Traffic Sign Detection with Yolo

Duygu Sesver
Computer Engineering
Istanbul Technical University
Istanbul, Turkey
duygusesver@gmail.com

***Abstract*—Autonomous vehicles are become so popular nowadays. There are a lot of research topic in this field such as lane detection, speed detection, traffic sign detection. In this project, traffic sign detection problem has been considered. You Only Look Once real time object detection system is used to solve this problem. Yolo version 3 [1], yolo version 4 [2] and some data augmentation techniques has been tested on German Traffic Sign Detection Benchmark [3].**

Keywords—YOLO, GTSDB, Object detection, Autonomous Vehicles

I. INTRODUCTION

As you may already know, autonomous vehicles are top research topic in automotive industry. Because of humanity not passed to fully automotive life yet, vehicles need to see what is going on its around and needs to detect other cars existence and should obey the traffic rules. Most of the accidents happen due to lack of watchfulness. AVs will be safer than human drivers if the AVs response fast to the warnings. All the actions should be fast, it should see the warnings first. Traffic sign detection can help AVs, for example when the vehicle sees the stop sign in the street. It should stop and check if there is another car or pedestrian. When detecting traffic signs, couple challenges can be occur. Because of traffic signs will be outside, whether conditions can affect the detection. Moreover, Traffic signs might vary shape, color and might distinguish left and right objects as separate class.

In this project, we worked on traffic sign detection in real time. With the rising of neural networks, deep learning starts to lead these improvements. Since, yolo v4 has just released, we trained the model with yolo v4 configuration on GTSDB and yolo v3 configuration. So, we get the results from both training and compared the results. As a novel contribution, number of train data increased by using augmentation techniques and new dataset has been trained on yolov3 and yolov4 configurations. Finally, we have a comparison on data which applied augmentation and without augmentation.

The purpose of this project is to create a system which detect traffic signs in the roads. Deep neural network model will be trained to detect traffic signs. Model will predict the bounding box of the found object and will be able to recognize what the object is. YOLO is used for this project because it is faster than other detection architectures. There are other fast detection systems such as Fast R-CNN [4] and Faster R-CNN [5]. But Yolo is still six times faster than these methods. Finally, we had a chance to test pros and costs of YOLO [6].

II. RELATED WORK

There are a lot of works in worldwide which use deep neural networks. A lot of detection problems are using neural networks. One of them is road traffic sign detection and classification [7]. Paper has two parts. First part, for the detection, uses color threshold to segment the image, second one, for the classification is uses a neural network.

Second paper is about Detection of traffic sign in real world images: The German traffic sign detection benchmark [8]. The dataset includes real time traffic sign images. The dataset is same as the dataset which we used. The original dataset contains forty-three class, but they grouped the dataset into 4 categories as prohibitory, danger, mandatory and other. In our research, we also reduced forty-three class into four class categories. In addition to these reduced classes, we also grouped it into eight classes. So, we can see how the models are performed in small number of classes dataset and middle number of classes dataset. Their baseline method is a Viola-Jones detector, a linear classifier based on HOG features.

Another research is traffic-sign detection and classification in the wild [9]. This research has two contributions. First one, they've created a dataset from real world images, second one is that they've applied CNN to these images.

III. DATASET

German Traffic Sign Detection Benchmark is used for this project. The original dataset contains road images with traffic signs from German and signs have forty-three labelled classes. forty-three classes grouped into four category such as prohibited, mandatory, danger and other. Six-hundred images for training and three hundred images are split for testing in original dataset. To compare the different results, dataset format has been changed. For the first try, the result between yolov3 and yolov4 on four class datasets are compared. After that, "other" class extracted into priority road, give way, stop, no entry, restriction ends and we get 8 classes at the end. Data distribution is shown in fig. 1. Because of unbalanced data in 8 classes, data augmentation methods are applied to small amount of data. So, to see how data augmentation effect models, two datasets (with augmentation and no augmentation) are used for feeding the yolov3 and yolov4.

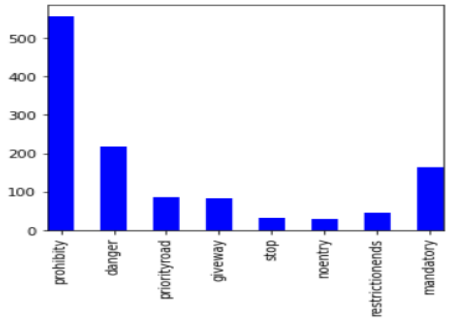

Figure 1 Histogram of 8 class dataset

A. *Data Augmentation*

Since, the dataset with eight class is unbalanced, some augmentation techniques applied specific classes which has small amount of data. In this case, except the images which contains prohibitory signs, all images processed until its count become equal or close to prohibitory classes' number. The processes that applied to images are: Random horizontal flip, randomly scaled, translated, rotated, sheared, resized, changed HSV, and finally most of the images are augmented with combination of all these individual methods. Some of the augmented images are shown fig. 2. Since, objects location changed after augmentation, new bounding boxes are calculated accordingly.

