Object Detection on Base of Modified Convolutional Network

Alexey Alexeev[1(✉)], Yuriy Matveev[1], and Georgy Kukharev[2]

[1] ITMO University, Saint-Petersburg, Russia
aaalexeev@corp.ifmo.ru, matveev@speechpro.com
[2] West Pomeranian University of Technology, Szczecin, Poland
gkukharev@wi.zut.edu.pl

Abstract. The work involves a new object detector using a convolutional network with a kernel of type NiN (Network in Network). Detection refers to the simultaneous localization of objects on an image and their recognition. The operation of the detector is possible on images of arbitrary size. To learn the network images 100×100 pixels are used. The proposed method has a high computational efficiency, so processing time of HD frame on a single CPU core is about 300 ms. As will be seen from the paper, a high degree of uniformity of network operations creates conditions for streaming parallel processing of data on the GPU, with an estimated operating time of less than 10 ms. Our method is resistant to small overlaps, the average quality of images of detected objects and represents the end-to-end learner model, the output of which is delimited by the boundaries and classes of objects throughout the image. In work, an open dataset of images obtained from car recorders is used to evaluate the algorithm for detecting objects. A similar approach can be used to detect and count other types of objects, for example people's faces. This method is not limited to the use of one type of objects, it is possible to simultaneously detect a mixture of objects. The algorithm of the detector was tested on our own a3net framework, without using third-party neural network programs.

Keywords: Object detection · Region proposal · CNN · NiN

1 Introduction

Recent advances in the detection of objects in images are due to the use of convolutional CNN networks in localization and recognition problems. Gradually, it was possible to significantly reduce the computational capacity of algorithms and simultaneously improve the quality of the systems. Some works divide the tasks of localization and recognition by constructing cascade systems [1], others are so-called end-to-end learning systems that allow a complete picture of

This work was financially supported by the Government of the Russian Federation (Grant 08-08).

R. Vera-Rodriguez et al. (Eds.): CIARP 2018, LNCS 11401, pp. 530–537, 2019.
https://doi.org/10.1007/978-3-030-13469-3_62

the detected objects in one run at the output [2]. A number of algorithms provide work with a fixed resolution of the input image or lead the latter to a predetermined size [2]. Other algorithms that do not have this limitation use a fixed number of candidates for detectable objects in each detector area [1]. Recently, the computationally costly semantic segmentation [3,4] has become popular, which allows you to select objects of interest per pixel. With known forms of objects, it is possible to superimpose the form of the object as a priori information taking into account the calculated position of the object [5]. Most algorithms work with color images. All these approaches have advantages and disadvantages. Either qualitative detection, but the presence of some limitations and low speed of work, or vice versa. The purpose of our work was to find a compromise by creating a fast end-to-end detector of a certain set of objects on images of arbitrary size running on the CPU and providing acceptable quality and speed. In this article, we describe the algorithmic refinement, which consists of the combination of NiN [6–8] as well as the significant repeatability of operations, which ultimately gives an elegant solution that allows the use of parallel stream processing. Using a non-linear convolution kernel in the form of a fully connected network allows you to provide a large stride and to abandon the pooling. Also we refused to take color into account, going to work with luminous images. The offered approaches allowed to achieve high speed of processing, comparable or exceeding speed state-of-the-art algorithms.

2 Related Work

Today, there are many methods for solving the task of detecting objects in images. A brief mention is made in the works [1,2,4,9]. To solve many of them, deep neural networks are used. In these works described the next algorithms: Fast R-CNN, Faster-CNN, YOLO, Mask R-CNN, which are using NN and which are currently the most popular algorithms in the tasks of object detection.

2.1 A Combination of Region Proposals and Object Classification

In [1] (Faster R-CNN method) two levels of CNN and one intermediate Region Proposal Network (RPN) are used. The first performs the role of calculating visual features on the last layer. The intermediate level regression over a set of 9 anchors forms a rectangle of the area and calculates the probability of the presence of objects. The third level provides a classification. This method does not require a fixed image size. In [2] (YOLO method), the emphasis is on combining all tasks to perform a one-pass detection. This is achieved using CNN, which eventually gets a set of areas and their confidence with the subsequent use of non-max suppression to filter and merge them. This allowed the authors to get high FPS on the GPU. The algorithm requires image scaling. The values of the domain detection errors do not depend on the size of the objects, which affects the quality of the definition of small objects. The method is inferior in recognition quality to the Faster R-CNN method, but has a lower false detection

value. The Mask R-CNN method [4] is based on Faster R-CNN, but an instance segmentation is added. It allow make pixel presize segmentation.

