The Real-Time Detection of Traffic Participants Using YOLO Algorithm

Aleksa Ćorović, Velibor Ilić, Siniša Đurić, Mališa Marijan, and Bogdan Pavković

***Abstract* — Object detection is one of the key software components in the next generation of autonomous cars. Classical computer vision and machine learning approaches for object detection usually suffer from the slow response time. Modern algorithms and architectures based on artificial neural networks, such as YOLO (You Only Look Once) algorithm, solve this problem without precision losses. In this paper we provide the demonstration of the usage of the newest YOLOv3 algorithm for the detection of traffic participants. We have trained the network for 5 object classes (car, truck, pedestrian, traffic signs, and lights) and have demonstrated the effectiveness of the approach in the variety of the driving conditions (bright and overcast sky, snow, fog, and night).**

***Keywords* — Advanced Driver Assistance Systems (ADAS), Convolutional Neural Networks, Deep Learning, Machine Learning, Object Detection, YOLO Algorithm**

I. Introduction

One of the main requirements for autonomous vehicles and most of Advanced Driving Assistance Systems (ADAS) is that they need to be able to perceive and understand their surroundings. Those vehicles are collecting information from their environment using diverse types of sensors such as cameras, LIDARs, radars, ultrasonic devices, etc. Comparing to the other types of sensors, cameras provide the most detailed information about the vehicle's environment in terms of high resolution and texture information. Interpretation and understanding of visual information captured by camera still represent a very complex and challenging task for computers especially if it is necessary to perform it in real time. The changeable lighting and weather conditions, complex backgrounds, and the presence of occluding objects could make this task even more difficult.

Proprietary solutions that are addressing the problem of the detection of traffic participants in the automotive industry are already developed. There are companies specialized for developing state of the art object detection algorithms such as Mobileye, NVidia, StradVision, ViNotion etc. and their solutions are used by major automotive companies such as Tesla. However, all of these solutions are closed and there is not much information publicly available about them.

In order to improve detection performance, those solutions are using computer vision algorithms combined with the sensor's data readings from LIDARs and radars. Unlike those solutions, the solution proposed in this paper only uses input from one camera. On the other hand, there are research papers addressing different approaches for object detection. Some of them, older ones, are about detectors based on Haar cascade classifier, SVMs or sliding window methods [1], but they are outperformed by deep learning [2] models such as deep convolutional neural networks (CNNs). Newer approaches which use deep CNNs, such as Fast R-CNNs [3] or Faster R-CNNs [4], have a problem of slow response time while achieving same or lower mAP value compared to YOLO [5] object detector. Most of those researches do not focus on the specific task such as traffic participant detection, but on object detection in general. The YOLO algorithm, based on CNNs shows outstanding results in object detection tasks while achieving real-time response rate.

In this paper, we present that the YOLOv3 algorithm is capable of accurate object detection (traffic participants) with near real-time performance (~ 25 fps on HD images) in the variety of the driving conditions (bright and overcast sky, snow on the streets, and driving during the night).

The paper is organized as follows. In the Section II, YOLO v3 neural network is presented. The Section III, describes the software solution. The Section IV presents the achieved results. In the final sections, we summarize presented work and give directions for the future work.

II. YOLO v3 Algorithm

YOLO v3 algorithm consists of fully CNN [7] and an algorithm for post-processing outputs from neural network. CNNs are special architecture of neural networks suitable for processing grid-like data topology. The distinctive feature of CNNs which bears importance in object detection is parameter sharing. Unlike feedforward neural networks, where each weight parameter is used once, in CNN architecture each member of the kernel is used at every position of the input, which means learning one set of parameters for every location instead a separate set of parameters. This feature plays important role in capturing whole scene on the road. On *Fig 1* is presented the overview of YOLO v3 algorithm.

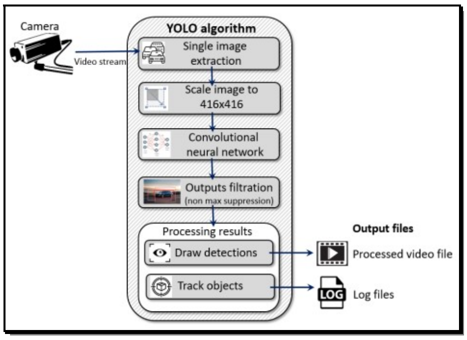


Fig. 1. Overview of Yolo algorithm.

