Investigation of the capabilities of convolutional neural networks in object classification problems

T V Kozhevnikova\textsuperscript{1}, E V Kadura\textsuperscript{2} and I S Manzhula\textsuperscript{1}

\textsuperscript{1} Computing Center FEB RAS, Kim Yu Chena street, 65, 680000, Khabarovsk, Russia
\textsuperscript{2} Far Eastern State Transport University, Serysheva street, 47, 680000, Khabarovsk, Russia

E-mail: manzhula1994@gmail.com

Abstract. This paper considers convolutional neural networks to solve the problem of improving the quality of images compressed by the JPEG algorithm. To analyze the qualitative characteristics of networks, the convolutional network L04 is implemented using the Caffe framework. By choosing the optimal optimization algorithm for training the network, its qualitative characteristics have been improved.

1. Introduction

Lossy image compression provides high compression ratios by removing information that is not essential to the visual perception of images. Nowadays, lossy image compression is necessary for large companies (for example, Twitter and VK), since it saves network bandwidth when transferring images and space for storing them. However, lossy compression in nature can lead to unwanted artifacts, which significantly degrades image quality. Lossy compression also negatively affects various high-level (search for images by content, classification of objects in images) and low-level image processing tasks that accept compressed images as input. In this regard, the actual problem today is the effective elimination of compression artifacts.

A number of methods have been proposed to eliminate compression artifacts. These include the method that performs filtering along the boundaries of the block [2]; a method for eliminating block artifacts using a discrete cosine transform with an adaptive form [3], which is one of the most popular and others.

In addition to the above methods for eliminating artifacts that occur during image compression, in practice, most programs use simple filters to view images and videos. So, the spp filter from the ffmpeg library eliminates artifacts by reusing JPEG compression for shifted versions of an already compressed image and averaging the results.

In addition to these methods without training, machine learning methods, in particular convolutional neural networks, can be used to eliminate image artifacts. These methods have begun to be used in practice relatively recently, but show better results in quality in comparison with classical methods.
Convolutional neural networks are successfully used in many image restoration tasks: in the superresolution problem, which consists in increasing the resolution of images, as well as in the noise reduction problem, etc. One of the most effective convolutional neural network architectures for eliminating image compression artifacts was proposed by Dong in [4] (we denote this architecture by AR-CNN) and Freedom in [5] (we denote this architecture by L04).

The object of study in this work are images compressed by the JPEG algorithm. The subject is the use of neural networks to improve the quality of JPEG-compressed images.

The aim of the study is to use convolutional neural networks to improve the quality of images compressed by the JPEG algorithm.

To achieve this goal, it is necessary to solve a number of problems, which include the following:

- study the features of convolutional neural networks;
- to study the features of the use of convolutional neural networks to eliminate artifacts when compressing images using the JPEG algorithm;
- implement the network L04;
- get the quality characteristics of the use of networks L04 and AR-CNN;
- to improve the quality characteristics of convolutional neural networks by changing the optimization algorithm used in training the network.

The study is based on a theoretical basis, the basis of which was the scientific research of the following authors: Dong [4], Liberty [5], Kim [6], Yang Qing [7], Kingma [8] and others.

2. Tools for machine training

To implement neural networks that can improve the quality of images compressed by the JPEG algorithm, the Caffe framework will be used. Caffe is a machine learning framework whose main features are expressiveness, speed and modularity. The framework is written in C++ using CUDA for computing on video cards, it also supports MATLAB and Python interfaces for efficient training and deployment of convolutional neural networks and other machine learning models on various architectures. By separating the presentation of the model from the actual implementation, Caffe allows you to experiment and seamlessly transition between platforms. The name is short for Convolution Architecture For Feature Extraction, a convolutional architecture for feature recognition [7].

Caffe provides a complete set of tools for training, testing, tuning and deploying models with well-documented examples for all these tasks. Thus, Caffe has a very low entry defect, which makes it an excellent framework for researchers and other developers seeking to switch to modern machine learning.

