GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Computer Vision Project Report

Real-time Object Detection for Autonomous Driving using Deep Learning

Duy Anh Tran, Pascal Fischer, Alen Smajic, Yujin So

15.03.2021

Examiner: Prof. Dr. Gemma Roig

Institute of Computer Science
Department of Computer Science and Mathematics
Goethe University Frankfurt

Contents

1 Introduction

1.1 Object Detection for Autonomous Driving

Autonomous driving is one of the most anticipated technologies of the 21st century and one of the most active research topics at the moment. Autonomous driving attempts navigating roadways without human intervention by sensing and reacting to the vehicles immediate environment [1]. It includes major challenges for Computer Vision and Machine Learning. Object detection is one of the most important requirements for autonomous navigation and consists of localization and classification of objects. Therefore, accurate object detection algorithms are needed. One challenge for example is the processing of numerous candidate object locations (often called "proposals"). These candidates provide only rough localization that must be refined to achieve precise localization [2]. However, solutions to these problems often compromise speed, accuracy, or simplicity [2]. Recent state-of-the-art deep learning models that address the problem of object detection include Region-Based Convolutional Neural Networks (R-CNN) and their improved versions Fast R-CNN and Faster R-CNN, designed for model performance and first introduced in 2013 [3]. A second model for object detection introduced in 2015 is YOLO, designed for speed and real-time use [3].

1.2 The Berkeley DeepDrive (BDD100K) Dataset

Datasets drive vision progress, yet existing driving datasets are limited in terms of visual content, scene variation, the richness of annotations, and the geographic distribution and supported tasks to study multitask learning for autonomous driving [4]. In 2018 Yu et al. [4] released BDD100K, the largest driving video dataset with 100K videos and 10 tasks to evaluate the progress of image recognition algorithms on autonomous driving. The dataset possesses geographic, environmental, and weather diversity, which is useful for training models that are less likely to be surprised by new conditions. Provided are bounding box annotations of 13 categories for each of the reference frames of 100K videos and 2D bounding boxes annotated on 100.000 images for "other vehicle", "pedestrian", "traffic light", "traffic sign", "truck", "train", "other person", "bus", "car", "rider", "motorcycle", "bicycle", "trailer".

1.3 Approach

The goal of our project is to detect and classify traffic objects in a video in real-time using two approaches. We trained the two state-of-the-art models YOLO and Faster R-CNN on the Berkeley DeepDrive dataset to compare their performances and achieve a comparable mAP to the current state-of-the-art on BDD100K, which is 45.7 using a hybrid incremental net [5]. We will focus on the context of autonomous driving and compare the models performances on a live video measuring FPS and mAP.

2 Related Work

2.1 Object Detection Models

Recent object detectors can be divided in one-stage detectors and two-stage detectors [6]. Two-stage detectors detect objects in two stages: a region proposal stage and a classification and localization stage. Compared to one-stage detectors, they achieve higher localization and object recognition accuracy but at the same time lower inference speed. One-stage detectors directly predict boxes from the input image without a region proposal stage and therefore achieve high inference speed and are well suited for real-time use [6].

Two-stage detectors
R-CNN was the first region based CNN for object detection using selective search for region proposals and a CNN for feature extraction on each region proposal. Because of its complex multistage training and slow test time, a faster version named fast R-CNN was proposed. Fast R-CNN extracts the features of an image once and then produces the region proposals, which are then classified. Faster R-CNN, the most representative region based detector, improved the region-based CNN further by using a region proposal network (RPN) instead of an external region proposal method for producing region proposals, which lead to a high speed-up in test time [7]. Mask R-CNN is an extending work to Faster R-CNN adding a fully connected mask head to the network. It is mainly used for instance segmentation tasks [6].

One-stage detectors
YOLO and its newer versions (v2,v3,v4) are one-stage detectors and very popular for real-time object detection because of their high speed. YOLO divides the image into a grid and predicts bounding boxes and class scores in each cell [8]. The Single-shot multibox detector (SSD) directly predicts class scores and box offsets for a fixed set of default bounding boxes of different scales at each location in several feature maps with different scales [6].