Aleksa Ćorović, Velibor Ilić, Siniša Đurić, Mališa Marijan, Bogdan Pavković, RT-RK, Institute for Computer Based Systems, Novi Sad, Serbia, (e-mail: name.surname@rt-rk.com).

978-1-5386-7171-9/18/$31.00 ©2018 IEEE

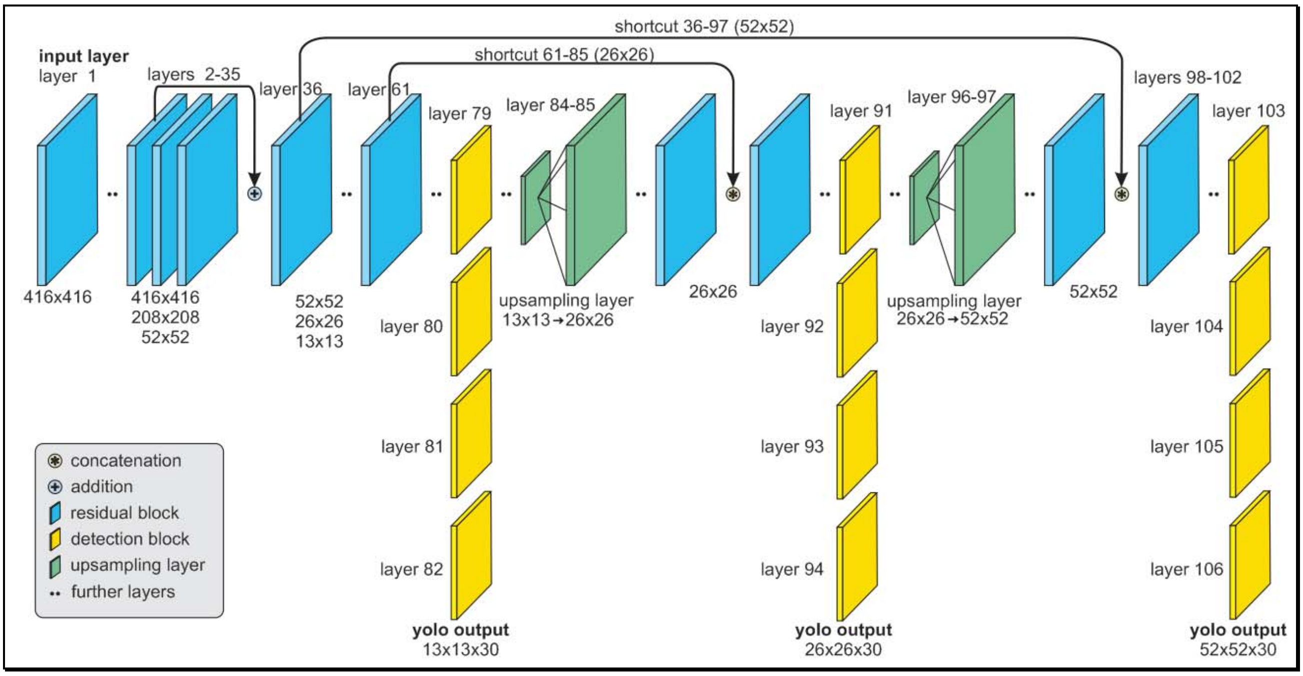


Fig. 2. Yolo network

This algorithm starts with extraction single image from video stream, in a next step extracted image is resized to 416x416 and that represent input to Yolo network.

As it is shown on *Fig 2*, YOLO v3 neural network consist of 106 layers. Besides using convolutional layers, its architecture also contains residual layers [8], upsampling layers, and skip (shortcut) connections.

CNN takes an image as an input and returns tensor (see Fig 3) which represents:

- Coordinates and positions of predicted bounding boxes which should contain objects,
- A probability that each bounding box contains object,
- Probabilities that each object inside its bounding box belongs to a specific class.