2.2 CNN and Convolution via Full Connect Net

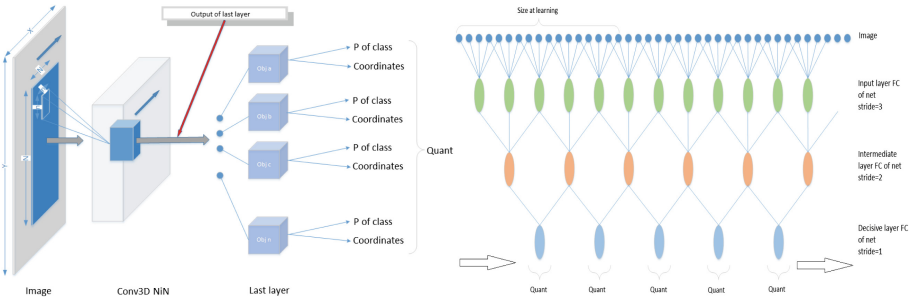
The basis of modern network methods is convolutional (CNN) networks. The standard CNN uses a set of standard linear filters with the kernel $k \in \mathbb{R}^{M,M}$. With a large stride, there are interlacing effects, which adversely affects the quality of the network. The proposed architecture uses non linear classifiers such as NiN [6–8]. At first the idea came to us on its own, later we saw a similar architecture in the works mentioned. Our differences from NiN consist in changing the last level and refusing to use the global average pooling. Instead, it uses the same convolutional NiN layer, the number of which will be determined by the size of the image. Also below is the developed algorithm of backpropagation for the entire network, since nowhere else in the works devoted to NiN did we see its description.

3 A3Net

The human visual system consists of two main subsystems - the primary V1 cortex, and then in the secondary V2 cortex [10]. Formally, our architecture follows this principle, without losing the properties of the end-to-end architecture.

3.1 Network Architecture

The neural network architecture is presented in Fig. 1a. An example of a neural network pyramid is shown in Fig. 1b.



(a) Neural network architecture (b) Neural network pyramid

Fig. 1. (a) Output classifiers are independent of each other and therefore can simultaneously display several classes of images. (b) An example of a three-layer FC neural network pyramid in one of the projections. The number of quants is proportional to the size of the image.

3.2 Working Algorithms

Network Training. Training takes place on images of fixed resolution NxN pixels. When choosing the number of layers, stride and window size, it is desirable to fully cover the NxN area to get maximum recognition efficiency. To do this, it is desirable that the fractional part of the expression be minimal $(Image_{size} - kernel_{size} + 2padding)/stride$. During the training, the parameters {P, x0, y0, w, h} were calculated.

3.3 The Algorithm Backpropagation

Direct sequential pass through all layers of images (features set) by a fully connected (FC) network

$$Z_{xy}^{n+1} = FC_{w,b}^{n}(Z_{s*x,s*y}^{n}) \tag{1}$$

where Z is the input or output of FC network, n is the convolution layer, s-stride, xy-coordinates, wb-weighting coefficients of the network.

Direct passage through layers inside the FC

$$FC_{w,b}^{l} => FC : A \to B = a^{l+1} = \sigma(w^{l+1}a^{l} + b^{l+1}) \tag{2}$$

where l is the layer of a fully connected FC network.

Calculating the delta for the last layer

$$\delta^{L} = \nabla_{a} C \odot \sigma'(z^{L}) \tag{3}$$

where C is the cost function, $\odot$-Hadamard product, σ-non linear activation function

Calculation of delta for subsequent layers

$$\delta^{l} = ((w^{l+1})^{T}\delta^{l+1}) \odot \sigma'(z^{l}) \tag{4}$$

Calculating the delta for the zero layer (for use by the last layer of the next parent convolutional layer)

$$\delta^{0} = ((w^{1})^{T}\delta^{1}) \tag{5}$$

Accumulation of delta for the zero layer (for use by the last layer of the next parent convolutional layer)

$$\delta_{n}^{L}(sx, sy)\big|_{x=1,y=1}^{X,Y} = \delta_{n}^{L}(sx, sy) + \delta_{n+1}^{0}(x, y) \tag{6}$$

where XY-dimensions of the image area at the output FC_{n+1}, s-stride.

