CSIT Conference 2021, Yerevan, Armenia, September 27 - October 1

# A Method of Coordinated Optimization of Neural Network Parameters for a Given Set of Images

Rail Gabbasov
Samara National Research University
Samara, Russia
e-mail: rail.g.r@gmail.com

Rustam Paringer
Samara National Research University
IPSI RAS - branch of the FSRC
«Crystallography and Photonics» RAS
Samara, Russia
e-mail: rusparinger@gmail.com

Alexander Kupriyanov
Samara National Research University
IPSI RAS - branch of the FSRC
«Crystallography and Photonics» RAS
Samara, Russia
e-mail: akupr@ssau.ru

David Asatryan
Institute for informatics and automation problems of NAS Armenia
Russian-Armenian university
Yerevan, Armenia
e-mail: dasat@iiap.sci.am

Mariam Haroutunian
Institute for informatics and automation problems of NAS Armenia
Yerevan, Armenia
e-mail: armar@sci.am

*Abstract*—**Today, neural networks are successfully applied to many different problems. Among them, a large class of problems related to computer vision can be distinguished. In this area, the use of convolutional neural networks is particularly successful. Most of the existing neural network architectures are trained on large clusters that require a large amount of computational resources. Therefore, urgent is the task of optimizing neural networks, which can include both increasing performance and reducing the size of the computing power used. In this paper, we propose a method for optimizing (increasing the performance and reducing the amount of consumed resources) of a convolutional neural network, applicable in conditions of redundancy in the input data. Using the Caltech256 dataset and VGG16 network architecture, it was shown that the proposed method can improve network performance by 10% while maintaining accuracy and reducing the amount of resources consumed by 25%.**

*Keywords*—**Neural networks, convolutional network, neural network optimization, knowledge distillation, classification, VGG16, Caltech256.**

## I. Introduction

In general, methods of optimization of neural networks can be divided into three groups [1]:

1. *Optimization of architecture and hyperparameters.* You can replace the architecture with a faster one (for example, change the recurrent neural network to a convolutional one) or use layers that require less computation. The selection of hyperparameters, such as the learning rate, batch size, and the number of learning epochs, can be attributed to the same optimization category.

2. *Model compression*: usually either quantization is used – a decrease in the numerical accuracy of the values of the weights of the network after training, or the so-called pruning – the removal of weights of lesser significance from the model, followed by the formation of sparse matrices of weights.

3. *Knowledge distillation*: a neural network training method in which a smaller model (student) is trained to simulate a previously trained larger model (teacher), that is, the original model is “distilled” into a smaller model [2, 3].

Generally speaking, the problem of optimization of the methods for working with data assumes some data redundancy, the level of which is sufficient so that the optimized method does not produce a worse result than the original one. In this paper, we propose a method for optimizing a convolutional neural network based on the assumption of redundancy in the initial data, which increases performance and reduces the amount of consumed resources: first, the degree of data redundancy is estimated, and then the neural network is directly optimized using the approaches of changing the network architecture and knowledge distillation.

The proposed method is described in the next section.

## II. Proposed Optimization Method

### A. First Stage

At the *first stage* of the proposed method, it is proposed to assess the presence of information redundancy in the initial data by studying the influence of various methods of preliminary distortion of the input data, implying the loss of a certain amount of initial information, on the result of training a neural network on a given set of images. These nine methods are described below.

Each image is divided into 2x2 squares. According to Figure 1, in each such square containing four pixels, the color of green pixels is replaced in the first case by the average color of the image (the arithmetic mean between the colors of all the pixels in the image), in the second – by black (value (0, 0, 0)), in the third – by white (value (1, 1, 1)). So tiling 1/4 of the image with medium / black / white are designated as MH, MH0 and MH1, respectively, 2/4 of the image – as CHB, CHB0 and CHB1, and 3/4 of the image – as GRI, GRI0 and GRI1.

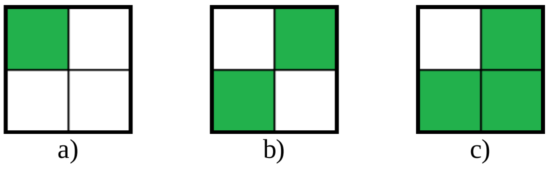

*Figure 1.* Pixels that are being changed during various operations: a) MH, MH0 and MH1; b) CHB, CHB0 and CHB1; c) GRI, GRI0 and GRI1

The DN label indicates the operation of copying an image without modification.

