An approach for automating the design of convolutional neural networks

Dmitry Plotnikov, Evgenii Sopov and Ilia Panfilov
Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russia
E-mail: dmitry.plotnikov.1995@gmail.com, evgenysopov@gmail.com, crook_80@mail.ru

Abstract. Image recognition is an independent field of the computer science nowadays. Image classification is one of its main domains, in which investigated objects can be represented by an image or a video stream. The objective of the image classification is correct assigning of objects to corresponding classes, and there exist many effective approaches for solving this problem. One of the most popular approaches is artificial neural networks, which are a method from the field of machine learning. Despite the fact that neural networks cover a wide range of machine learning problems, they are also able to solve the problem of the image classification. However, there is one more specific approach for neural networks-based images classification that applies the deep learning conception. The best-known deep learning algorithm is called the convolutional neural network (CNN). The CNN uses a principle of using the same parts of a neural network to manipulate with different local parts of an input image. As well as the standard neural network architecture, the convolutional neural network should be fine-tuned for solving a certain problem. Because of the CNN’s depth and complexity, the tuning process usually is complex and needs huge computational efforts. In this study, we have proposed an approach for creating ensembles of previously trained convolutional neural networks. The approach allows to increase the performance of the image classification. The results of experiments for image classification problems are presented and discussed. The experiments show that the proposed approach is able to outperform the standard perceptron and single convolutional neural network.

1. Introduction
Image recognition is one of the most complicated problem for machine learning techniques. For many real-world applications a solution to the problem cannot be described as a set of instructions for a machine learning algorithm. Instead of that it is needed to design a model which will be able to estimate the instances (input data) by itself. For many machine learning techniques, both the structure of the model and its parameters are needed to be fine-tuned. In a case of simple models, the tuning process can be done manually by a human-expert, but for complex image recognition problems simple models do not perform well, thus an automated designing of more complicated structures is preferable.

Starting with the LeNet-5 [1], convolutional neural networks (CNNs) usually had the standard structure included lots of layers (Figure 1). This structure shows perfect results working with MNIST, CIFAR and ImageNet benchmark problems [2, 3].
Figure 1. A convolutional neural network structure (the LeNet-5 architecture).

For many real-world problems, there is no stand-alone algorithm which can achieve a high performance for the given problem. In this case one of the most powerful technique for obtaining the best possible results for the problem is creating an ensemble model. Ensemble models consist of pooling together the predictions of different algorithms, to produce better predictions [4]. There are some popular methods of designing ensemble models, such as the simple voting and the weighted voting. In this study, we will use the simple voting approach in order to take into consideration all predictions from all CNN models. The output of the ensemble will be a label, which polled the majority of the votes.

In this study we will automate the process of CNNs designing by defining some special filters which are used as kernels in the CNN structure. Moreover, we will use an ensemble of previously trained CNN with the identic subsampling layers. The results of experiments are compared with fully connected NN with the same structure as in the fully-connected layers in CNNs.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 describes the proposed approach. In Section 4 the results of numerical experiments are discussed. In the Conclusion the results and further research are discussed.

2. Related work
The term “neural network” has its origins in attempts to find mathematical representations of information processing in biological systems (McCulloch and Pitts, 1943; Widrow and Hoff, 1960; Rosenblatt, 1962; Rumelhart et al., 1986). Indeed, it has been used very broadly to cover a wide range of different models [5].

Before deep learning appeared, David Hubel and Torsten Wiesel performed one of the most important research about striate cortex [6]. They explored parts of the visual cortex which are responsible for detecting edges. They found that some neurons activated when there were vertical lines, other neurons activated when there were horizontal lines. There were also neurons which were responsible for detecting other particular angles. A few later scientists explored that visual cortex is organized as a layered structure. Exactly these ideas affected on designing convolutional neural nets.

