Pedestrian detection in video surveillance using fully convolutional YOLO neural network

Conference Paper · June 2017

DOI: 10.1117/12.2270326

5 authors, including:

Vladimir Molchanov
GosNIIAS
2 PUBLICATIONS **14** CITATIONS
SEE PROFILE

Boris V. Vishnyakov
GosNIIAS
26 PUBLICATIONS **79** CITATIONS
SEE PROFILE

Yu. V. Vizilter
GosNIIAS
83 PUBLICATIONS **271** CITATIONS
SEE PROFILE

Oxana V. Vishnyakova
National Research University Higher School of Economics
1 PUBLICATION **12** CITATIONS
SEE PROFILE

Pedestrian detection in video surveillance using fully convolutional YOLO neural network

V. V. Molchanov[a], B. V. Vishnyakov[a], Y. V. Vizilter[a], O. V. Vishnyakova[a], and V. V. Knyaz[a]

[a]The Federal State Unitary Enterprise "State Research Institute of Aviation System", Viktorenko street, 7, Moscow, Russia

ABSTRACT

More than 80% of video surveillance systems are used for monitoring people. Old detection algorithms, based on background and foreground models, couldnt even deal with a group of people, to say nothing of a crowd. Recent robust and highly effective pedestrian detection algorithms are a new milestone of video surveillance systems. Based on modern approaches in deep learning these algorithms produce very discriminative features that can be used for getting robust inference in real visual scene and deal with such tasks as distinguish different persons in a group, overcome problem with sufficient enclosures of human bodies by the foreground, detect various poses of people. In our work we use considerably new approach which enables to combine detection and classification tasks into one challenge using convolution neural networks to accomplish it. As start point we choose CNN YOLO whose authors propose a very efficient way combining mentioned above tasks and converge the problem to learning one neural network, which showed competitive results with state-of-the-art models such as FAST R-CNN overcoming them in speed, which enables applying it in real time visual systems. Despite all advantages it suffers from some know drawbacks related to fully-connected layers that obstruct applying it to images with different resolution and ability to distinguish small close objects in groups which are critical for our tasks since we work with rather low quality images which often include dense small groups of people. In this work we gradually change network architecture to overcome mentioned above problems and finally get network for robust pedestrian detection in real scenes.

Keywords: Video surveillance, abandoned object detection, object classification, convolutional neural networks

1. INTRODUCTION

Nowadays monitoring are subject of interest from video surveillance systems side especially for those one that set their goals to monitor people. As example of using these methods one can suggest different security video systems which aims to detect violators in restricted areas, video monitoring in airports, public spaces, roads etc. Therefore the problem of efficient monitoring is important and can be applied to abundance of different tasks. However, in all these applications we have to be faced problems connected with bad quality images, sufficient enclosures of objects by the foreground, strict conditions imposed on computational and time resources given for particular algorithm. This restrictions influence algorithm's quality as negative factors reducing it's efficiency calculated in assumption to ideal data.

Over the last decade it was proposed a huge number of approaches (more than 1000, including the approaches and their modifications [1,2]) to detect people. Comparative testing and brief overview of the key components of them can be found in [3]. At this point it is obvious that the methods, based on the convolutional neural networks provide much better results than other ones.

Further author information: (Send correspondence to V. V. Molchanov)
V. V. Molchanov: E-mail: vmolchanov@gosniias.ru,
B. V. Vishnyakov: E-mail: vishnyakov@gosniias.ru,
Y. V. Vizilter: E-mail: viz@gosniias.ru,
O. V. Vishnyakova: E-mail: vishnyakova.oxana.pro@gmail.com,
V. V. Knyaz: E-mail: knyaz@gosniias.ru

In the first papers on neural networks for pedestrian detection authors used prior detection systems that repurpose classifiers or localizers to perform detection. They apply the model to an image at multiple locations and scales. Such strategy was exploited in paper which proposed rather complicated algorithm included three successive stages and called R-CNN [4]. In the first stage authors generate large number of region of interests which probably could contain some objects or their parts. There are a lot of technique for doing this. On the next stage to all proposal regions pre-trained on classification task convolution neural network was applied to get discriminative features and then on third stage based on these features authors used per class SVM classifiers for final inference. This approach has some evident drawbacks among which are slow speed that prevent using it in real-time systems and also too complicated architecture that couldn't be learned as a whole one. To eliminate these shortcomings several very similar to each other in architecture neural networks were proposed which let to combine separate stages in previous works into one CNN and converge this task to regression problem, which enables to learn this networks as whole from begin to end. One of such nets is YOLO [5] and DetectNet [6]. We continue discuss YOLO, but DetectNet is very similar. Strictly speaking YOLO isn't a single net but rather a way for building networks for detection purposes. Authors use common CNN which is pre-trained on classification tasks and then add some fully-connected layers on top which are trained on detection task. In their paper their aim to predict bounding boxes for each object on image and correspondent class labels for them. To accomplish this they divide image on regular 7x7 grid and for each grid cell next data are predicted: object bounding boxes sizes and locations relative to the center of the cell, per class probabilities scores. Loss function include both b-boxes term and probabilistic term so classification and detection tasks resolved simultaneously. Thanks to applying net to the whole image rather to only it's parts such nets make their predictions using global context as a result making less mistakes connected with background. Their also show high performance in real time challenges.