It is proposed to train the neural network on three samples of the initial data using each of 10 tiling methods (including DN for generality). If the result of training, namely, the values of the network loss function together with the value of errors on the test sample, for any of the tiling methods will be comparable to DN or better than DN (that is, the values of the loss function and the error values for this tiling method are equal or less than the same values for the DN case), the fact of the presence of redundancy in the data will be established. In this case, it is worth moving on to the next stage, and otherwise, it should be concluded that the proposed method is not applicable to the dataset studied.

## B. *Second Stage*

At the *second stage*, it is proposed to change the input layer of the convolutional neural network. Namely, it is proposed to impose various binary masks (3x3) on the convolutional filter (3x3) of the input layer. Thus, one can "reproduce" the effect of tiling the original images with black (since multiplying by 0 in the mask is equivalent to filling the pixel with black). And what amount of the original data is “virtually tiled” with black (that is, literally, what amount of the information one is getting rid of), just characterizes the “degree of redundancy” of the input data.

The masks proposed at this stage correspond to various types of image tiling with black, described in the previous stage (i.e., MH0, CHB0, GRI0). This correspondence is achieved due to the fact that the step of the convolutional filter is equal to 2, therefore, the passage of the filter with such a mask over the entire image is "equivalent" to the previously considered operation of tiling an image using a pattern of 2x2 squares, which is highlighted in the following figures of masks by a dashed outline.

It should be noted, however, that the aforementioned "equivalence" is not meant in the strict sense: the tiling effect is not fully reproduced. At the stage with image tiling, the convolutional kernel stride of 1 is used, and at this stage it is equal to 2. Such an increase of the stride of the convolutional kernel, firstly, causes a 2-fold decrease in the spatial dimensions of the feature maps both in width and height, and secondly, reduces the area of intersection of adjacent convolutions.

Figures 2, 3 and 4 show the masks corresponding to the MH0, CHB0 and GRI0 methods, respectively. Each mask is denoted by a binary sequence, which is obtained by reading the mask from top to bottom line by line from left to right.

The experiment was also performed for a mask 111111111, and the results obtained for this case were used as a reference for comparing the effect of each mask on the learning result.

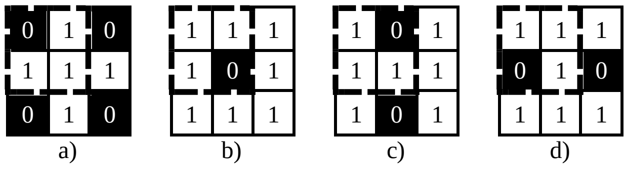

*Figure 2.* Convolutional filter masks, the use of which is "equivalent" to the MH0 operation: a) 010111010, b) 111101111, c) 101111101, d) 111010111

*Figure 3.* Masks of the convolutional filter, the use of which is "equivalent" to the operation CHB0: a) 101010101, b) 010101010

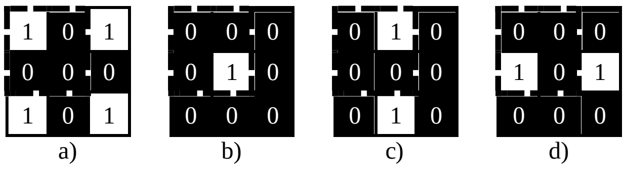

*Figure 4.* Convolutional filter masks, the use of which is "equivalent" to the GRI0 operation: a) 101000101, b) 000010000, c) 010000010, d) 000101000

It is proposed to train the neural network on three samples of the initial data, with the imposition of each of the above masks on the convolutional filter of the input layer (including for generality a mask consisting of all ones – 111111111). If the training result, namely, the values of the network loss function, the values of errors on the test sample, the values of the precision and recall metrics, for the case of applying a certain mask, will be comparable or better than the result with the mask 111111111 (then i.e., the values of the loss function and the error values for that mask usage are equal or less, and the values of the precision and recall metrics are equal or greater than the same values for the result with the mask 111111111), then, depending on which of the operations MH0, CHB0, GRI0 application of this mask is "equivalent", it will be possible to conclude about the "amount of redundant information" in the input data – 1/4, 1/2 or 3/4, respectively. In this case, it is worth moving on to the next stage. If no improvement in the result is observed, but at the first stage the fact of the presence of redundancy in the data was established, it is worth concluding that the "type" of this redundancy, which the data tiling operation made it possible to reveal, cannot be identified / "repeated" with the described at this stage attempt to zero out some pixels. In this case, the proposed method will be inapplicable for the studied data set.

## C. *Third Stage*

At the *third stage*, the network is optimized.

The idea under the optimization is as follows: if there is a disposal of any amount of the data that does not cause a deterioration in the result of the network operation (that is, such that should have been observed in the previous subsection), then the assumption arises that a decrease in the number of training parameters of the network, respectively, for the same amount can have a similar effect of non-degradation of accuracy. Such a “lightweight” network model can be trained using an already trained original network using a knowledge distillation approach.