Deep Learning and its subfield CNNs appeared not so long ago on a scale of machine learning. Now we can claim that the first CNN was called Neocognitron and it was designed by Fukushima K. in 1979-1980 [7, 8]. Today there is designed a wide range of CNNs architectures. And despite the fact that they show excellent results, the majority of them uses the same ideas. On the other hand, some other approaches become more popular. And one of these approaches is ensemble modeling. Ensemble models usually consist only of CNNs. There are some researches that proves its higher efficiency on various image recognition problems, such as MNIST [9] or FMOW [10].

3. Proposed Approach
Each CNN has three types of layers: convolutional layers, subsampling layers and fully-connected layers. First two layers differ this algorithm from the others. Convolutional layer represents the list of kernels, where each kernel is a special filter applied to the input image. These kernels aimed to find some graphical patterns in original image and in the images, which were processed by the previous layers. The output of this layer is represented by the feature maps. Subsampling layer created to reduce the dimensionality of the problem and the feature maps. This layer pools closely-spaced pixels. Let’s discuss each layer in details.
3.1. Convolutional layer.
As it was mentioned, one of the most important part of convolutional layer is kernel. Kernel represents the process of filter applying to the input image. In practice, various types of kernels are used. What is more, some training techniques are implemented to tune the kernels. Usually the adapted back-propagation algorithm is used. Each filter follows its own purpose. However, the majority of them is used to increase the quality of pixel image. In this paper the following filters were considered:

- Gradient filters (directions: north, south, west, and east). Belong to difference linear filters. These filters are also called edge detection filters.
- Laplacians (kernel = 3x3 pixels, 5x5 pixels). These filters commonly used to increase the sharpness and applied to the images which were previously blurred in case to reduce the noise.
- Line detection filters (incrementation of an angle degree = 45°).
- Sobel operator (lines: horizontal, vertical). Discrete differentiation operator used to create an image with emphasizing edges.
- Sharpening filters (2 levels of sharpness). These filters are used to highlight fine detail in an image or to enhance detail that has been blurred.
- Smoothing filter. These filters are used to clean noised images.

Kernel is just a special kind linear operator applied to the input data. Let’s define the following variables:
- feature map of \( l \)-th layer,
- kernel size = \( 2d + 1 \),
- weight matrix, where \( W_{\text{size}} = (2d + 1) \times (2d + 1) \).

The result of convolution with kernel of size \( 2d + 1 \) can be represented as follows:

\[
y_{l,j} = \sum_{-d \leq a,b \leq d} W_{a,b} x_{l+j-a,j+b}^{l}
\]

where \( y_{l,j} \) - the result of convolution at the \( l \)-th layer, and \( x_{l,j}^{l} \) - the output of the whole the previous layer. In other words, to get the component \((i; j)\) of the next layer, linear operator is applied to the previous layer, so scalar multiplication of pixel values and kernel vector is performed. The example of filter applying is shown in Figure 2.

![Filter applying](image)

**Figure 2.** Filter applying.

3.2. Subsampling layer.
In recognition problems it is possible to pool feature pixels in some cases. First of all, the information about some features existing is more valuable knowledge than the concrete location of feature border pixels. What is more, the dimensionality of some problems is too high and it is needed to reduce the dimensionality. The example of pooling operator is shown in Figure 3. There are some functions can be used, such as finding maximum, minimum, median, etc. And one more parameter which needed to be defined is step.
In many papers maximum function is used. For example, in object detection [11] and human pose detection [12] problems this operator shows better results. Moreover, some researches explored that max-pooling is more efficient function [13].

After applying listed layers and training fully-connected layers, we need to design ensemble models. For choosing the general prediction simple voting is used. There is not a concrete strategy for uniting different CNNs, because each filter determines unique features, and it is hard to predict which combination of extracted features works better. What it more, as we shall see throughout the paper, the CNNs combinations are different in all problems.

4. Experimental Settings and Results
To estimate the proposed approach performance, we have used 3 image recognition problems.

4.1. MNIST.
It is a database of handwritten digit. It is one of the most popular databases in image recognition sphere. It has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image. Dimensionality of the problem equals 784 (each image has a resolution = 28x28 pixels). Number of classes = 10. Distribution destiny of all classes is shown in Figure 4. Basing on Figure 4 we can claim that data is balanced in MNIST dataset. Color model: grayscale.