Despite all advantages YOLO suffers from some know drawbacks related to fully-connected layers that obstruct applying it to images with different resolution and ability to distinguish small close objects in groups that are critical for our tasks since we work with rather low quality images which often include dense small groups of people. First problem is obvious: fully-connected layers required fix input size and this requirement propagates to input image, so we can't apply YOLO to arbitrary image without resizing it to sizes defined by net. Resizing can lead to geometry deformations which could be harmful and decreasing object sizes which relates to problem with small objects. Challenge with small objects has two sources. First is that YOLO use models pre-trained on data sets such as ImageNet [7] and then fine-tune top layers on PascalVOC [8] or Kitti [9] which don't contain small objects and second YOLO predicts fix number of objects per cell (2 in original paper), so if there are group of small objects which much less compared to cell size then most of them would be cut off.

In this paper we use next approaches to solve mentioned above problems. We first decide issue with fix input image size by replacing fully-connected layers with convolutions what could be made without retraining for original image sizes since fully-connected layers can be treated as convolutions. So whole net transformed into convolution one and could be use as slicing window for images with bigger sizes, Then we address issue connected with objects sizes, we first fine-tune net on our data sets, containing small objects on low quality images, we also use grid with cell sizes comparable with sizes of the smallest objects we want to detect, such approach also used in DetectNet. As a result we get efficient network architecture for robust pedestrian detection in video surveillance.

The remainder of this paper is organized as follows. In Section 2 describes our approach addressing problem of fix input size Section 3 is dedicated to issue which relates to small object sizes, Section 4 describes experimental results and Section 5 concludes this paper.

2. ADAPTATION YOLO TO ARBITRARY IMAGE SIZES

As has been said the way of adaptation YOLO to arbitrary sizes is based on idea of substitution fully-connected layers on convolution layers [10]. Below we describe this method in details.

When image passes to neural network input it goes through all it's layers until fist fully-connected layer, let's denote it's dimension as c_{out} and let input feature map for this layer has dimensions w, h, c_{in}. Fully-connected layer threats this feature map as one-dimensional array with size $w \cdot h \cdot c_{in}$. So weight of this layer forms matrix

$W_{c_{out},w\cdot h\cdot c_{in}}$ and every from c_{out} neurons from this layer has $w \cdot h \cdot c_{in}$ connections with neurons from previous one. Hence, if we threat layer as convolution one and it's neurons are filters of sizes $w \cdot h \cdot c_{in}$ then it helps easily combat problem caused by fully-connected layers. We apply this approach to different network architectures that would be described in more detains further.

In simple case when we use network for images of it's original size this is the only thing that we have to do. Indeed, to convolve input feature map with filter they are both transformed as shown in (Fig. 1) and then performed simple matrix multiplication with successive transformation received result to correct shape. So we simply can initialise filter's weights in convolution layers by values of weights from fully-connected layers, and due to nature of convolution operation we get the same result if we transform it to one dimensional array.

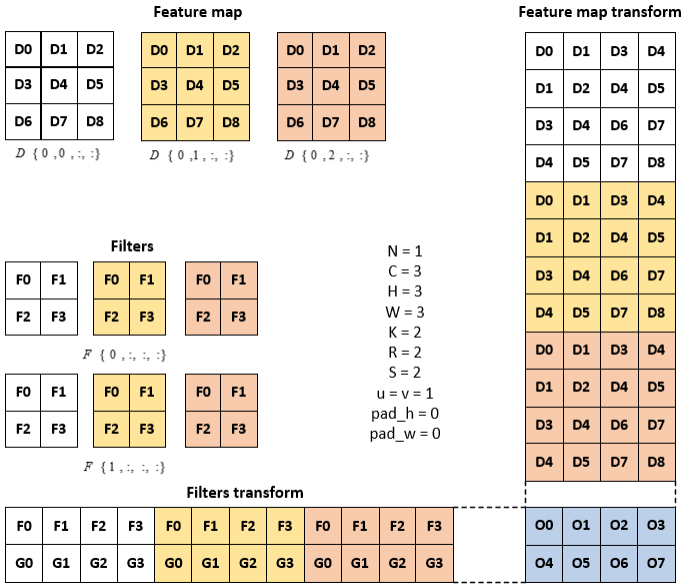

Figure 1. Representing convolution operation as matrix multiplication.