The main advantages of Caffe:
- Modularity. The framework was originally developed as modular, which makes it easy to add new data formats, new layers and loss functions. Moreover, there are a large number of already implemented layers and loss functions;
- Separation of presentation and implementation. Caffe model definitions are written as configuration files using the Protocol Buffer standard. Caffe supports network architectures in the form of randomly directed acyclic graphs. After creating the model instance, Caffe reserves exactly as much memory as needed for the model. Switching between CPU and GPU implementation is one function call;
- Coverage with tests. Each individual module in Caffe has a test, and new code is not added to the project without appropriate tests. This allows you to quickly improve the code and refactor the code base;
- Interfaces for Python and MATLAB. For rapid prototyping and interaction with already written code, Caffe provides interfaces for Python and MATLAB. Both languages can be used to build networks.

Table 1 presents a comparison of popular software for convolutional neural networks. Caffe differs from other modern frameworks in that its implementation is completely based on C++, which simplifies integration into existing C++ systems.
Table 1. Comparison software for convolutional neural networks.

| Framework       | Primary language | CPU | GPU |
|-----------------|------------------|-----|-----|
| Caffe           | C++              | +   | +   |
| cuda-convnet    | C++              |     | +   |
| Decaf           | Python           | +   |     |
| Theano / Pylearn2 | Python         | +   | +   |
| Torch7          | Lua              | +   | +   |

3. Methods for eliminating artifacts of compression of images

There are a large number of methods designed to reduce compression artifacts, ranging from relatively simple and fast filters, developed manually, to completely probabilistic methods of image restoration [9] and methods based on advanced approaches to machine learning [10].

Simple artifact removal filters are included in most programs for viewing images and videos. For example, the FFmpeg library includes a simple postprocessing filter (spp) that simply reapplies JPEG compression to shifted versions of an already compressed image and averages the results. The spp filter uses a quantization matrix (compression quality) of the original compressed image, therefore, for decompression, the matrix must be stored with the image.

The most successful method based on the elimination of block artifacts is perhaps the method of eliminating block artifacts using DCT with an adaptive form (Shape-Adaptive DCT - SA-DCT) [3]. However, like most methods for eliminating block artifacts, this method cannot restore sharp differences in brightness and tends to overly smooth small details of images.

In the field of video compression, in the H.264 and H.265 standards, built-in filters are used (block artifact elimination filters and Sample Adaptive Offset filters). However, the release methods in the video, the release of SA-DCT (only for parameter estimation), use the value of the arrangement of DCT blocks. In contrast to these methods, the methods studied in this paper are capable of processing images without such knowledge.

This work focuses on the use of convolutional networks to improve images distorted by JPEG compression artifacts. Recent work in the field of machine learning shows that deep convolution networks can effectively solve various problems of image recovery [11, 12]. Today, convolution-based methods are used in many areas of computer vision.

The AR-CNN model, the qualitative characteristics of which will be further investigated in this paper, is largely close to the model presented in Dong's paper [4] - super-resolution convolutional neural network (SRCNN).

The SRCNN model is designed to solve the super-resolution problem. It was developed based on the principle of sparse coding and contains three layers: a feature extraction layer, a multidimensional display layer, and an image recovery layer. The model with a large number of layers proposed by Kim [11] is based on the work of Dong [10], and shows that deep networks can be trained to solve the super-resolution problem using certain approaches. For training, we used a special strategy for initializing the network weights, and also used the so-called residual learning, in which the network predicts a change in the input image instead of predicting the desired image directly. Residual learning is used in another model considered in this article - L04.

4. Experimental study of the convolutional network L04

To implement the L04 network described in [5], the Caffe framework was used. To store the training sample, the HDF5 (Hierarchical Data Format) format was used. Keeping large collections of small files is a standard issue for file systems. In [14], it was pointed out that the use of key-value storage can significantly accelerate the reading of the training set in comparison with storing the sample in
separate files. HDF5 allows you to break the training set into separate training data sets (batch), access
to which becomes faster. In the case of the L04 network, the size of one set is 64.