2.2 Datasets for Object Detection

Several labeled datasets with bounding boxes were created to tackle specific detection problems. They are used as benchmarks to compare different architectures and algorithms and set goals for solutions. Popular datasets for general object detection are PASCAL VOC, MS COCO (Microsoft Common Object in Context) and ILSVRC (ImageNet Large Scales Visual Recognition Challenge). Datasets for object detection in traffic and driving scenes tailored for autonomous driving are for example:

- KITTI: This dataset covers traffic scenes of the city Karlsruhe. The scenes are divided in 5 categories including annotations (8 classes) and bounding boxes [9].
- Waymo Open: This dataset contains 1950 driving segments and considers 4 object classes. Each segment consists of 20s driving. The data also considers various environmental and urban traffic scenes [10].
- nuScenes: This dataset considers complex urban driving scenes with 3D object annotations for autonomous driving. It contains 1000 scenes of 20s each, 23 object classes, around 1,4 million camera images of the cities Boston and Singapore [11].

3 System Description

3.1 YOLO (You Only Look Once)

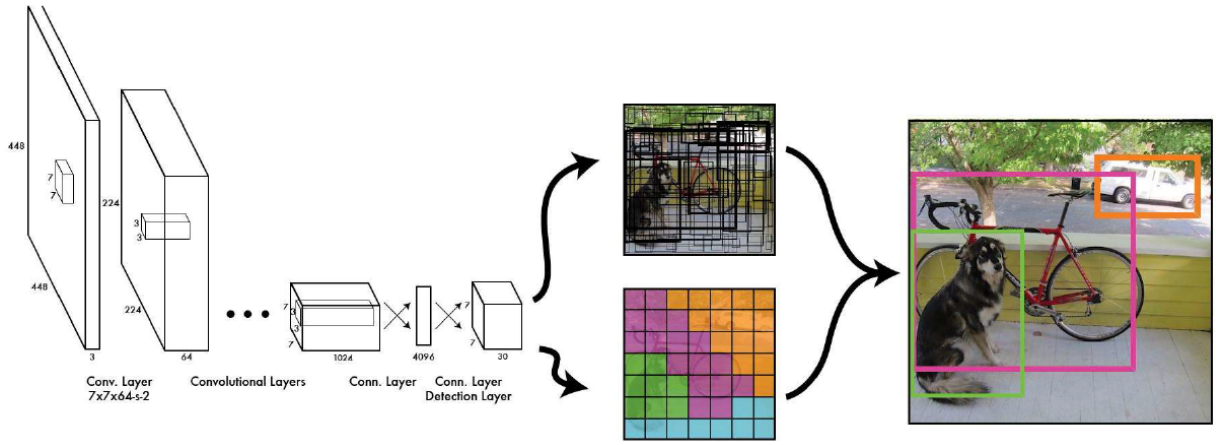

Figure 3.1: Pipeline of YOLO's algorithm [12]

You Only Look Once (YOLO) is a modern object detection algorithm developed and published in 2015 by Redmon et al. [8]. The name of the algorithm is motivated by the fact that the algorithm only looks once at the image and requires only one forward propagation pass through the neural network to make predictions unlike other state of the art object detection algorithms which work with region proposals and look at the image multiple times. YOLO uses a single end-to-end convolutional neural network which processes RGB images of size 448 x 448 and outputs the bounding box predictions for the given image (3.1). It basically reframes object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities [13]. The algorithm divides the input image into an S x S grid (in the paper S = 7). For each grid cell it predicts B bounding boxes (in the paper B = 2), where each bounding box consists of 4 coordinates and a confidence score for the prediction, and C class probabilities per grid cell taking the highest one as the final class. All of these predictions are encoded as an S x S x (B * 5 + C) tensor which is being outputted by the neural network (3.2). What the algorithm finally does, is identifying objects in the image and mapping them to the grid cell containing the center of the object. This grid cell will be responsible for predicting the final bounding box of the object and will have the highest confidence score.

