UDC 004.93

OBJECT LOCALIZATION AND CLASSIFICATION BASED ON CONVOLUTIONAL NEURAL NETWORKS

DENIS VOROBYOV, RYKHARD BOHUSH
Polotsk State University, Belarus

The paper deals with the basic structural elements of the convolution neural network as well as methods for describing and tested methods of localization of objects in the neural network. Effective approaches are proposed for the construction of algorithm of objects localization. The results of the implementation of the algorithm are presented.

Neural networks are used to solve complex problems that require analytical calculations similar to those that the human brain does [1]. The structure of the neural network in the world of programming came straight from biology. With this structure, the machine acquires the ability to analyze and even memorize a variety of information. The most common applications of neural networks are:

– classification - distribution of the parameters of data. For example, the input contains a set of people and you need to decide who of them you will grant a credit to. This work can be performed by a neural network by analyzing information such as age, solvency, credit history and so on;

– prediction - ability to predict the next step. For example, the rise or fall of shares, based on the situation on the stock market;

– recognition - currently the widest application of neural networks. It is used by Google, when you are looking for photos, or in phone cameras when it determines the position of your face and makes it stand out and many more.

Simple localization and classification tasks can be solved quite well with small size datasets. But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. Convolutional neural networks (CNNs) constitute class of models, for which capacity can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images [2].

A convolutional neural network consists of alternating convolution layers, pooling, local response normalization (LRN). A convolution neurons layer uses the same weights. Neurons using the same weights are combined into the card features (feature maps), and maps each neuron characteristics associated with a part of the previous layer of neurons. In the calculation of the network it turns out that each neuron performs convolution (conversion of determining the degree of similarity) of a region of the previous layer (defined by the set of neurons associated with the neuron). Each neuron of m layer is associated with only a part of neurons of a led layer (m-1). Each neuron layer m has one and the same weight. Pooling is a layer spatial association or subsampling layer (performing the functions of reduction of the dimension of feature maps space) from several neighboring neurons of a feature map a maximum one is chosen, i.e. from multiple input signals the highest goes to the neuron output. LRN is a normalization of the contrast layer which does not have adaptive link weights. The same operation is applied to each connection which produces data scaling, increases too small numbers and decreases too big ones.

Each convolution layer has only the width, height and depth Fig.3. In a separate layer of neurons all neurons have the same weight, and each neuron is a convolution filter. A convolution layer allows determining the coordinates of the point and structures that are responsive to this network, while a fully connected layer allows you to define specific structure belonging to this class. To keep track of these data a convolution layer is built in the network. The layer is called class activation mapping (CAM) which is followed by a Global average pooling layer (the layer of a conventional Inner product) [3].

CAM layer will look deep 1024, and with the core 3:

```
layer {
name: "CAM_conv"
type: "Convolution"
bottom: "output"
convolution_param {
num_output: 1024                    #layer depth
pad: 1
kernel_size: 3              #convolution kernel size
group: 2
}
}
}
```

This is followed by normalization of the contrast layer.

```
layer {
name: "CAM_relu"
type: "ReLU"
bottom: "CAM_conv"
}
```

Pooling layer with kernel 14 and step 14, the nucleus size depends on the number of neurons convolution layer, in this layer the size of the convolution happened 196x196x1024:

```
layer {
name: "CAM_pool"
type: "Pooling"
bottom: "CAM_conv"
pooling_param {
pool: AVE
kernel_size: 14                 # core
stride: 14                      # step
}
}
```

Inner Product used for classification network, in this case 1000 classes:

```
layer {
name: "CAM_fc"
type: "InnerProduct"
bottom: "CAM_pool"
inner_product_param {
num_output: 1000                #1000 classes
}
}
```

From an inner product layer coming after CAM we take neuron links weights with the highest value, the number of such neurons will be determined by the number of classes found in the image. If one class is found than one neuron is used, if two classes are found than two neurons are used and so on, i.e., the number of neurons is determined by the number of classes found.

Each output value of a convolution layer is multiplied by the weight value of the link with a neuron of the next layer, and then summed over the depth. As a result we get a 2-D matrix of a value where the probability of finding an object of this class according to these coordinates will be equal, the coordinates of the object will be equal to the coordinates in the resulting matrix:

$$M_c(x,y) = \sum_k w_k^c f_k(x,y) \; , \tag{1}$$

where w_k^c – is the weight of a convolution neuron layer link to neuron 'k' of a complete coherent layer, belonging to class 'c';

$f_k(x,y)$ – is output value of the convolution neuron layer.

Further, for the resulting matrix the average value is selected and the maximum is divided and then we obtain the threshold values which keep out noise. Formula 2 is used for the construction of object contour:

$$Conmap = \frac{\max(M_c(x,y))}{\sum M_c(x,y)} \; . \tag{2}$$

The result is a contour of an object. Fig. 1 shows an example for contours of objects detection.