The error derivatives with respect to the weights b and w are standard.

3.4 Network Features

Optimization of Convergence. To speed up learning, is used the ADAM optimizer [11].

Initializing the Weights. The initial initialization of the weights is carried out through

$$w_{ij} = \sqrt[2]{\frac{2}{N_{l-1}}}\mathcal{N}(\mu, \sigma^2) \tag{7}$$

where N_{l-1} is the number of neurons at layer before.

Activation Functions. Standard LRELU function with $0.01x$ at the negative part. This reduces the effect of the vanishing gradient problem, in addition, the derivative of the function is not computationally resource-intensive. On the last layer of the last FC, the sin(x) activation function is selected.

Cost Function. Standard Error Function MSE (Mean squared error).

Network Settings. The problem with large batch values is poor convergence [12]. So we chose batch = 10. We also consider this low value in the low learning rate parameter value = 0.00001.

Network Regularization. The well-known dropout $r_j^l \sim Bernoulli(p)$ algorithm was not used, since it is not effective for small networks. To ensure a good generalizing ability, augmentation of images of objects was carried out during training. In particular, the center of the object was displaced within a certain window by specifying the parameters {a, b} with respect to the x and y coordinates through the uniform distribution $r_j^l \sim P(i|a, b)$. The parameters {a, b} are chosen equal to the final step of the network, $\{a, b\} = \prod_i stride(i)$.

Stop Network Training. The network stopping during training occurred with the long absence of the accuracy parameter reduction on the validation sample, the size of which was taken 2000 objects images dataset. Recognition results on a testing sample of 2,000 images dataset coincided with the results on a validation dataset.

3.5 Filtering and Merging Areas

To eliminate false detections after regression calculation of regions, their filtration occurs. The value of IoU should exceed 0.6, and the number of areas from different detections should be at least 3. The resulting zone is selected as the average area of all zones with an average central coordinate.

4 Experiments

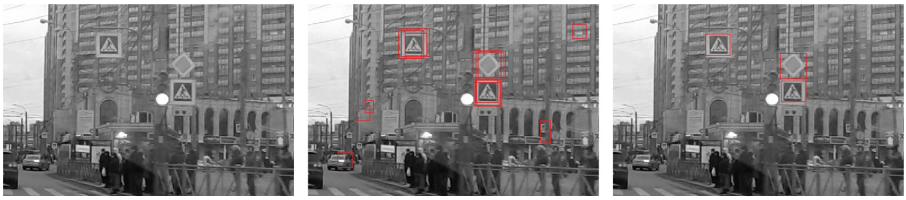
The testing of algorithms and software took place on a PC with an x64 processor Intel (R) Core (TM) i5-2500 CPU @ 3.30 GHz, with a memory of 12 GB. The software in terms of matrix calculations was optimized, including taking into account the hardware capabilities of SSE2, but the work was done only on one core of the CPU.

4.1 Recognition

The experiments were based on the a3net framework and two datasets - MNIST and RTSD. The standard MNIST dataset was used to check the quality of the classification and the overall quality of the framework. The RTSD dataset includes 104,000 HD and FullHD images with resolution from 18 to 120 pixels. We selected only 30 types of traffic signs with limited size in the range from 30 to 85 pixels. The final size of the dataset became 28,000 traffic signs. Recognition results for MNIST and RTSD datasets showed 99.45% and 97.3% respectively.

4.2 Detection

When the object detector was working on images, the percentage of false detections was 3%. This is not so much, given that the work on improving the detector will continue.



(a) Fragment of image before localization

(b) Fragment of image after localization

(c) Fragment image after filtering areas

Fig. 2. Localization of objects in the image taken by the authors on a Nokia3 phone.