## III. Experiment Setup

The images were taken from the Caltech256 dataset [4], 10 classes containing an equal number of images were selected (the classification problem was solved). One seventh of the images from each class was used to compose the test sample. The set of other images constituted the training sample, and it was augmented with applying of rotations up to 60 degrees, Gaussian noise with variance up to 20, vertical or horizontal reflections of about half of the images. As a result, the training set contained 9666 images, and the test set – 297. There were three such pairs “test set + training set” prepared. Thus, the idea of k-fold cross-validation with k = 3 is implemented [5].

The experiments were carried out using a neural network with the VGG16 architecture and an input layer size of 128 by 128. For each variant of the experiment, the network was trained using the Adam optimizer [6] with the following parameters: learning rate = 0.001; number of epochs was 20 (100 iterations for each epoch); and the Focal Loss with parameter $\gamma$= 0.5 was used as a loss function.

The described training parameters were selected as a result of tests and observations of changes in accuracy and values of loss functions in the process of training the network on the initial input data (without modification). In each variant of the experiment, the network was trained with the given parameter values five times, and the training results were averaged.

The networks were built and further trained using the MakiFlow framework [7]. When implementing this network model, batch normalization was used on each layer [8].

## IV. Method Verification

### A. *First Stage*

At the *first stage*, the data redundancy was estimated by tiling the original images. Each image entering the input of the neural network was reduced to a size of 128 × 128, and then subjected to one of the modification methods.

The achieved values of the error rate of networks (the proportion of incorrectly classified images from among all images) trained using various methods of preliminary distortion of the input data are presented in Table 1.

*Table 1.* Error rate values of trained networks on a test dataset

| Tiling method | Set #1 | Set #2 | Set #3 |
|---|---|---|---|
| *DN* | *0,36* | *0,38* | *0,32* |
| MH | 0,40 | 0,34 | 0,37 |
| CHB | 0,44 | 0,42 | 0,44 |
| GRI | 0,51 | 0,41 | 0,51 |
| **MH0** | **0,36** | **0,30** | **0,31** |
| CHB0 | 0,36 | 0,33 | 0,34 |
| GRI0 | **0,35** | 0,35 | 0,36 |
| MH1 | 0,44 | 0,37 | 0,35 |
| CHB1 | 0,38 | 0,32 | 0,32 |
| GRI1 | 0,37 | 0,37 | 0,40 |

Analyzing the obtained error values, we can conclude that the MH0 tiling method is the most effective way to reduce the network error in relation to this dataset, which determines the nature of the data redundancy in the dataset.

### B. *Second Stage*

At the *second stage*, the effect of imposing different masks on the convolutional filter of the input layer on the result of training a neural network on a given set of images was investigated.

To assess the results of the experiment, the accuracy, precision and recall metrics were calculated, which are shown in Table 2.

*Table 2.* Overall accuracy, precision and recall values obtained as a result of the experiment

| Convolutional filter mask | Overal accuaracy | Overall precision | Overall recall |
|---|---|---|---|
| *111111111* | *1-0,34* | *0,68* | *0,67* |
| 010111010 | 1-0,33 | 0,68 | 0,67 |
| 111101111 | 1-0,36 | 0,66 | 0,68 |
| **101111101** | 1-**0,34** | **0,69** | **0,68** |
| 111010111 | 1-0,36 | 0,68 | 0,69 |
| 101010101 | 1-0,37 | 0,66 | **0,70** |
| 010101010 | 1-0,36 | 0,67 | 0,69 |
| 101000101 | 1-0,34 | 0,69 | 0,66 |
| 000010000 | 1-0,34 | 0,68 | 0,69 |
| 010000010 | 1-0,34 | 0,67 | 0,70 |
| 000101000 | 1-**0,31** | **0,70** | 0,68 |

It can be seen that lower values of the network accuracy correspond to higher values of the precision and recall metrics. This is in line with expectations, since there is no class imbalance in the dataset [9], and it convinces us that using only the network error as a metric for assessing the quality of the result is justified.

The results obtained allow us to conclude that, in relation to this dataset, in general, the best result was shown by using the mask 101111101. Since the use of this mask is "equivalent" to the MH0 operation, which tilts a quarter of the image, presumably, the "amount of redundant information" in the input data is 1/4. It should be noted that this result corresponds to the result of the first stage (about achieving the best result when using the tiling MH0).

Thus, the amount of information redundancy was estimated – 1/4.

### C. *Third Stage*

At the *third stage*, an experiment was carried out on the application of the knowledge distillation approach for training a network "lightweighted" by 1/4 weights.

