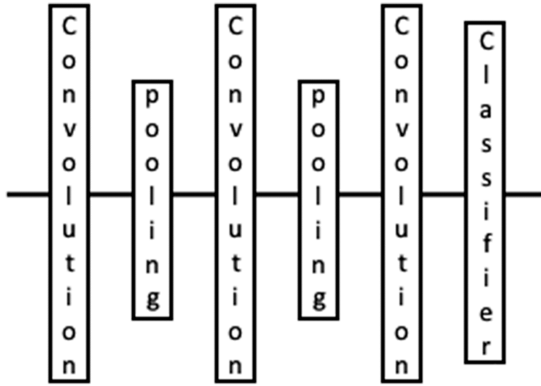


Fig. 2. General representation of convolutional neural network

kernel for each feature map. The number of neural units in each feature map is 10×10 . The connections between layers S2 and C3 are not fully connected [22], as is shown in the Table 1.

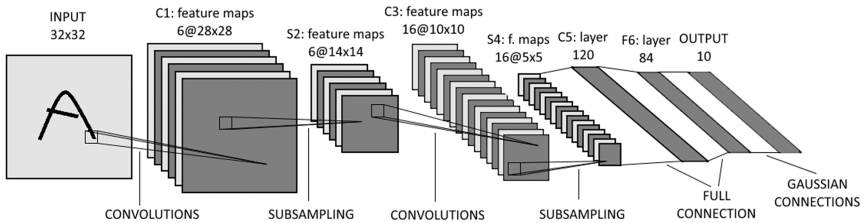


Fig. 3. Architecture of LeNet-5

Table 1. Connections between layers S_2 and C_3

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	X				X	X	X			X	X	X	X		X	X
2	X	X				X	X	X			X	X	X	X		X
3	X	X	X				X	X	X			X		X	X	X
4		X	X	X			X	X	X	X			X		X	X
5			X	X	X			X	X	X	X		X	X		X
6				X	X	X			X	X	X	X		X	X	X

Layer S^4 is a subsampling layer with 16 feature maps and a 2×2 kernel for each feature map. As a result each feature map of this layer contains 5×5 units. Each receptive field with size 5×5 is mapped into 120 neurons of the next layer C^5 . Therefore layer C^5 is a convolution layer with 120 neurons. The next layer F^6 and output layer are fully connected layers.

3 The Simplified Convolutional Network

In this section we proposed convolutional neural network which has a simpler architecture compared with LeNet-5. The simplified convolutional neural network for handwritten digit classification is shown in Fig. 4. This network consists of convolutional layer (C^1), pooling layer (S^2), convolutional layer (C^3), pooling layer (S^4) and convolutional layer (C^5). The convolutional layer C^1 has 8 feature maps and each feature map contains 24×24 neurons. The pooling layer S^2 contains 8 feature maps and 12×12 units for each feature map, i.e. $k = 2$. Layer C^3 is a convolutional layer with 16 feature maps and 8×8 neurons in each feature map. The layers S^2 and C^3 are fully connected in comparison with conventional network LeNet 5. Layer S^4 is a pooling layer with 16 feature maps and 4×4 units for each feature map. The last layer C^5 is the output layer contains 10 units and performs classification. As can be seen the main differences are the following: 1) we removed two last layers in LeNet-5 2) the layers S^2 and C^3 are fully connected 3) the sigmoid transfer function is used in all convolutional and output layers. The goal of learning is to minimize the total mean square error (MSE), which characterizes the difference between real and desired outputs of neural network. In order to minimize a MSE we will use gradient descent technique. The mean square error for L samples is defined using outputs of last layer:

$$E_s = \frac{1}{2} \sum_{k=1}^L \sum_{j=1}^m (y_j^k - e_j^k)^2 \quad (8)$$

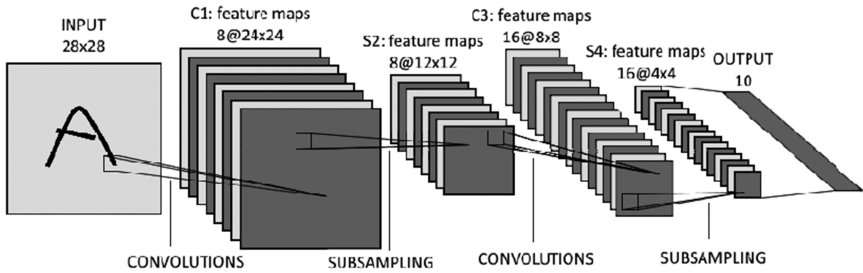


Fig. 4. Architecture of simplified convolutional neural network

where y_j^k and e_j^k – respectively real and desired output of j -th unit for k -th sample. Then using gradient descent approach we can write in case of mini-batch learning, that