However, it's not valid for images of arbitrary sizes, that can be explained by existence of non zero padding in many layers. To show this, consider next feature map as input to particular convolution layer (Fig. 2). For simplicity let's assume that there is only one filter with sizes 3×3 and feature map has one channel, other layer parameters: $stride = 1, pad = 1$. Together with this feature map consider another one which is gotten from first with additional one row and one column (this is situation when we have large image and cut some part from it).

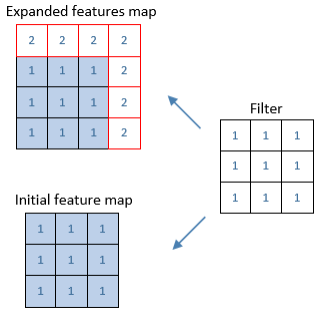

Figure 2. Expanded feature map (size - 4×4) with Initial feature map (size - 3×3) and filter with sizes 3×3.

Convolution result of both these feature maps is present on (Fig. 3) below. From received results it could be concluded that despite similarity of some parts (marked yellow) in resulted maps they are not equal due to side

effects caused by non zero padding. This effects increased with network depth. So by contrast to simple case when image has suitable sizes or zero padding it's could be harmful to use weight from fully-connected layers especially for too large image sizes thereby new convolution layers should be retrained.

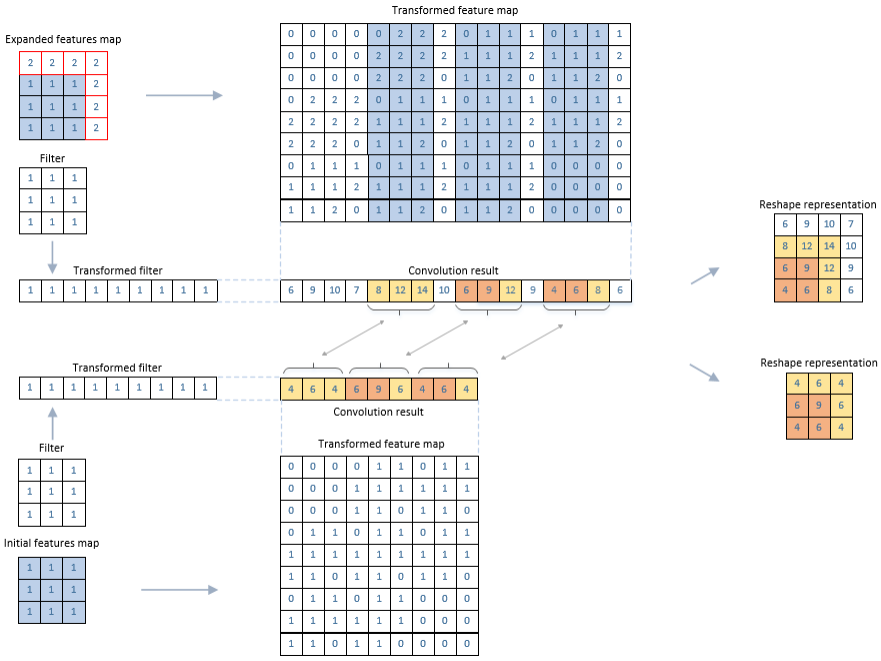

Figure 3. Convolution resault for both fetature maps. Boxes with red borders influence on calculation result due to non zero padding. In resulting maps we marked with orange calor outs that are coincide for both maps and yellow color that arn't.

3. SMALL OBJECT SIZES

As was mentioned above in Introduction YOLO has problems with small objects. The reason is that YOLO use models pre-trained on data sets such as ImageNet and then finetune top layers on PascalVOC or Kitty which don't contain small objects and also YOLO predicts fix number of objects per cell (2 in original paper), so if there are group of small objects which sizes are much less compared to cell size then most of them would be cut off. To address this drawbacks we first retrain new added convolution layers on our data sets, and also decrease cells size to be equal the smallest object that we want to detect.