The architecture of the lightweight network is shown in Figure 5. This is the original network, from which two layers have been removed as shown in Figure 5.

The number of trained parameters of this network is 11319434, while the original network has 16042058 training parameters. Thus, the "lightweight" network has 29.4% fewer learning parameters, in other words, it is just getting rid of a quarter of the weights that takes place.

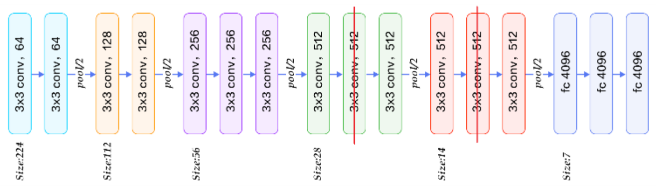

*Figure 5.* "Lightweight" version of the network architecture (removed layers are crossed out)

For each of the three samples, the teacher network (original network) was trained with the same parameters as in previous experiments. The learning parameters of the student network were the same as those of the teacher, except for the main loss function (before the addition of the distillation loss function) – in this case, not Focal Loss, but the usual cross-entropy was used.

The Focal Loss function at $\gamma = 0$ is the same as the Cross Entropy function: the Focal Loss function has historically been proposed as a generalization of Cross Entropy.

The values of the network error rate calculated after training, which are presented in Table 3, make it possible to verify this.

*Table 3.* Values of the network error rate calculated after training

| | Set #1 | Set #2 | Set #3 |
|---|---|---|---|
| *Teacher network* | *0,34* | *0,34* | *0,36* |
| Student network | 0,29 | 0,30 | 0,30 |

The results of measuring the average time spent on the classification of one image by the "lightweight" network are presented in Table 4.

*Table 4.* Average time of classification of one image, ms

| | Set #1 | Set #2 | Set #3 |
|---|---|---|---|
| *Teacher network* | *0,92* | *0,95* | *0,95* |
| Student network | 0,83 | 0,87 | 0,84 |
| Inference time gain | 10% | 8% | 12% |

Based on the results presented in Table 4, there is an acceleration of the network inference time by 10%.

Thus, the knowledge distillation carried out in the presented way made it possible to train a network on a given dataset, which has 29.4% less parameters compared to the original architecture, so that there was an acceleration of the network inference time by 10%. In addition to this, it was also observed that the error rate values of the student network are less than those of the teacher network.

Thus, the proposed method of network optimization, based on the "amount of information redundancy" in the data, really allows one to increase the network performance (in this case, by 10%) and reduce the amount of resources consumed by it (in this case, by 29.4%) while maintaining accuracy.

## V. Conclusion

In this paper, a method was proposed for a convolutional neural network optimization (i.e. increasing performance and reducing the amount of consumed resources), which is coordinated with the data set. The essence of the method consists in a preliminary assessment of the degree of data redundancy and subsequent appropriate optimization of the neural network using approaches of network architecture changing and knowledge distillation.

Experiments were carried out in accordance with the proposed method. The experimental results showed that the amount of data redundancy in the set used can be estimated (using the input layer filtering approach) as 1/4 and that the network model trained using the knowledge distillation approach, "lightened" by 1/4 weights relative to the original model, processes each image is 10% faster, and the error rate values of the optimized network turned out to be not only comparable, but also less than the error values of the original network on the used dataset.

## Acknowledgment

This work was supported by the Russian Foundation for Basic Research and RA Science Committee in the frames of the joint research project RFBR 20-51-05008 Arm_a and SCS 20RF-144 accordingly.

## References

[1] G. Sapunov, (2019) “Speeding up BERT. How to make BERT models faster”. [Online]. Available: https://blog.inten.to/speeding-up-bert-5528e18bb4ea/

[2] C. Bucilua, R. Caruana, and A. Niculescu-Mizil, “Model compression”, *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, Philadelphia, PA, USA, pp. 535-541, 2006.

[3] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network”, *arXiv preprint*, arXiv:1503.02531, 2015.

[4] G. Griffin, A. Holub, and P. Perona, “Caltech-256 object category dataset”, California Institute of Technology, 2007.

[5] M. Stone, “Cross-validatory choice and assessment of statistical predictions”, *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 36, no. 2, pp. 111-133, 1974.

[6] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, *arXiv preprint*, arXiv:1412.6980, 2014.

[7] MakiResearchTeam, (2021) MakiFlow Framework. [Online]. Available: https://github.com/MakiResearchTeam/MakiFlow/

[8] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, *International Conference on Machine Learning*, Lille, France, pp. 448-456, 2015.

[9] N. Japkowicz and S. Stephen, “The class imbalance problem: A systematic study”, *Intelligent data analysis*, vol. 6, no. 5, pp. 429-449, 2002.