Caffe provides ready-made layers for implementing the L04 network:
- an input layer for reading data from the HDF5 storage;
- a convolution layer with a set method for initializing weights (Xavier method);
- loss function layer (standard error);
- summation layer (to summarize the network input with the output of the last convolutional layer to
  implement residual learning).

Since, when applying the convolution operation without expanding the input, the data dimension
decreases, in our case, when the size of the input image is 64 * 64 and when filters are used with the
dimension $f_1 = 11 \times 11; f_2 = 3 \times 3; f_3 = 3 \times 3; f_4 = 5 \times 5$ at the output of the last convolutional layer,
the image dimension is 46 * 46. The summation layer receives data of the same dimension, therefore,
it is necessary to reduce the dimension of the input image. For this, a slicer layer is used, which allows
you to divide the output into several outputs along a given dimension with the indices indicated.

To accelerate the training of neural networks, the Google Cloud Platform was used. This platform
allows the use of graphics processors such as NVIDIA Tesla K80, P100 and V100. Using a GPU can
significantly accelerate learning, compared to learning on a central processor. To deploy the Caffe
framework on the server, the NVIDIA-Docker virtualization environment is used. Docker container
contains all the dependencies for this framework, which makes it easy to port it to other cloud
platforms. Docker add-on - NVIDIA-Docker also allows you to distribute GPU processing power
between containers. The architecture of the virtualization system for training networks is shown in
Figure 1.

![Virtualization System Architecture](image)

**Figure 1. Virtualization System Architecture.**

AR-CNN and L04 use the stochastic gradient method for optimization. In the AR-CNN method,
the learning rate is 0.0001, it is fixed, i.e. does not change during subsequent iterations. In method
L04, the initial learning rate is 0.0005, and it decreases by 2 times every 50 thousand iterations. For
more effective training of the AR-CNN network, the training parameters were selected: the initial
learning rate is 0.0002, it decreases every 100 thousand iterations.

In this work, the BSDS500 [15] and LIVE1 [16] datasets are used. Network training is carried out
on a set of BSDS500, which contains 400 images. Color images were converted to grayscale images
using the YCbCr color model, while only the luminance component, Y, was saved.

Despite the fact that networks can process color images, networks are evaluated using grayscale
images because in this paper we consider the suppression of ringing artifacts and block artifacts, rather
than the suppression of chromatic distortions. Images were compressed using the MATLAB JPEG
encoder and divided into six sets with different compression quality. The following values were used: 10, 20, 30, 40, 50, and 60.

The LIVE1 set containing 29 images is used as a test sample. Images from this set were also converted to grayscale, and compressed with the same encoder.

There are several indicators for an objective assessment of image quality. We use Peak Signal to Noise Ratio (PSNR), Peak Blocked Signal to Noise Ratio (PSNR-B), Structural Similarity Index (SSIM) and Visual Peak Signal to Noise Ratio (VPSNR). Quite often, the PSNR metric does not correlate well with the visual quality of images. The SSIM metric is an evolution of the PSNR metric. It takes into account the “perception of error” due to the structural change in information. The PSNR-B metric can provide additional information, since in this paper we consider block JPEG artifacts.

In some experiments, the IPSNR metric is indicated, which reflects how much PSNR has increased compared to the compressed image PSNR.

For L04 and AR-CNN networks, we will find the most effective optimization method. For all the studied methods, the optimal learning rate is selected by sorting the values on the grid. The learning results of networks are presented in Tables 2 and 3. The number of iterations during training is limited to 150 thousand for ARCNN and 100 thousand for L04.