After network convolutionalization we have two dimensional array of predictions each column of which contains network outputs of the same type like in original yolo-tiny. For example, original yolo architecture was trained on images with $448\times448\times3$ size. Output from last convolution layer has size $7\times7\times1024$. First fully-connected layer has 256 neurons, so we get out with size 256. If fe use image size $640\times640\times3$ as input to our convolution version yolo-tiny, then the same convolution layer gives output size $11\times11\times1024$ and layer instead fully-conected one gives output size $5\times5\times256$.

We then combine these predictions using clustering algorithm called "groupRectangles" from OpenCV. Instead of original way suggested in YOLO paper where intersection over union measure IOU is used to find the best predicted box for ground truth one, it uses special metric to determine measure of similarity between rectangles and based on it, forms clusters. After that, average rectangle calculated for each cluster and announced as prediction result. Algorithm has some configuration parameters: minimal number of rectangles in cluster (if less then cluster isn't take part), metric's sensitivity measure (in extreme cases all rectangles follows up in one cluster or each rectangle forms separate cluster).

4. EXPERIMENTS

In our experiments we base on three original neural network architectures arranged in decreasing power order : yolo, yolo-small, yolo-tiny. The first one possesses the best quality characteristics and could competes with state-of-the-art models in detection tasks. But it's consume to many resources for our particular tasks, so for experiments we focused on yolo-tiny which is much faster and suit for our computation resources budget. This model was designed to be small but powerful. It attains the same top-1 and top-5 performance as AlexNet [11] but with ten times less parameters. It uses mostly convolutional layers without the large fully connected layers at the end. It is about twice as fast as AlexNet on CPU. We change it's fully-connected layers with new convolution ones and retrain model on images from our own data set that contain people of small sizes, starting with 12 pixels wide. This data set is a mix of Caltech, KITTY data sets with our own data gathered from video sequences of Moscow city surveillance system which contain about 500 hundred images with 10-15 persons per image in average. The train process was divided on two stages: at first one we pre-trained new added convolution layers on Caltech [12], KITTI data sets and next we fine-tune it on our own data sets which contain pedestrians of smaller sizes and includes a lot of scenes in night-time and pure quality. During first stage we didn't totaly frozen first layers and allow them to perform adaptation to new data, however we restricts their learning rates by factor 0,1 in comparison with new layers. On final stage we performed learning only for new added layers. On both stages we used SGD RMSProp with exponential decay schedule, started from 0,1 learning rate. We also used Batch Normalisation since first layers were trained with it. We choose Caffe for our experiments and realise yolo's loss layers and migrate yolo-tiny model to it. Figures (Fig. 4) present experimental results on mix data sets which contains images with different sizes.