Figure 3. Pooling operator (kernel = 2x2).

Figure 4. Distribution destiny of classes. MNIST problem.
4.2. CIFAR-10.
It is an object recognition problem. Number of examples = 60000. Dimensionality of the problem: 1024 (each image has a resolution = 32x32 pixels). Number of classes = 10. Each class has 6000 images, so it is a balanced dataset. The classes are following: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. The classes are completely mutually exclusive and there is no overlap between automobiles and trucks, for example. Color model: RGB.

4.3. The ORL Database of Faces.
It is a face recognition problem. This dataset contains a set of face images taken between April 1992 and April 1994 at the lab. There are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement). Dimensionality of the problem: 10304 (each image has a resolution = 92x112 pixels). Number of classes = 40. Color model: grayscale.

4.4. Experimental settings.
Firstly, all the filters listed in Section 3 were used to create and train different CNNs (each CNN had a one specific filter). Results of some filters applying are shown in Figures 5 - 7. After that, the ensemble models were created, based on previously trained CNNs. Highest possible number of CNNs in one ensemble model equals 3.

![Figure 5. Filters applying. CIFAR-10 problem.](image-url)
Figure 6. Filters applying. The ORL Database of Faces problem.

(a) Sharpening filter (1st level)  
(b) Sharpening filter (2nd level)

Figure 7. Filters applying. MNIST problem.

(a) Laplacian (kernel = 3x3 pixels)  
(b) Laplacian (kernel = 5x5 pixels)

4.5. Experimental Results.
For each experiment 10 independent runs were computed. Performance shows mean value of these runs. Performance of fully-connected NN for all considered problems is presented in Table 1.
Table 1. Experimental results of NN on all problems.

| Problem                                      | Performance, % |
|----------------------------------------------|----------------|
| MNIST                                        | 97.52          |
| CIFAR - 10                                   | 48.63          |
| The ORL Database of Faces                    | 95.13          |

Performance of the best single CNNs using one filter is presented in Tables 2-4.

Table 2. Experimental results of best single CNNs. MNIST problem.

| Best used filter                           | Performance, % |
|--------------------------------------------|----------------|
| Sobel (horizontal)                         | 98.11          |
| Line detection filter (vertical)           | 97.48          |
| Laplacian (5x5)                            | 97.33          |

Table 3. Experimental results of best single CNNs. CIFAR - 10 problem.

| Best used filter                           | Performance, % |
|--------------------------------------------|----------------|
| Sobel (vertical)                           | 48.02          |
| Gradient filter (direction: west)          | 47.9           |
| Gradient filter (direction: east)          | 47.33          |

Table 4. Experimental results of best single CNNs. The ORL Database of Faces problem.

| Best used filter                           | Performance, % |
|--------------------------------------------|----------------|
| Sharpening filter (2nd level of sharpness) | 95             |
| Sobel (vertical)                           | 93.62          |
| Sharpening filter (1st level of sharpness) | 92.77          |

Performance of the best ensemble models is presented in Tables 5 - 7.

Table 5. Experimental results of CNNs on the MNIST problem.

| Ensemble model components                   | Performance, % |
|--------------------------------------------|----------------|
| Ensemble 1: Laplacian (3x3), Laplacian (5x5), Sobel (horizontal) | 98.54 |
| Ensemble 2: Laplacian (3x3), Laplacian (5x5) | 96.73 |

Table 6. Experimental results of CNNs on the CIFAR - 10 problem.

| Ensemble model components                   | Performance, % |
|--------------------------------------------|----------------|
| Ensemble 3: Sharpening filter (1st), Laplacian (3x3), Gradient filter (south) | 39.21 |
| Ensemble 4: Sharpening filter (1st), Sharpening filter (2nd level of sharpness), Laplacian (3x3) | 45.06 |

Table 7. Experimental results of CNNs on the The ORL Database of Faces problem.