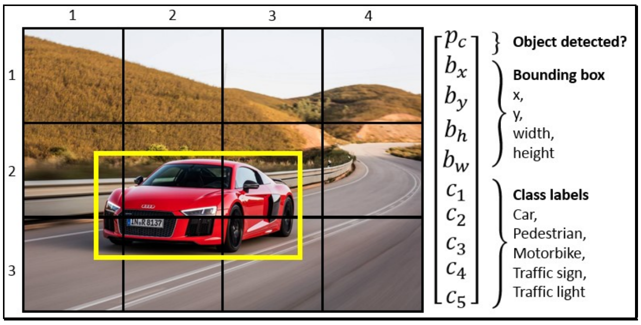


Fig. 3. Bounding box prediction.

The detection is done on the three separate layers, whose input dimensions (width and height) are 13x13, 26x26 and 52x52. Object detection done at 3 different scales addresses [9] the issue of older YOLO neural network architectures, the detection of the small objects. Output tensors from those detection layers have the same widths and heights as their inputs, but depth is defined as:

$$depth = (4 + 1 + class\ probabilities)\ x\ 3,$$

where 4 is the number of bounding box properties such as width (b_w), height (b_h), x and y position of the box (b_x, b_y) inside the image, 1 is the probability that box contains the detectable object (p_c) and class probabilities for each of the classes (c_1, c_2, …, c_5). That sum is multiplied by 3, because each of the cells inside the grid can predict 3 bounding boxes. As the output from the network, we get 10 647 bounding box predictions.

This network has an ability to simultaneously detect multiple objects on the single input image. Features are learned during the network training process when the network analyzes the whole input image and does the predictions. In that way, the network has knowledge about the whole scenery and objects environment, which helps the network to perform better and achieve higher precision results comparing to the methods which use the sliding window approach. The concept of breaking down the images to grid cells is unique in YOLO, as compared to other object detection solutions. The input image is divided into an *S x S* grid of cells where each grid cell can predict 3 bounding boxes.

Predictions whose p_c is lower than 0.5 are ignored and that way, most of the false predictions are filtered out. Remaining bounding boxes are usually prediction of the same object inside the image. They are filtered out using the **non max suppression algorithm**.

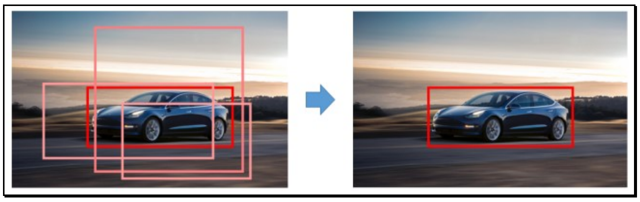


Fig. 4. Filtering multiple detections – non max suppression.

Non max suppression algorithm works as follows. The bounding box with max p_c is compared with all other bounding boxes which has the intersection with it, one by one. If their IoU (Intersection over Union) is greater than 0.5, they are ignored.

In order to enable each grid cell to detect three objects, concept of **anchor boxes** is used. The idea of anchor boxes adds one more "dimension" to the output labels by pre-defining several anchor boxes and dimensions of anchor boxes. That way, we are able to assign one object to each anchor box. For illustration purposes, we'll choose two anchor boxes of two different objects as it is presented on *Fig. 5*.

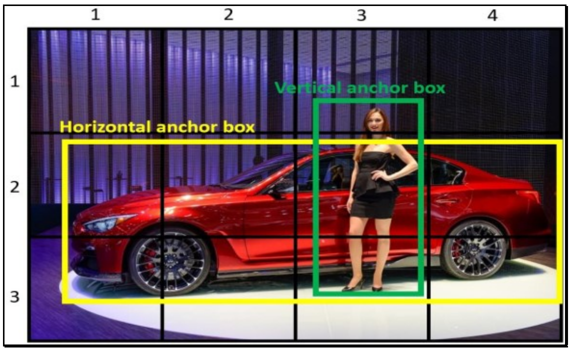


Fig. 5. Anchor boxes.

Dimensions of the anchor boxes are calculated on the training dataset before the training of the network using a k-means algorithm [10].