In the example (3.2) each cell of the 7 x 7 grid is represented by a vector of size 30 representing a particular area of the image. Each vector contains 2 bounding box predictions (5 values each) and 20 conditional class probabilities P(class|object). The first step upon extracting a valid prediction is to choose the bounding box with the higher confidence score and check if the confidence score is above a predefined threshold (threshold = 0.25 in the paper) to output it as a valid prediction. This confidence score represents the prior in the conditional probability for the class prediction stating the probability that the given grid cell is the center of an object with a correct bounding box. To extract the class prediction YOLO outputs the conditional probability with the highest score. YOLO spatially defines each bounding box by four coordinates (X, Y, Width, Height), where (X, Y) represent the center of the bounding box relative to the cell, while (Width, Height) represent the width and the height of the bounding box relative to the whole image. Because of this, a bounding box can be bigger than the cell where it was predicted. The cell is only used as the anchor point for the prediction. One

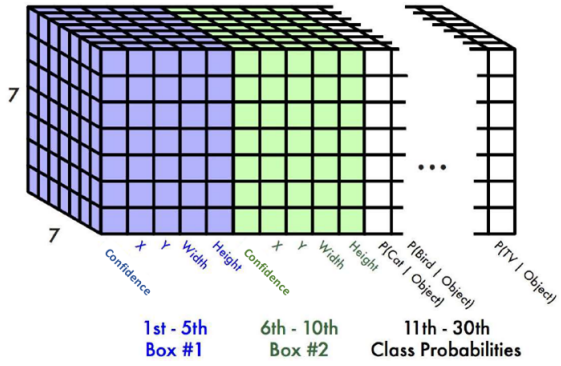

Figure 3.2: Output tensor of YOLO [12]

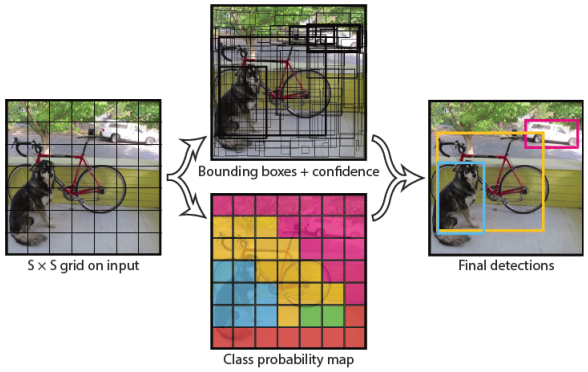

Figure 3.3: SxS grid and final detections [12]

disadvantage of this approach is the fact that every cell is able to predict only one object. If multiple objects are having their center points in the same cell, only one will be predicted.

3.1.1 Model Architecture

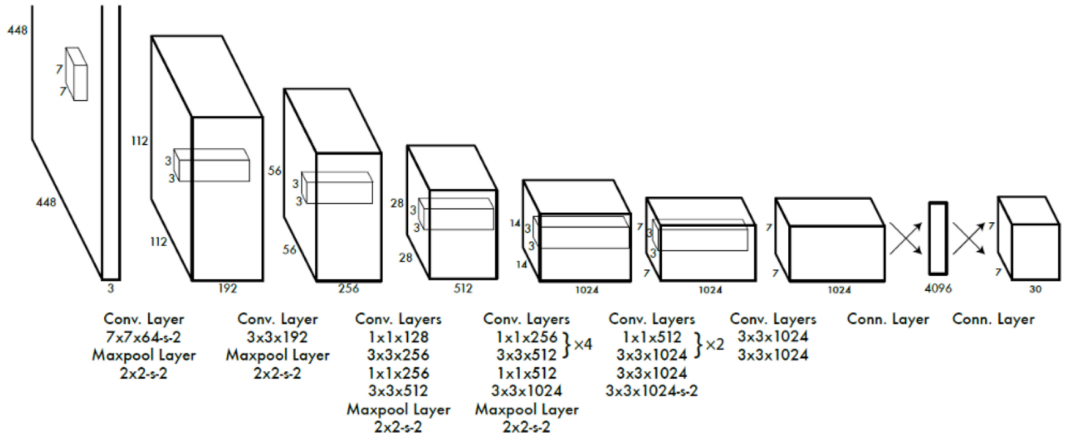

Figure 3.4: YOLO's model architecture [12]

The model architecture consists of 24 convolutional layers followed by 4 pooling layers and 2 fully connected layers ((3.4). It uses 1 x 1 convolutions to reduce the amount of feature maps which is motivated by the Inception Modules of GoogLeNet [14]. Furthermore it applies the Leaky ReLu activation function after all layers except for the last one and uses dropout between the two fully connected layers in order to tackle overfitting.