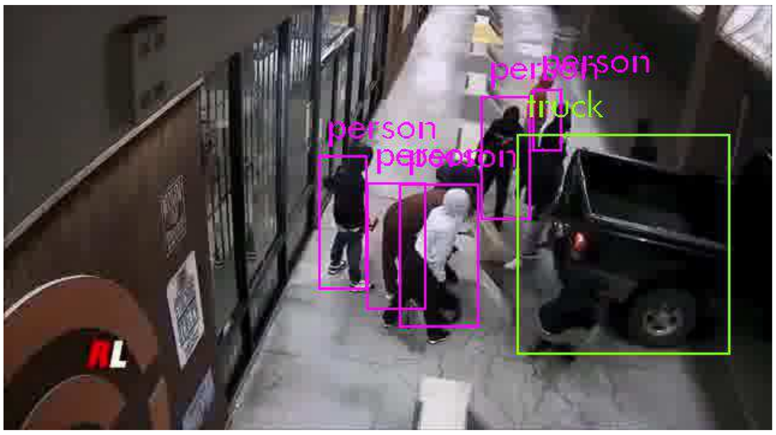

Fig. 1. Example for contours of objects detection

During the experiments, it was found a number of shortcomings of the algorithm such as low accuracy of determining the coordinates of objects in the images that have a large number of objects of the same class. This neural network is characterized by a high working speed. To increase the accuracy of the search the detection of existing facilities was made and for further experiments Faster RCNN was used.

Faster RCNN generates hypotheses about finding the object in the feature space maps conv5 (top convolution) [4]. To do this, run fully convolution 3×3 layer that generates a hypothesis called RPN, moreover, several for different aspect ratios and scale. RPN generates hypotheses and scales them using ROI pooling, and transmits them to a completely coherent layer, perceptron. This layer is trained on the basis of hypotheses to find the coordinates of the object. ROI Poling is identical to Max Pooling except that for ROI Poling the size of the input data is not set in advance it divides the input into the hypothesis of equal size squares, and selects one maximum value.

At the output of the network we get hypotheses and probabilities, which assess the presence of an object. The results are presented in Fig. 2. As you can see the algorithm copes with the task, and has a high accuracy in the images with a lot of objects of the same class.

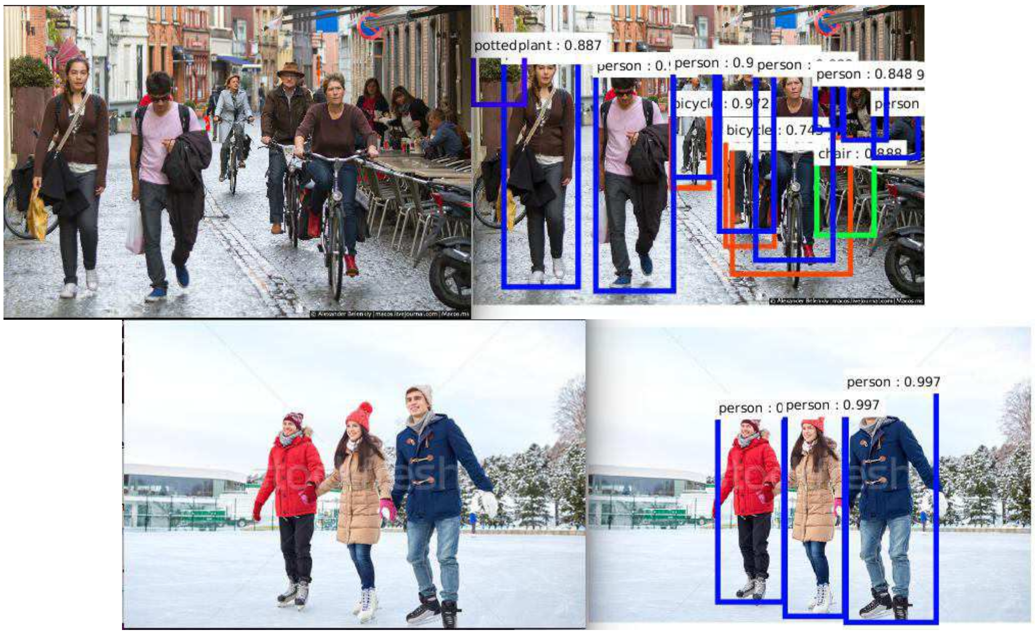

Fig. 2. Example of localization of many objects of the same class using Faster RCNN

Next, we plan to combine two neural networks Faster RCNN and CAM into a single network. CAM network will be used to focus on the most significant elements of the image. Next, to the found image elements hypotheses about the position of objects found on sites will be generated for a subsequent processing by Faster

RCNN method. This approach will significantly increase the speed of Faster RCNN while maintaining the accuracy of the classification.

REFERENCES

1. Neural network & its applications [Electronic resource]/ – Mode of access: http://www.slideshare.net/Ahmed_hashmi/neural-network-its-applications. – Date of access: 10.09.2016.
2. Krizhevsky, A. Image Net Classification with Deep Convolutional Neural Networks [Electronic resource] / A. Krizhevsky, I. Sutskever, G. Hinton // University of Toronto. – 2012. – Mode of access: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. – Date of access: 14.11.2016.
3. Learning Deep Features for Discriminative Localization [Electronic resource] / B. Zhou [et al.]. – Mode of access: https://arxiv.org/pdf/1512.04150v1.pdf. – Date of access: 12.09.2016.
4. Rich feature hierarchies for accurate object detection and semantic segmentation [Electronic resource] / R. Girshick [et al.]. – Mode of access: https://arxiv.org/pdf/1311.2524v5.pdf. – Date of access: 12.09.2016.