Table 2. Learning outcomes L04 on the LIVE1 kit using various optimization methods.

| Optimization method | L04          |
|---------------------|--------------|
|                     | PSNR | PSNR-B | SSIM | VPSNR |
| SGD                 | 28.82 | 28.35 | 0.817 | 42.45 |
| RMSProp             | 28.92 | 28.46 | 0.822 | 42.56 |
| Nesterov            | 28.42 | 27.71 | 0.806 | 42.04 |
| Adam                | 28.96 | 28.57 | 0.821 | 42.58 |

Table 3. Learning outcomes of AR-CNN on the LIVE1 kit using various optimization methods.

| Optimization method | AR-CNN          |
|---------------------|-----------------|
|                     | PSNR | PSNR-B | SSIM | VPSNR |
| SGD                 | 28.46 | 27.94 | 0.8091 | 42.07 |
| RMSProp             | 28.80 | 28.44 | 0.8166 | 42.42 |
| Nesterov            | 28.66 | 28.19 | 0.8132 | 42.28 |
| Adam                | 28.82 | 28.45 | 0.8167 | 42.43 |

For both networks, the best result was shown by the adaptive inertia method (Adam). This coincides with the opinion presented in the article [8], in which the Adam method is compared with other optimization methods for training deep convolution networks and shows a good result.

The use of a simple spp post-processing filter from the ffmpeg, AR-CNN, and L04 libraries for recovering JPEG images compressed with quality 10 and 20 was also shown. The number of iterations during training is limited to 250 thousand for L04 and 500 thousand for AR-CNN. To train neural networks, the adaptive inertia method (Adam) was used. The results are presented in tables 4 and 5.
Table 4. Recovery results for images compressed with quality 10 on the LIVE1 set.

| Method  | PSNR  | PSNR-B | SSIM  | VPSNR |
|---------|-------|--------|-------|-------|
| Compressed image | 27.76 | 25.33  | 0.790 | 41.43 |
| spp     | 28.29 | 27.54  | 0.794 | 41.90 |
| AR-CNN  | 28.69 | 28.30  | 0.815 | 42.30 |
| L04     | 28.98 | 28.98  | 28.98 | 28.98 |

Table 5. Recovery results for images compressed with quality 20 on the LIVE1 set.

| Method  | PSNR  | PSNR-B | SSIM  | VPSNR |
|---------|-------|--------|-------|-------|
| Compressed image | 30.07 | 30.07  | 30.07 | 30.07 |
| spp     | 30.33 | 30.33  | 30.33 | 30.33 |
| AR-CNN  | 30.43 | 30.43  | 30.43 | 30.43 |
| L04     | 31.32 | 31.32  | 31.32 | 31.32 |

The L04 network is superior to other methods with both an image quality of 10 and a quality of 20. The results of applying the L04 and AR-CNN networks to an image are shown in Figure 2.

Figure 2. A) Original image B) Compressed JPEG image with quality 10, PSNR = 30.41 C) Image restored with AR-CNN, PSNR = 31.83 D) Image restored with L04, PSNR = 31.92.
We investigate the ability of networks to “remember” images compressed with different quality. To do this, we will train the L04 and AR-CNN network on images with a compression quality of 10, and calculate the quality metrics on images compressed at a quality of 10, 20, 30, ..., 60, etc. To assess the ability of one network to handle many different qualities, we will train the L04 and AR-CNN networks on a training set of images compressed at quality 10, ..., 60. We will designate these networks L04_10-60 and AR-CNN_10-60, respectively. The results are presented in figures 3 and 4.

L04_10-60 provides consistent results for all compression qualities. However, networks trained on only one quality show the best result for a given compression quality. The maximum of graphs for the models L04_20, L04_40, L04_60 is found at the value of compression quality, on which these models were trained.

AR-CNN_10-60 was not capable of effective training to eliminate compression artifacts with different qualities. This may be due to the use in this network of direct mapping from the input of the network to the output, rather than residual learning, as in the L04 network. A network with residual learning only remembers changes caused by compression, and learns more efficiently than a network with direct mapping.

5. Discussion and conclusions
In the course of this scientific study, the features of convolutional neural networks that are used to process data having an ordered, grid structure were studied. They are used to solve many problems in the field of image processing and in other areas, which makes them one of the most used neural network architectures.