III. Software For the Detection of Traffic Participants

The software solution is based on Darknet neural network framework written in C programming language. The algorithm used in the solution is presented on Fig 1. The video stream is captured from the forward facing camera placed on the car. Frames captured with the camera are resized and forwarded as the YOLO network's input. After the post-processing outputs, valid detections are drawn on the original frame and forwarded to the output file stream. Furthermore, valid detections from the last three frames are also analyzed and taken into account, when visualizing current frame predictions.

After detecting the object in the current frame, search for that object is performed in the last three frames. Depending on the dimensions and class of the object, previous detections are filtered out. Also, the size of the area in the image where the search is performed depends on the dimensions of the object. For example, the bigger the object is, the larger the search area is. If the object is not found in the search area, new unique identification number is assigned to it, otherwise it takes the identification number from the detection in the one of the three previous frames.

In this way, using a simple class and position and dimensions matching, tracking detected objects across multiple frames is achieved. Also, all of the valid detections and their relevant information such as the frame number, class, position, and the probability are logged.

The YOLO v3 neural network was trained and evaluated on NVidia GeForce GTX 1060 GPU. The Berkley Deep Drive dataset [11], which consists of 70 000 train images and 30 000 validation images, was used for the training. At the beginning of the training, weights were initialized using a model pretrained on the COCO [12] dataset.

The YOLO v3 neural network was trained to detect five classes of objects (traffic participants / road signalization):

- Cars
- Trucks
- Pedestrians
- Traffic signs
- Traffic lights.

Different batch sizes and optimization algorithms were used during the training of the neural network. Initially, stochastic gradient descent was used and batch size was 128 images. Learning rate parameter was set on $10e^{-5}$. After 75 epochs, when the loss function stopped decreasing on plateau, learning rate was increased to $10e^{-3}$ to get out from the plateau. This change in learning rate enabled the loss function value to start decreasing again. Also, batch size was changed to 256 images and optimization algorithm was changed to Adam optimizer.

As illustrated in *Fig 6.* and *Table 1.* the change to Adam optimizer lead to more improvements of the neural network, because there was a more substantial decrease to the loss function while mAP (mean average precision) value was steadily increasing.

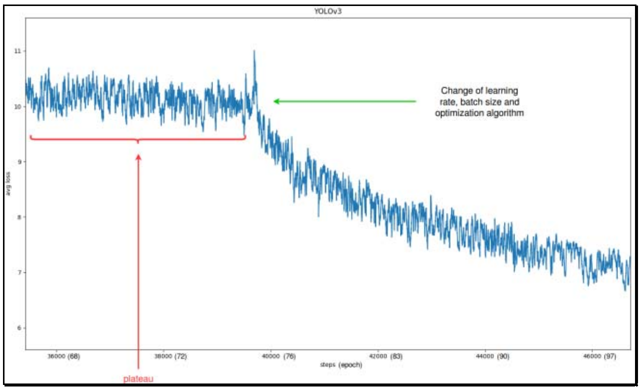


Fig. 6. Decreasing average loss function value by increasing learning rate parameter.

For testing and validation purposes, a small dataset with 300 images showing various traffic situations from city of Novi Sad, Serbia was made and manually labelled. Alongside it, dashboard camera video covering different weather and lighting conditions was also made.

IV. Results

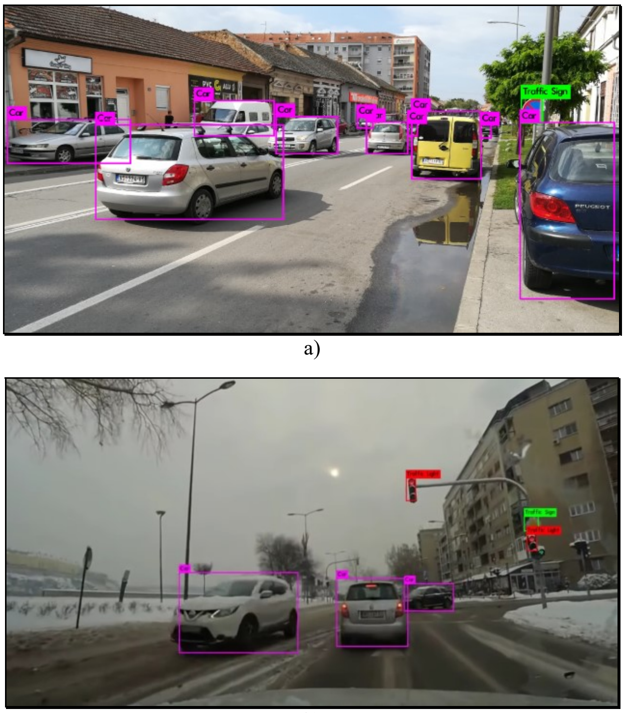
The neural network was trained for 120 epochs and that took two weeks on the previously specified hardware. Key values that were regularly checked during the training were: the loss function value, precision, recall, mAP and average IoU. As seen from the *Table 1*, all of those values were improving until the 120th epoch, when the model started overfitting and then the training was stopped.