3.1.2 Loss Function

The YOLO algorithm uses a custom loss function in order to control the different output domains and their influence on the final loss by using special hyperparameters ((3.5):

1. first term: penalizes bad locations for the center coordinates if the cell contains an object.
2. second term: penalizes bad bounding box width and height values. The square root is present so that errors in small bounding boxes are more penalizing than errors in big bounding boxes.
3. third term: penalizes small confidence scores for cells containing an object.
4. fourth term: penalizes big confidence scores for cells containing no object.
5. fifth term: simple squared classification loss.

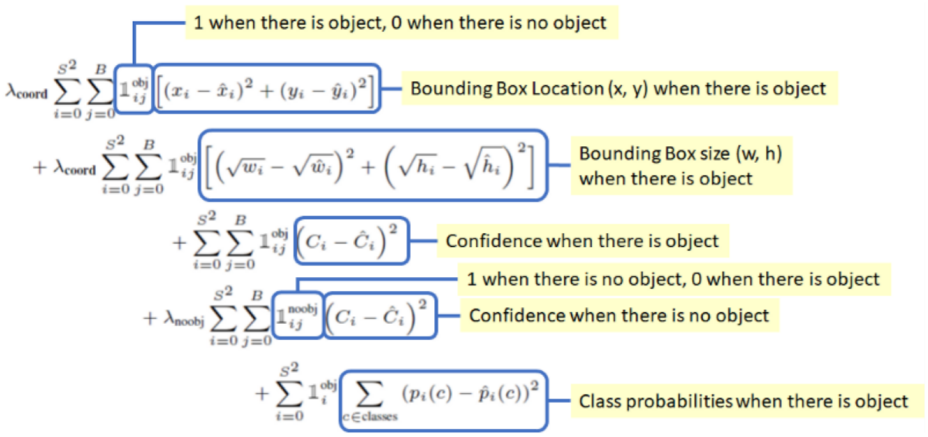

Figure 3.5: YOLO's custom loss function for every grid cell [12]

3.1.3 Training YOLO on BDD100K

We implemented and trained YOLO completely from scratch by using only the BDD100K dataset. Since there are numerous objects inside the images of the BDD100K dataset, we decided to increase the split size for the YOLO algorithm from 7 to 14 in order to mitigate the problem of multiple object centres falling into one cell (3.6). By deploying a much finer grid, we increased the amount of output parameters from 1127 to 4508 as well as the amount of parameters inside the last 5 layers. To achieve this, we adjusted the stride of the 23th convolutional layer from 2 to 1 to retain the output size of 14 x 14.

Furthermore, we added batch normalization between all layers to increase the train speed and retained the original loss hyperparameters from the paper during training (coord=5 and noobj=0.5). Finally we trained YOLO for 100 epochs with a learning rate of 1e-5 and batch size of 10. Our YOLO algorithm produces 2 bounding box predictions per grid cell on a 14 x 14 grid. The input image has dimension (3, 448, 448) and the algorithm produces a tensor of size (14, 14, 23) as output.