$$w_{cij}(t+1) = w_{cij}(t) - \alpha \frac{\partial E(r)}{\partial w_{cij}(t)} \quad (9)$$

where α is learning rate, $E(r)$ is mean square error for r samples (size of minibatch). Since the units of each feature map in convolutional layer contain the same set of weights then the partial derivative $\frac{\partial E(r)}{\partial w_{cij}(t)}$ is equal to the sum of partial derivatives for all neurons of the feature map:

$$\frac{\partial E(r)}{\partial w_{cij}(t)} = \sum_{ij} \frac{\partial E(r)}{\partial w_{cij}(t)} \tag{10}$$

As a result in case of batch learning we can obtain the following delta rule to update synaptic weights:

$$w_{cij}(t + 1) = w_{cij}(t) - \alpha(t) \sum_{ij} \sum_k \gamma_{ij}^k F'(s_{ij}^k) x_c^k \tag{11}$$

where $c = 1, p^2$, $F'(s_{ij}^k) = \frac{\partial y_{ij}^k}{\partial s_{ij}^k}$ – the derivative of activation function for k-th sample, s_{ij}^k – the weighted sum, γ_{ij}^k the error of ij-th unit in a feature map for k-th sample, x_c^k – the c-th input.

4 Experiments

In order to illustrate the performance of proposed technique we present simulation results for handwritten digits classification using MNIST dataset. The MNIST dataset contains 28×28 handwritten digits in gray-scale and has a training set of 60000 samples, and a test set of 10000 samples. You can see some examples of handwritten digits from MNIST data set in the Fig. 5.








Fig. 5. Examples of handwritten digits

Table 2. Comparative analysis

Classifier	Preprocessing	Test error rate (%)
CNN LeNet-1	Subsampling to 16×16 pixels	1.7
CNN LeNet-4	None	1.1
CNN LeNet-4 with K-NN instead of last layer	None	1.1
CNN LeNet-4 with local learning instead of last layer	None	1.1
Convolutional net LeNet-5, [no distortions]	None	0.95
Convolutional net LeNet-5, [huge distortions]	None	0.85
Classifier	Preprocessing	Test error rate (%)
Convolutional net LeNet-5, [distortions]	None	0.8
Simplified convolutional net, [no distortions]	None	0.71
Convolutional net Boosted LeNet-4	None	0.7

We used simple backpropagation algorithm for convolutional neural network training without any modifications. The size of mini batch is 50; learning rate is changed from 0.8 to 0.0001. The results of experiments are illustrated in the Table 2. As can be seen we can achieve test error rate 0.71% using simple shallow convolutional neural network. The best result for convolutional network LeNet-5 without distortions is 0.95%. Thus the use of simplified convolutional network with the elementary backpropagation technique permits to obtain the better performance compared conventional architecture. The processing results of each layer of digit 7 are shown in the Table 3.

Table 3. The results of handwritten digits

Layer type	Number and size of map	Processing results (%)
Input	1@28x28	
Convolutional	8@24x24	
Subsampling	8@12x12	
Convolutional	16@8x8	
Subsampling	16@8x8	

5 Conclusion

This paper deals with a convolutional neural network for handwritten digits classification. We propose a simplified architecture of convolutional neural networks, which permits to classify handwritten digits with more precision than a conventional convolution neural network LeNet-5. The main differences from the conventional LeNet-5 are the following: we removed two last layers in LeNet-5; the layers S^2 and C^3 are fully connected; the sigmoid transfer function is used in all convolutional and output layers. We have shown that simple neural network is capable of achieving test error rate 0.71% on the MNIST handwritten digits classification.