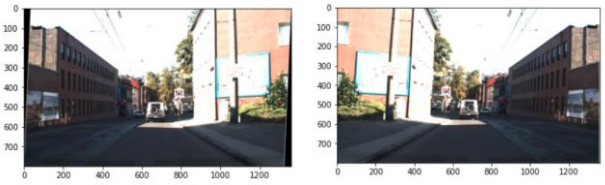

Figure 2 Rotated and flipped images

B. *Dataset reformatting*

To convert German Traffic Sign Detection Benchmark to Yolo format, couple of files needs to be created and restructured. 3 files are created to feed the network (.data, .names, .cfg). One of these files is class.names. This file is created to keep the labels in text file, every new class must be in a new line and line number should match with class number. Other file that needs to be create is sign.data. This file contains information about the train and validation images' paths. The pointed file contains train and test images' paths. Backup is the place where the darknet save the trained weights. Classes represents number of classes and names represents the path to classes.names.

Images are converted from .ppm format to .jpg format. In the same directory between .jpg files, .txt files are created with the same name but .txt extension put class number of object and its location.

For each object in new line: <object-class> <x_center> <y_center> <width> <height>.

Where:

- <object-class> - integer object number from 0 to (classes-number)
- <x_center> <y_center> <width> <height> - float values relative to width and height of image, it can be equal from (0.0 to 1.0].

Augmented images and their annotations are kept in a separate class. Train.txt and test.txt files contain both augmented and without augmented images' paths.

Finally, .cfg file is the one that contains the whole architecture. Since, we are going to train custom objects, some parameters will change.

- Changed batch and subdivision parameters to 64.
- Changed max_batches to 8000 for 4 class training and 16000 for 8 class training (classes*2000)
- Changed steps to 80% and 90% of max_batches. 6400, 7200 for 4 class training and 12800, 14400 for 8 class training.
- Images size set to 608 for width and height.
- All classes parameters are changed to number of classes (4, 8).
- Number of filters in convolutional layer before classes in yolo layer is set to (classes + 5) x 3. (27 for 4 class training, 39 for 8 class training.).

C. *Data flow*

Data flow is shown in Fig. 3. Darknet framework is used for training and testing. So, to train the network, you need to use .data, .cfg and .weights as a parameter. Since, we do not have many data in GTSDB, transfer learning approach is used for training. It was beneficial because you can save time and might get better performance.

When we start training, as initial weight, darknet53.conv.74 and yolov4.conv.137 are used. They are