5 Conclusions

As a result of the work carried out, a new fast neural network detector of objects working on images of arbitrary size is presented. The high speed of the detector is achieved due to the use of a neural network pyramid with a high stride size. Standard CNN can not afford a big step because of the interleasing effects associated with the fact that the task of filters is to simply correlate the weight coefficients with the luminance values of the image pixels. In our architecture, a large step is possible, since the neural network components are invariant to displacements. From the shortcomings of the work of our algorithms, it is worth noting the presence of false detections (Fig. 2b), the number of which we plan to significantly reduce in the future by including one class classification algorithms, for example [13]. Also, we want to abandon the use of real images for learning and move on to using model objects, which we think give us the capsule approach [14,15], which allows to dynamically calculate the links between layers and thereby cut off the background from the zone of interest along the course of the

detector. We also plan to further develop our neural network framework, include batch-normalization, softmax and crossentropy. The development of our own project helps us better understand the work of the neural network components, and also contributes to the generation of our own ideas.

A Detailed Network Architecture

Net consists of three convolutional layers which are represented by an FC network. The actual network composition used in the work is given in the Table 1. At the output of the network, there are 150 outputs for 30 classes of RTSD dataset objects, since each class is represented by five components {p, x0, y0, width, height}. The neurons of each last FC layer with the next shift of the convolution kernel window form a column on a rectangular area. The number of such columns is determined by the number of window shifts on this convolutional network layer. The connection of these FC columns to the input of the next conv layer occurs via a fully connection between them (Fig. 3).

Table 1. Network composition

Network composition			
	Conv3D 1, 10x10, stride 5	Conv3D 2, 5x5, stride 3	Conv3D 3, 5x5, stride 1
1	FC 1, 64, lrelu	FC 1, 64, lrelu	FC 1, 64, lrelu
2	FC 2, 64, lrelu	FC 2, 64, lrelu	FC 2, 64, lrelu
3	FC 3, 64, lrelu	FC 3, 64, lrelu	FC 3, 64, lrelu
4			FC 4, 150, sin

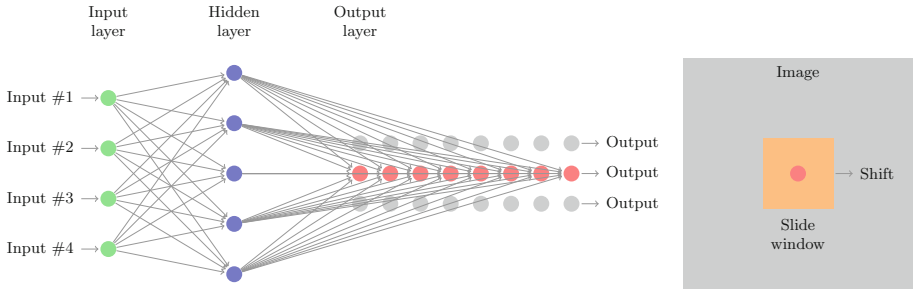


Fig. 3. Showing one of the FC networks, as well as its outputs, represented by a column. The window of the next convolutional layer is shown at the right image. Convolution is in three-dimensional space.

References

1. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. ArXiv e-prints, June 2015
2. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: unified, real-time object detection. CoRR, abs/1506.02640 (2015)
3. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. CoRR, abs/1411.4038 (2014)
4. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. CoRR, abs/1703.06870 (2017)
5. Lee, H.S., Kim, K.: Simultaneous traffic sign detection and boundary estimation using convolutional neural network. CoRR, abs/1802.10019 (2018)
6. Lin, M., Chen, Q., Yan, S.: Network in network. CoRR, abs/1312.4400 (2013)
7. Pang, Y., Sun, M., Jiang, X., Li, X.: Convolution in convolution for network in network. CoRR, abs/1603.06759 (2016)
8. Chang, J., Chen, Y.: Batch-normalized maxout network in network. CoRR, abs/1511.02583 (2015)
9. Girshick, R.B.: Fast R-CNN. CoRR, abs/1504.08083 (2015)
10. Laskar, M.N.U., Giraldo, L.G.S., Schwartz, O.: Correspondence of deep neural networks and the brain for visual textures. ArXiv e-prints, June 2018
11. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR, abs/1412.6980 (2014)
12. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: generalization gap and sharp minima. CoRR, abs/1609.04836 (2016)
13. Perera, P., Patel, V.M.: Learning deep features for one-class classification. CoRR, abs/1801.05365 (2018)
14. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. CoRR, abs/1710.09829 (2017)
15. Frosst, N., Hinton, G.E., Sabour, S.: Matrix capsules with EM routing. In: International Conference on Learning Representations (2018)