| Ensemble model components                   | Performance, % |
|--------------------------------------------|----------------|
| Ensemble 5: Sharpening filter (1st), Sharpening filter (2nd) | 98.5 |
| Ensemble 6: Sharpening filter (1st), Sharpening filter (2nd), Laplacian (5x5) | 97.75 |

Some experiment results represent quite close values. In this case wilcoxon signed rank test was computed (Table 8). P-value is a statistical significance level of two vectors (experiment results). If $p - value < 0.05$, it means that there is not a statistical significance.
Table 8. Wilcoxon signed rank test.

| Algorithms         | p-value |
|--------------------|---------|
| Ensemble 1 and NN  | 0.18e-3 |
| Ensemble 2 and NN  | 0.18e-3 |
| Ensemble 3 and NN  | 0.11e-4 |
| Ensemble 4 and NN  | 0.11e-4 |
| Ensemble 5 and NN  | 0.55e-3 |
| Ensemble 6 and NN  | 0.24e-2 |

The experimental results show that creating ensemble models from previously trained CNNs can improve the performance, comparing with a single NN. Ensemble models also outperforms single CNNs, designed only with one filter and give more reliability. What is more, there is not a statistical significance between these algorithms.

5. Conclusions
In this study, an automation of creating an ensemble of various CNNs is considered. All CNNs were designed using different types of image filters for increasing the quality of bitmap images and for detecting graphical patterns in images. The feedforward NN stage in all CNNs is the same for providing the equal conditions of the experiments. The experimental results have demonstrated that the performance of the proposed approach is comparable with the state-of-the-art techniques. Furthermore, the dimensionality of input data has been decreased, thus it is required smaller computational capabilities for implementing the CNN algorithm.

In further work, we will implement the proposed approach using the CUDA parallel computation framework for improving the computational performance and for carrying out more experiments. We will explore a wider range of existing image filters and will try to design new ones. More complicated structures of CNNs and ensemble models will be investigated with various image recognition problems.

Acknowledgements
This research is supported by the Ministry of Education and Science of Russian Federation within State Assignment № 2.1676.2017/PP.

References
[1] LeCun Y et al 1989 Back-Propagation Applied to Handwritten Zip Code Recognition Neural Computation 1(4) 541-51
[2] Krizhevsky A, Sutskever I, and Hinton G 2012 Imagenet classification with deep convolutional neural networks In Advances in Neural Information Processing Systems 25 1106-14
[3] Matthew D and Fergus R 2014 Visualizing and understanding convolutional networks Computer Vision - ECCV 2014 - 13th European Conference, (Zurich, Switzerland, September 6-12, 2014) pp 818-33
[4] Chollet F 2007 Deep Learning with Python Electronic Materials 80(1) 453
[5] Bishop C M 2007 Pattern recognition and machine learning Information science and statistics 1-738
[6] Hubel D and Wiesel T 1959 Receptive fields of single neurones in the cat's striate cortex Journal of Physiology 148, 574-91
[7] Fukushima K 1979 Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position - Neocognitron Transactions of the IECE J62-A(10) 658-65
[8] Fukushima K 1980 Neocognitron: A Self-Organizing Neural Network for a Mechanism of Pattern Recognition Unaffected by Shift in Position Biological Cybernetics, 36(4) 193-202
[9] Chen L 2014 Learning Ensembles of Convolutional Neural Networks (The University of Chicago)
[10] Minetto R, Segundo M P and Sarkar S 2018 Hydra: an Ensemble of Convolutional Neural Networks for Geospatial Land Classification. (Preprint arXiv:1802.03518)

[11] Szegedy C, Toshev A and Erhan D 2013 Deep neural networks for object detection. 2013 Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems (December 5-8, 2013, Lake Tahoe, Nevada, United States) pp 2553-61

[12] Singh D, Balasubramanian V and Jawahar C V 2016 Fine-tuning human pose estimations in videos Applications of Computer Vision (WACV) 2016 IEEE Winter Conference pp 1-9

[13] Scherer D, Muller A and Behnke S 2010 Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition Proc. 20th International Conference on Artificial Neural Networks: Part III, (Berlin, Heidelberg: Springer-Verlag) pp 92-101