References

1. LeCun, Y., Bengio, Y., Hinton, G.E.: Deep learning. *Nature* **521**, 436–444 (2015)
2. Golovko, V., Bezobrazov, S., Kachurka, P., Vaitsekhovich, L.: Neural network and artificial immune systems for malware and network intrusion detection. In: Koronacki, J., Raś, Z.W., Wierchoń, S.T., Kacprzyk, J. (eds.) *Advances in Machine Learning II. Studies in computational intelligence*, vol. 263, pp. 485–513. Springer, Heidelberg (2010)
3. Golovko V., Kachurka P., Vaitsekhovich L.: Neural network ensembles for intrusion detection. In: *The 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS-2007)*, pp. 578–583, September 2007
4. Dziomin, U.: A multi-agent reinforcement learning approach for the efficient control of mobile robots. In: Dziomin, U., Kabysh, A., Stetter, R., Golovko, V. (eds.) *Advanced in Robotics and Collaborative Automation*, pp. 123–146. River Publishers, San Francisco (2015)
5. Golovko, V., Artsiomenka, S., Kisten, V., Evstigneev, V.: Towards automatic epileptic seizure detection in EEGs based on neural networks and largest Lyapunov exponent. *Int. J. Comput.* **14**(1), 36–47 (2015)
6. Hinton, G.E., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. *Neural Comput.* **18**, 1527–1554 (2006)
7. Hinton, G.: Training products of experts by minimizing contrastive divergence. *Neural Comput.* **14**, 1771–1800 (2002)
8. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. *Science* **313**(5786), 504–507 (2006)
9. Hinton, G.E.: A practical guide to training restricted Boltzmann machines. (Technical report 2010-000). Machine Learning Group, University of Toronto, Toronto (2010)
10. Bengio, Y.: Learning deep architectures for AI. *Found. Trends Mach. Learn.* **2**(1), 1–127 (2009)
11. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Scholkopf, B., Platt, J.C., Hoffman, T. (eds.) *Advances in Neural Information Processing Systems*, vol. 11, pp. 153–160. MIT Press, Cambridge (2007)
12. Golovko, V., Kroshchanka, A., Rubanau, U., Jankowski, S.: A learning technique for deep belief neural networks. In: Golovko, V., Imada, A. (eds.) *Neural Networks and Artificial Intelligence. Communication in Computer and Information Science*, vol. 440, pp. 136–146. Springer, Heidelberg (2014)
13. Golovko, V. From multilayer perceptron to deep belief neural networks: training paradigms and application. In: *Lectures on Neuroinformatics*, pp. 47–84, Moscow (2015)
14. Golovko, V., Kroshchanka, A., Turchenko, V., Jankowski, S., Treadwell, D.: A new technique for restricted Boltzmann machine learning. In: *Proceedings of the 8th IEEE International Conference IDAACS-2015, Warsaw*, pp. 182–186, 24–26 September 2015
15. Golovko, V., Kroschanka, A.: The nature of unsupervised learning in deep neural networks: a new understanding and novel approach. *Opt. Mem. Neural Netw.* **25**(3), 127–141 (2016)
16. Golovko, V., Kroschanka, A.: Theoretical notes on unsupervised learning in deep neural networks. In: *Proceedings of the 8-th International Joint Conference on Computational Intelligence, NCTA 2016, Porto, Portugal*, pp. 91–96, 9–11 November 2016
17. Golovko, V.: Deep neural networks: a theory, application and new trends. In: *Proceedings Of The 13th International Conference On Pattern Recognition and Information Processing (PRIP 2016)*, pp. 33–37. Publishing Center of BSU, Minsk (2016)

18. Golovko, V., Mikhno, E., Brich, A.: A simple shallow convolutional neural network for accurate handwritten digit classification. In: Proceedings of the 13th International Conference on Pattern Recognition and Information Processing (PRIP 2016), pp. 209–212, Publishing Center of BSU, Minsk (2016)
19. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.* **36**, 193–202 (1980)
20. LeCun, Y., Boser, B., Denker, J., Henderson, R., Howard, R., Hubbard, W., Jackel, L.: Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1**(4), 541–551 (1989)
21. LeCun, Y., Haffner, P., Bottou, L., Bengio, Y.: Object recognition with gradient-based learning. In: Forsyth, D.A., Mundy, J.L., di Gesù, V., Cipolla, R. (eds.). LNCS, vol. 1681, pp. 319–345. Springer, Heidelberg (1999). doi:[10.1007/3-540-46805-6_19](https://doi.org/10.1007/3-540-46805-6_19)
22. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2324 (1998)