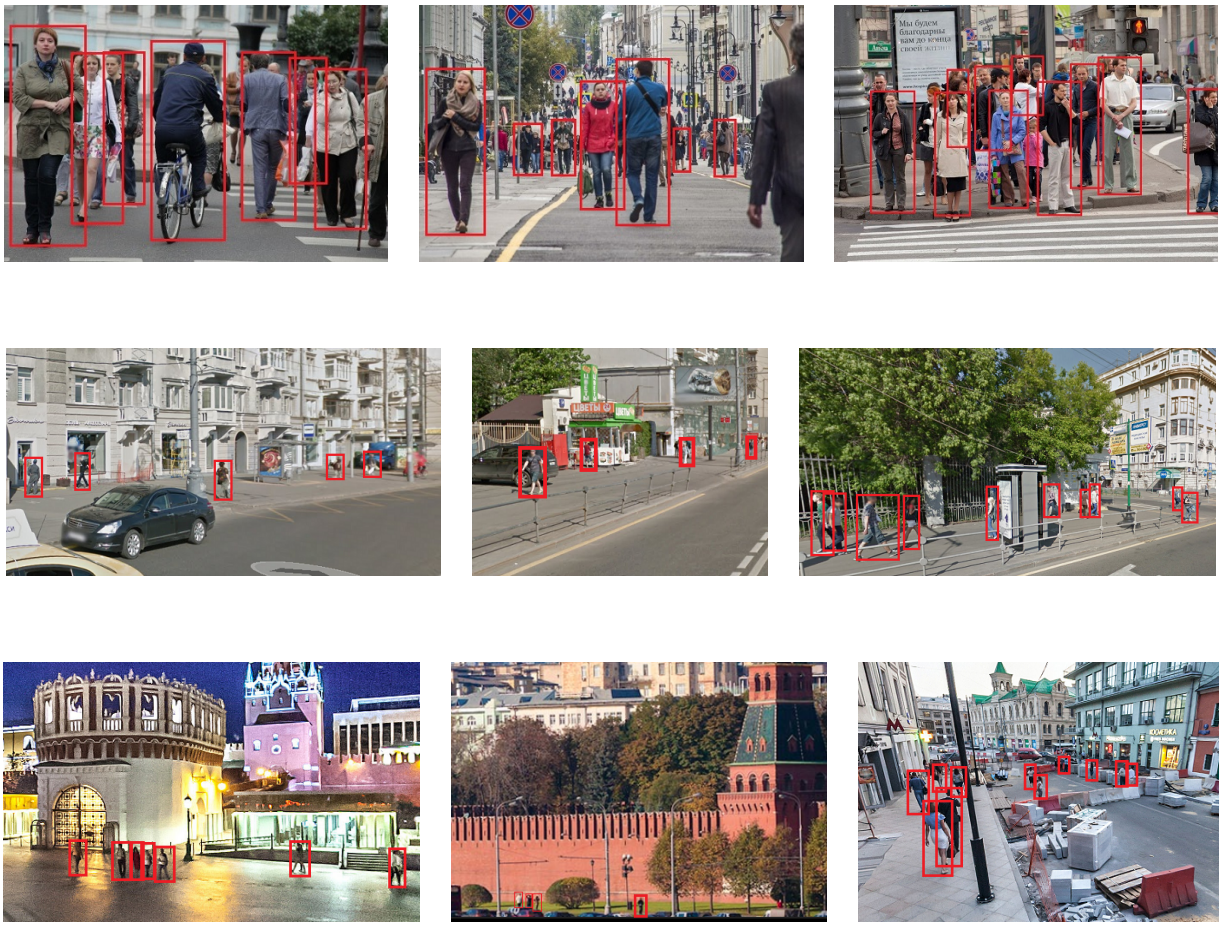

Figure 4. Examples of real bag images from Saint Petersburg Metro.

5. CONCLUSION

In this paper, we apply method for objects detection with single convolution neural network to pedestrian detection in video surveillance. We first adopt it to images with arbitrary sizes that prevents us from retraining network for each particular size and helps to avoid image resizing which leads to geometry deformation and decrease detection quality. It was reached by changing fully-connected layers with convolutions one and train them from scratch. Then we address problem connected with groups of small objects. First of all we train models on our tasks specific data sets with small persons and decrease grid cell size made it approximately equals the smallest object we wand to detect. We also change algorithm for inference final boxes, by splitting all bulk of predictions on clusters and getting average rectangle from it's cluster as final infer.

ACKNOWLEDGMENTS

This work was supported by Russian Foundation For Basic Research (RFBR) under Grants 15-07-09362 .

REFERENCES

[1] Zhang, S., Benenson, R., and Schiele, B., "Filtered channel features for pedestrian detection," *CoRR* **abs/1501.05759** (2015).

[2] Benenson, R., Omran, M., Hosang, J. H., and Schiele, B., "Ten years of pedestrian detection, what have we learned?," *CoRR* **abs/1411.4304** (2014).

[3] Viola, P., Jones, M. J., and Snow, D., "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision* **63**(2), 153–161 (2005).

[4] Ross B. Girshick, Jeff Donahue, T. D. J. M., "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR* **abs/1311.2524** (2013).

[5] Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A., "You only look once: Unified, real-time object detection," *CoRR* **abs/1506.02640** (2015).

[6] Tao, A. and Barker, J., "Detectnet: Deep neural network for object detection in digits." https://devblogs.nvidia.com/parallelforall/detectnet-deep-neural-network-object-detection-digits/.

[7] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)* **115**(3), 211–252 (2015).

[8] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A., "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision* **88**, 303–338 (June 2010).

[9] Geiger, A., Lenz, P., and Urtasun, R., "Are we ready for autonomous driving? the kitti vision benchmark suite," in [*Conference on Computer Vision and Pattern Recognition (CVPR)*], (2012).

[10] Shelhamer, E., Long, J., and Darrell, T., "Fully convolutional networks for semantic segmentation," *CoRR* **abs/1605.06211** (2016).

[11] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," in [*Advances in Neural Information Processing Systems 25*], Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., eds., 1097–1105, Curran Associates, Inc. (2012).

[12] Dollár, P., Wojek, C., Schiele, B., and Perona, P., "Pedestrian detection: An evaluation of the state of the art," *PAMI* **34** (2012).