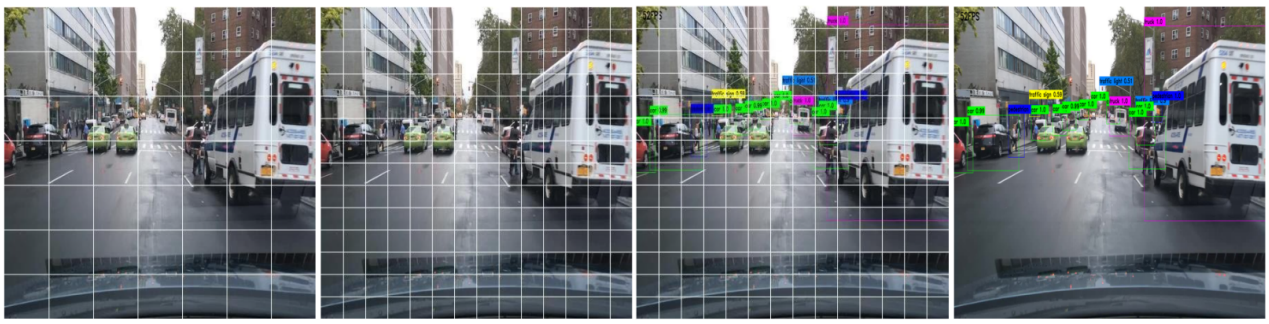

Figure 3.6: YOLO with split size 14 on the BDD100K dataset

3.2 Faster R-CNN

The architecture of faster R-CNN [7] consists of three parts: the network backbone, the region proposal network (RPN) and the older version of this algorithm called fast R-CNN (3.7). The network backbone is in general a classification network like VGG-Net or ResNet pretrained on an image classification dataset. It is used to generate high resolution feature maps and requires an image size of 640 x 640 pixels. In our approach we used ResNet50 as the backbone network pretrained on the ImageNet dataset.

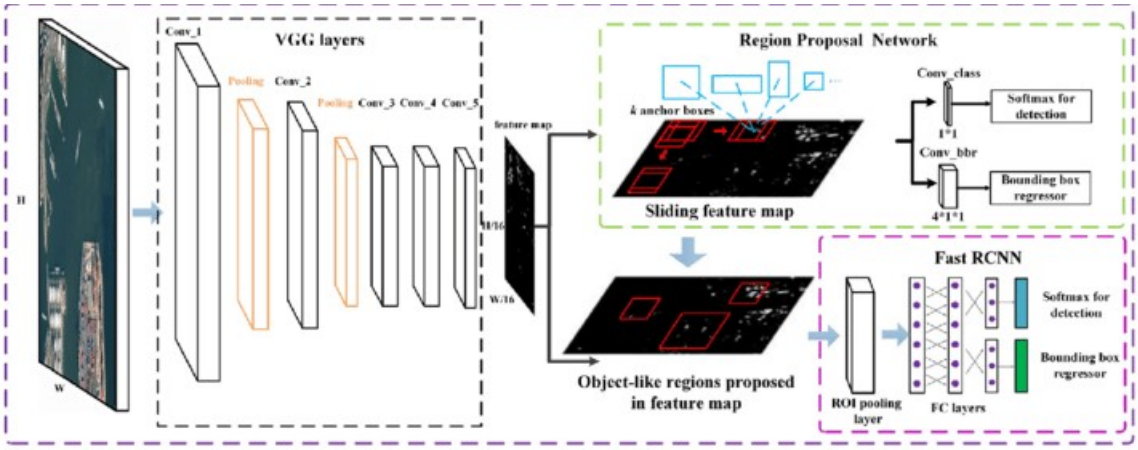

Figure 3.7: Faster R-CNN model architecture [15]

3.2.1 Region Proposal Network

The region proposal network consists of a single convolutional layer which is then being divided into 2 separate convolutional layers to predict a classification score and bounding boxes for the region proposals. The network uses predefined anchor boxes to generate approximately 2.000 region proposals.

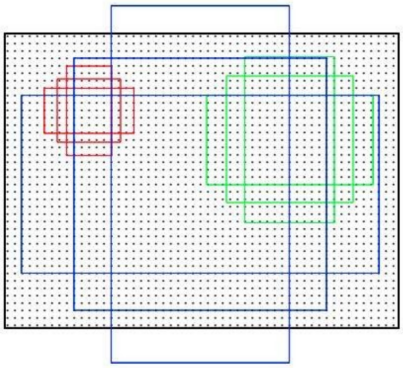

Figure 3.8: Anchor boxes with the sizes $\{128, 256, 512\}$ with aspect ratios of $\{0.5, 1, 2\}$ [16]

An anchor is a smaller part of an image. In Our approach we are using anchors with the sizes $\{128, 256, 512\}$ with the aspect ratios of $\{0.5, 1, 2\}$ which leads to 9 different anchor boxes (3.8). Each of these anchors will be slided over the window with a stride of 16 and cuts out the overlapping parts of the image. The classification score decides for all anchors if they include an object or not. The bounding box regressions are for better identifying the objects in the anchors. After the prediction the number of region proposals will be reduced, with non maximum suppression.