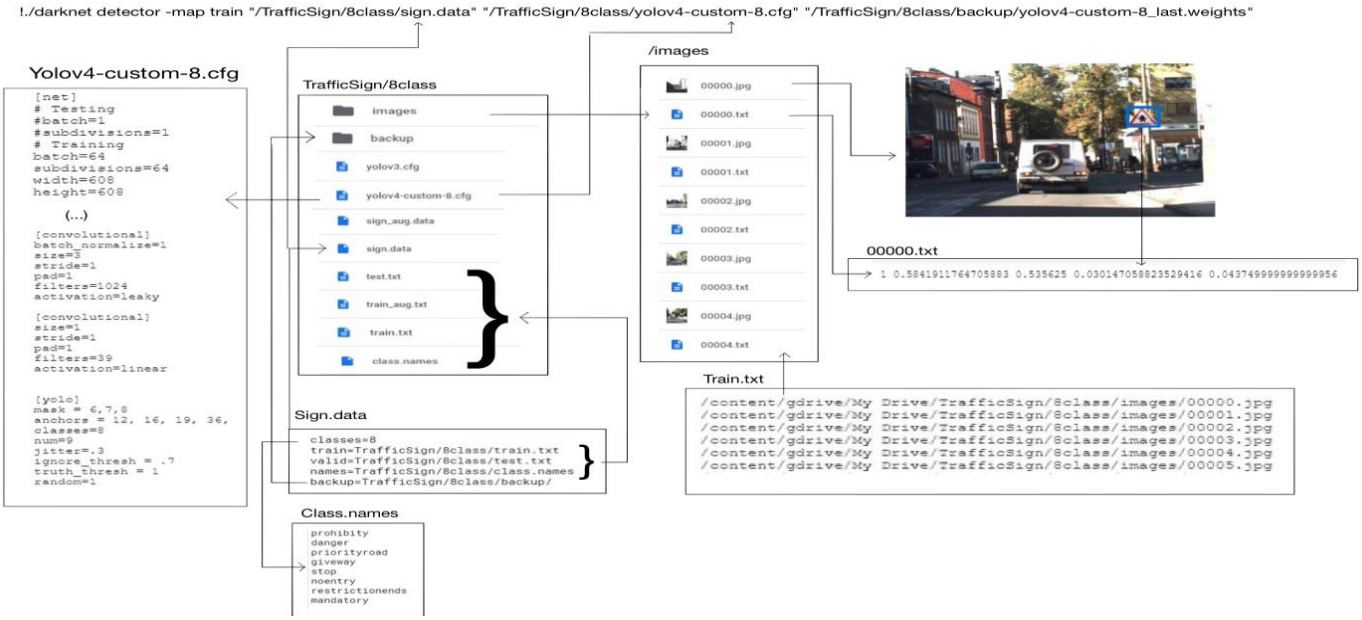

Figure 3 Data Flow

belongs to yolov3 and yolov4 pre-trained weights. They already trained on ImageNet 1000 class. After getting pre-trained model, we trained it with our configurations and saved best weights which gave best accuracy on test set to backup folder.

IV. TRAINING

Yolo architecture is chosen for training because detection should be real-time. Darknet is an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation. So, TESLA K-80 GPU is used in this project.

A. *Yolo Architecture*

They considered object detection as a regression problem. It predicts the objects places in image and predicts what is this object. It has a simple pipeline. There is only one convolutional neural network which predicts the bounding boxes and class probabilities for those boxes. After that, according to threshold, it eliminates some bounding boxes which has lower confidence.

The base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. It detects objects real time indeed with approximately 25 milliseconds of latency.

The reason behind it is fast is that yolo is only look once. it just sees the entire image during training and test time after that it encodes contextual information about classes. Other benefit of YOLO is it learns generalizable representation of objects. It means it can be applied to new domains and unexpected inputs.

Since, yolov4 has just released, we tried yolov4 on selected dataset. Yolov4 architecture has shown Fig. 4.

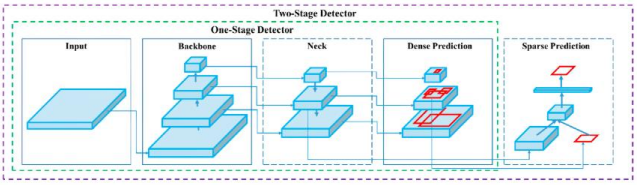

Figure 4 Yolo v4 design

Yolov4 has a lot of new feature comparing to yolov3. It is faster and accuracy is quite high. Most important new features in yolov4 are bag of freebies and bag of specials for backbone. Bag of freebies features for backbone include:

- CutMix and Mosaic data augmentation,
- DropBlock regularization, and
- Class label smoothing

CutOut data augmentation removes a region of an image. This reduces overfitting. **CutMix**, a piece of image is cut and paste to another image. This technique avoids over confidence on specific features. **Mosaic data augmentation** method combines 4 training images into a training image. **Drop block regularization** is like dropout layer in fully connected layers. Some pixels are dropped but remaining parts are still detectable. Class label smoothing is based on an idea that you cannot be sure % 100 percent about a prediction. Label smoothing adjusts the target output upper limit 0.9 instead of 1.

Bag of specials for backbone include:

- Mish activation,
- Cross-stage partial connections (CSP), and
- Multi-input weighted residual connections (MiWRC)

Mish activation contains different function candidates. So, we can make random guesses and pick best performed activation function. **Dense block** contains multiple

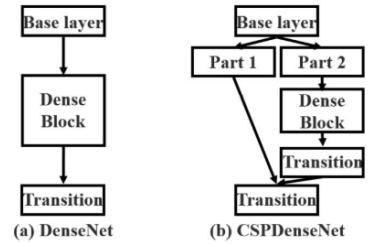

Figure 5 Cross-stage partial connections (CSP)

convolution layers. Instead of using last layer only, next layers will get the previous layers as well as the original as its input. CSPNet separates features into two parts. First one goes to Dense block, second one passed the dense block as shown in fig. 5-b.

Bag of freebies has effect on training process to improve accuracy. Bag of specials impacts the timing in a good way. These two bags are improving accuracy and performance.

V. TESTING

Evaluation metric is mAP with 0.5 threshold. For the mAP calculation, precision and recall metrics are found. If the predicted bounding box contains a correct object, it is called as true positive. If predicted bounding box does not contain correct object, it called false positive. But, if model predicts a bounding box below a threshold, it will be false negative. There is no calculation for true negative because any place might be true negative. It means there is no object and its number would be infinite.