To effectively solve the problem of eliminating image compression artifacts using the JPEG algorithm, special convolution network architectures have been proposed, which largely rely on the principle of sparse coding. These include the AR-CNN network, which is a development of the SRCNN architecture that solves the problem of super-resolution by adding a layer to improve the features extracted from the image, as well as the L04 network, which uses the principle of residual learning, which accelerates network learning, and special initialization weights - initialization of Xavier.

To study the qualitative characteristics of networks, in this work, the L04 network was implemented using the Caffe framework, the main features of which are expressiveness, speed, and modularity. Caffe model definitions are written as configuration files using the Protocol Buffer standard. Creating a training set based on images is carried out using a script in the MATLAB language, which writes to the HDF5 repository for quick reading of examples from the training set.
The application of the trained network is also carried out using a script in the MATLAB language. Caffe supports the CUDA library for using GPUs, so Google Cloud Platform was used to speed up network learning, which provides servers with NVIDIA Tesla K80, P100 or V100 graphics accelerators.

Based on the experiment, we can conclude that the L04 network shows the best result compared to simple methods and the AR-CNN network. When researching networks for the ability to “remember” images of different quality, it was revealed that the L04 network trained on images with compression quality 10, ..., 60 is capable of showing stable results for all compression qualities. The AR-CNN network was unable to demonstrate such a result.

In this paper, we managed to improve the qualitative characteristics of convolutional neural networks by changing the optimization algorithm used in training the network. The best result was shown by an algorithm with an adaptive learning rate and Adam inertia.

The networks considered in this paper can be used by companies that use images compressed with a lossy compression algorithm. This will not only improve image quality, but also more efficiently solve various tasks of computer vision, which accept compressed images as input.

Acknowledgments
This research was supported in through computational resources provided by the Shared Services Center “Data Center of FEB RAS” (Khabarovsk) [17].

References
[1] Wallace G K 1992 The JPEG picture compression standard IEEE Transactions on Consumer Electronics 38(1) 18-34
[2] List P 2003 Adaptive deblocking filter IEEE Transactions on Circuits and Systems for Video Technology 13(7) 614–619
[3] Foi A, et al. 2007 Pointwise shape-adaptive DCT for high-quality denoising and deblocking of gray-scale and color images TIP 16(5) 1395–1411
[4] Dong C, et al. 2015 Compression Artifacts Reduction by a Deep Convolutional Network (Department of Information Engineering, The Chinese University of Hong Kong)
[5] Svoboda P, et al. 2016 Compression Artifacts Removal Using Convolutional Neural Networks (Brno University of Technology)
[6] Kim J, et al. 2015 Accurate image super-resolution using very deep convolutional networks CoRR
[7] Jia Y, et al. 2014 Caffe: Convolutional Architecture for Fast Feature Embedding (UC Berkeley EECS, Berkeley)
[8] Kingma D P, Ba K L 2015 Adam: a method for stochastic optimization Conference paper at ICLR
[9] Wong T S, et al. 2009 A document image model and estimation algorithm for optimized JPEG decom pression IEEE Transactions on Image Processing 18(11) 2518–2535
[10] Dong C, et al. 2014 Learning a deep convolutional network for image super-resolution Lecture Notes in Computer Science (LNCS) 184–199
[11] Kim J, et al. 2015 Accurate image super-resolution using very deep convolutional networks CoRR
[12] Lecun Y, et al. 1998 Gradient-Based learning applied to document recognition Proc. of the IEEE
[13] Lim S-H, et al. 2016 An analysis of image storage systems for scalable training of deep neural networks Conference Paper
[14] Malardalen Real-Time Research Center, The worst-case execution time (WCET) analysis project, http://www.mrtc.mdh.se/projects/wcet/
[15] Arbelaez P, et al. 2011 Contour detection and hierarchical image segmentation IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 33(5) 898–916
[16] Sheikh H R, et al. 2015 LIVE image quality assessment database release
[17] Sorokin A A, Makogonov S V, Korolev S P 2017 The Information Infrastructure for Collective Scientific Work in the Far East of Russia Scientific and Technical Information Processing 44(4) 302-304.