3.2.2 Fast R-CNN

Fast R-CNN, the last part of the architecture gets the high resolution feature maps from the network backbone and the region proposals from the region proposal network as input. To get a fixed size from the different sized region proposals, a method called ROI pooling (3.9) is used, which stands for region of interest pooling.

For every region of interest from the input list, it takes a section of the input feature map that corresponds to it and scales it to some predefined size, in our case 7x7. The scaling is done by:

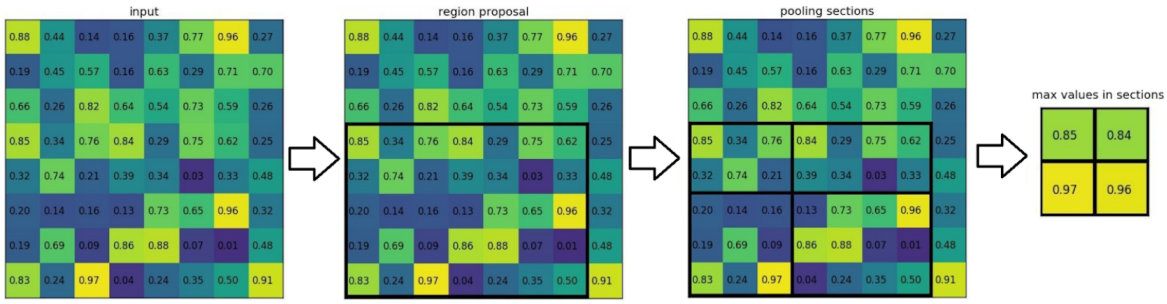

Figure 3.9: Schematic representation of ROI pooling [17]

1. Dividing the region proposal into equal sized sections, where the number of sections is the same as the dimension of the output
2. Finding the largest value in each section
3. Copying these max values to the output buffer

After that there are only 2 fully connected layers followed by two separate output layers which predict the softmax score for every class and the corrected bounding boxes for every region proposal. As loss functions for the regression they use log loss and for the bounding boxes they use smooth L1 Loss and backpropagate the losses together by factors which determine how much the bounding box and the classification loss should affect the entire loss.

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{x,y,w,h,\}} \text{smooth}_{L1}(t_i^u - v_i),$$

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

4 Experimental Results

4.1 Training

YOLO was trained for 81 Epochs with a decreasing learning rate of 1e-5 and batchsize 10. Faster R-CNN was trained for 60 epochs with a decreasing learning rate of 1e-4 and batchsize 16. The normalized total loss and the mAP of the training progress is shown in (4.1).

4.2 Results

The implementation of the models and the processed data are found in this GitHub repository. Furthermore, we provide videos of real-time object detection with our models: Faster R-CNN and YOLO.

We measured the FPS and mAP for the evaluation and comparison of YOLO and faster R-CNN on a NVIDIA V100 SXM2 32GB (4.1).

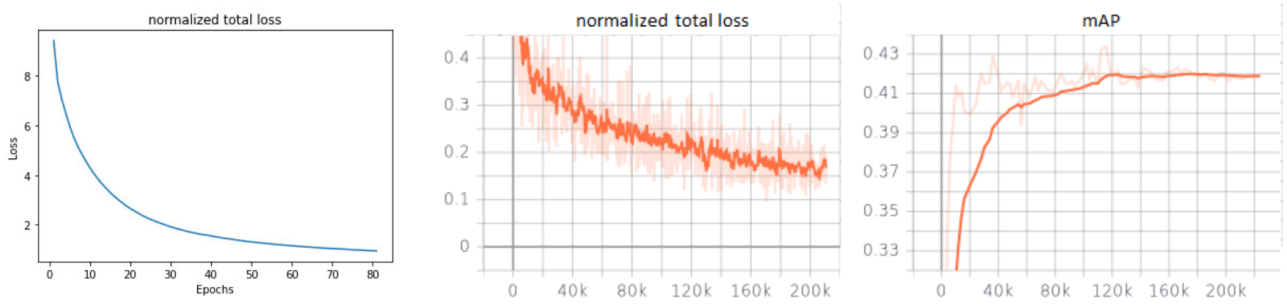

Figure 4.1: Normalized total loss for YOLO (left), normalized total loss for Faster R-CNN (middle) and mean average precision for Faster R-CNN (right)