Intersection between ground truth and prediction over union between ground truth and prediction presents confidence value. Another term about yolo is its loss function, it can be shown in Fig. 6. The loss function will adjust to reduce the cost and it will give hint when the model did the prediction wrong. The goal is to reduce the central point's separation between two boxes. First term represents intersection over union between ground truth and predicted bounding box. Second term shows distance where b, b^{gt} are central points, ρ^2 is the Euclidean distance and c is the diagonal length of the smallest enclosing box covering two boxes. α is trade-off parameter, v measures consistency of aspect ratio. Final term represents whole loss function. If IoU overlapped perfectly, then term will be 1 and loss will be 0[10].

$$IoU = \frac{|B \cap B^{gt}|}{|B \cup B^{gt}|} \quad (1) \qquad \mathcal{R}_{DIoU} = \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} \quad (2) \qquad \mathcal{R}_{CIoU} = \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v, \quad (3)$$

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v. \quad (4)$$

Figure 6 Loss function

mAP@0.5	4 Class	8 Class waug	8 Class with aug
Yolo V3	85.86	63.36	74.40
Yolo V4	95.60	75.75	80.57

Figure 7 Accuracy table

When we compare yolo v3 and yolo v4, we can see that there is an improvement in accuracy for the all cases. Let's look at my own augmentation technique how changed the accuracy, on yolo v3 without my augmentation technique, we get 63.36 mAP@0.5 accuracy but when we applied the augmentation technique to images with less number, we can see it increases to 74.40. Same for the yolo v4, augmentation techniques increased accuracy from 75.75 to 80.57. The accuracy table illustrated in Fig. 8 as a graphic.

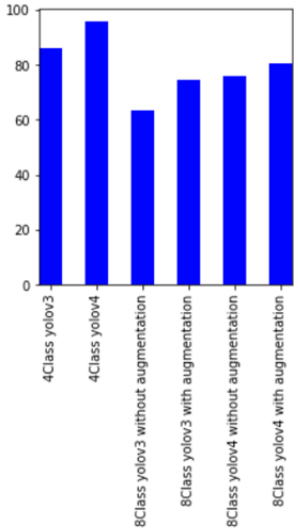

Figure 8 Accuracy Graphic

Figure 9 Some Example

In these pictures, we can see that models are successful to predict traffic sign object. For some pictures, I observe that all models can detect object with different confidence value and for some picture I observe that the models are more successful according to others. The successful model predicts all object, but other models can miss some object. We can see in figure 10 and figure 11,

Figure 10 original image - yolov3 wout aug - yolov3 with aug

Figure 11 original image - yolov3 wout aug - yolov3 with aug

WHAT I LEARNED

I have a lot of contribution to myself from this course. Before this project, I always want to develop a project with yolo. I have a chance to try yolo v3 and yolo v4 in this project. I have learned Yolo's main idea, its basic concept. Since yolo v4 just released, I could try it also in this project and learned new techniques about increasing accuracy and performance. I have learned how to use darknet framework.

I am thinking that I saw all version of models and I have learned a lot.

REFERENCES

[1] J. Redmon and A. Farhadi "YOLOv3: An Incremental Improvement," arXiv, cs.CV, 1804.02767, 2018

[2] A. Bochkovskiy and C. Wang and H. M. Liao "YOLOv4: Optimal Speed and Accuracy of Object Detection", arXiv, cs.CV, 2004.10934, 2020

[3] J. S. Houben and J. Stallkamp and J. Salmen and M. Schlipsing and C. Igel, "Detection of Traffic Signs in Real-World Images: The {G}erman {T}raffic {S}ign {D}etection {B}enchmark", "International Joint Conference on Neural Networks," 1288, 2013.

[4] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.

[5] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.

[6] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[7] A. de la Escalera, L. E. Moreno, M. A. Salichs and J. M. Armingol, "Road traffic sign detection and classification," in IEEE Transactions on Industrial Electronics, vol. 44, no. 6, pp. 848-859, Dec. 1997, doi: 10.1109/41.649946.

[8] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing and C. Igel, "Detection of traffic signs in real-world images: The German traffic sign detection benchmark," The 2013 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, 2013, pp. 1-8, doi: 10.1109/IJCNN.2013.6706807.

[9] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li and S. Hu, "Traffic-Sign Detection and Classification in the Wild," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 2110-2118, doi: 10.1109/CVPR.2016.232.

[10] Z. Zheng and P. Wang and W. Liu and J. Li and R. Ye and D. Ren, "Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression", arXiv, cs.CV, 1911.08287, 2019