TABLE 1: TRAINING YOLOV3 NEURAL NETWORK RESULTS

Epoch	Precision	Recall	F_1 score	*mAP* value	Average *IoU*
40	0.37	0.35	0.36	18.98%	24.19%
47	0.39	0.37	0.38	21.44%	26.12%
56	0.37	0.39	0.38	23.49%	25.44%
75	0.40	0.48	0.44	30.98%	28.12%
90	0.58	0.53	0.56	44.06%	44.06%
109	0.60	0.54	0.57	44.53%	43.65%
120	0.63	0.55	0.59	46.60%	45.98%

Because of the too many small and occluded, but labeled objects in the dataset, precision and recall values were not reaching high values as expected. Common false detections found in the outputs of the neural network were further investigated on the smaller custom dataset and video. Most of the undetected objects were in heavy traffic situations where one cell in the detection grid would be responsible for detecting more than three objects. In the remaining cases, some of the objects were undetected because they were occluded by other detected objects. However, in both previous cases, all of the closest objects to the camera's position (vehicle) were successfully detected and classified as shown in *Fig 7*. Because of this accuracy, the safety of vehicles and passengers will not be put in danger at any moment if using this algorithm for the detection of traffic participants.

The YOLO algorithm processed input video stream from the dashboard camera with the frame resolution 1920 x 1080 px at the average of 23 frames per second using previously described hardware. That confirmed the statement that YOLO algorithm can be used for the real time video processing.



a)

b)

Fig. 7. Detection visualization examples in different weather conditions: a) sunny, b) snowy.

On the following link, results of the YOLO algorithm for the detection of traffic participants can be seen: www.youtube.com/watch?v=nnVPnK-B-P0 .

V. CONCLUSION

In this paper, we have presented application of YOLOv3 algorithm for real time detection of traffic participants. The weights of the neural network were initialized using a pretrained model trained on the COCO dataset. The neural network was further trained on the Berkley Deep Drive dataset to specialize in the detection of five classes of traffic participants. False detections were investigated on the custom dataset made from images representing different traffic situations in Novi Sad.

The main benefit of the solution proposed in this paper is specializing YOLO neural network for the real-time detection and tracking of the multiple classes of traffic participants. Also, the solution provides real-time response, which is required for development of the ADAS components. The application of YOLO algorithm presented in this paper provides a solid base for the object detection module as a part of the ADAS. In the future work, precision of the algorithm could be improved with training on the bigger and more diverse datasets that cover different weather and lighting conditions. Also, this algorithm could be used in fusion with other sensor's readings to reduce the number of false detections.

ACKNOWLEDGMENTS

This work was partially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, under grant number: III44009-1.

REFERENCES

[1] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. IEEE transactions on pattern analysis and machine intelligence, 32(9), 1627-1645.

[2] Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1). Cambridge: MIT press.

[3] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

[4] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

[5] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[6] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[7] Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806.

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[9] Lin, T. Y., Dollár, P., Girshick, R. B., He, K., Hariharan, B., & Belongie, S. J. (2017, July). Feature Pyramid Networks for Object Detection. In CVPR (Vol. 1, No. 2, p. 4).

[10] Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28(1), 100-108.

[11] Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., & Darrell, T. (2018). BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling. *arXiv preprint arXiv:1805.04687*.

[12] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

[13] Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. In Advances in neural information processing systems (pp. 3320-3328).