Table 4.1: Comparison between YOLO, Faster R-CNN and Hybrid incremental net in mAP and FPS on the BDD100K dataset

	mAP	FPS
YOLO	18,6	212
Faster R-CNN	41,8	17,1
Hybrid incremental net	45,7	N/A

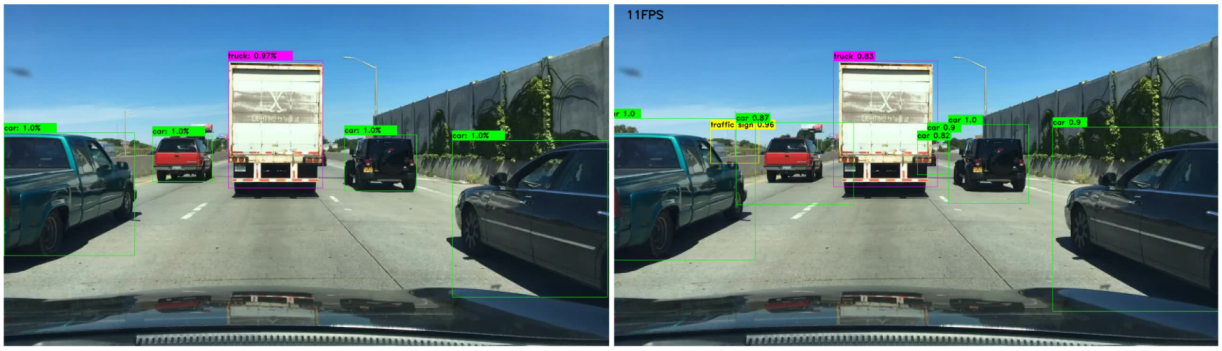

Figure 4.2: Results on BDD100K: Faster R-CNN (left) and YOLO (right). More results from the YOLO and the Faster R-CNN architecture are shown in the appendix.

5 Conclusion and Future Work

In our project we implemented and trained a one-stage detector YOLO and a two-stage detector Faster R-CNN on the BDD100K dataset in the context of autonomous driving. As expected, the results of the evaluation showed that Faster R-CNN has a higher accuracy but lower FPS. In comparison YOLO has a much higher FPS, but also much lower accuracy because of its simple architecture. Future work includes further experiments with newer models, for example the newer versions of YOLO, since we used the first version of YOLO in this project. Future work in the long term would be reaching performances with high accuracy and high FPS which are suitable for the goal of autonomous driving.

Bibliography

[1] Gene Lewis. Object detection for autonomous vehicles, 2014.

[2] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[3] Jason Brownlee. A gentle introduction to object recognition with deep learning. *Machine Learning Mastery*, 5, 2019.

[4] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645, 2020.

[5] Prajjwal Bhargava. On generalizing detection models for unconstrained environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[6] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[9] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[10] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[11] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[12] Sik-Ho Tsang. Review: Yolov1 — you only look once (object detection). `https://towardsdatascience.com/yolov1-you-only-look-once-object-detection-e1f3ffec8a89` zuletzt besucht: 15.03.21, 2018.

[13] Manish Chablani. Yolo – you only look once, real time object detection explained. `https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006` zuletzt besucht: 15.03.21, 2017.

[14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[15] Shilin Zhou Juanping Zhao Zhipeng Deng, Hao Sun. Multi-scale object detection in remote sensing imagery with convolutional neural networks. In *ISPRS Journal of Photogrammetry and Remote Sensing 145*, 2018.

[16] Mata. faster rcnn in rpn the anchor, sliding windows, proposals of understanding. `https://www.programmersought.com/article/31012543832/` zuletzt besucht: 15.03.21.

[17] Tomasz Grel. Region of interest pooling explained. `https://deepsense.ai/region-of-interest-pooling-explained/` zuletzt besucht: 15.03.21, 2017.

Appendices

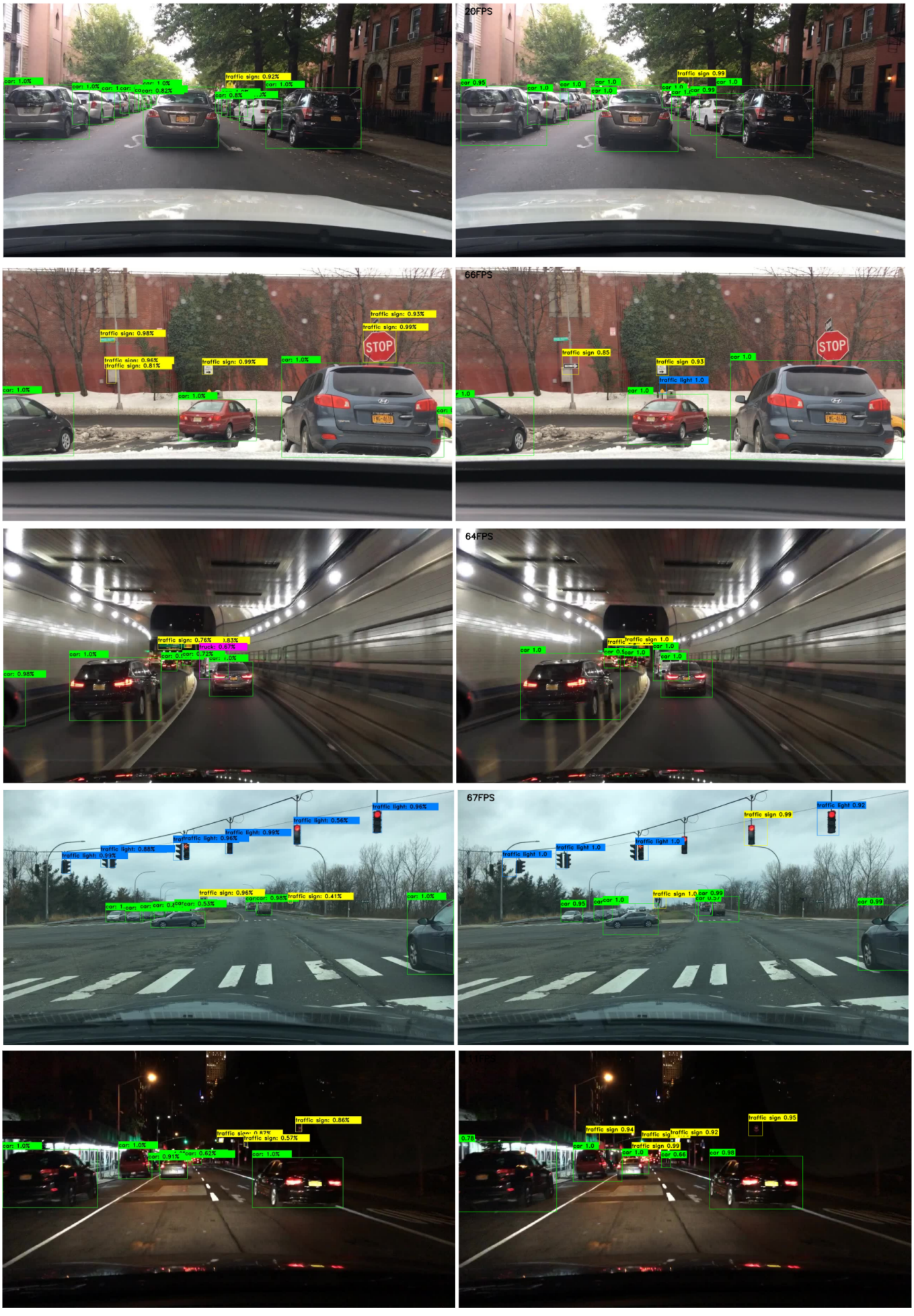

Figure .1: Results on BDD100K: Faster R-CNN (left) and YOLO (right)