Analyzing Traffic Streams Using CNNs

Lennart Faber (s2500523)[a], Rohit Malhotra(s3801128) [a],

Niels Broekmans (s2589036)[a], Finn Gaida (s3838730)[b]

[a] *University of Groningen*
[b] *Technical University Munich*

Abstract

Inspired by a challenge from the Gemeente Groningen, this project aims to find an algorithm that is capable of detecting the number of cyclists and pedestrians in a (CCTV) video stream. This could give us more insight into traffic streams and possible implementations of dynamic traffic control systems. As with the ongoing rapid development in deep learning, convolutional neural networks are the state of the art in object detection. In this paper we have compared different CNN models, these models differ in their architectures and their training strategies.

We compared different setups, including single shot detectors (YOLOv3 [13], Focal loss based (RetinaNet) [9]) and Region Proposal Network based (Faster R-CNN [14]).
Keywords: CNN, object detection, bicycle detection, deep learning, Faster R-CNN [14], YOLOV3 [13], RetinaNet [9].

1 Introduction

In order to control the traffic flow in a city in an efficient way, it is important to have knowledge about the amount of road users at a given point in time. Because the streets are covered by cameras in most big cities, the remaining problem is to localize and classify the users of the road in a certain video frame. In formal terms this is a problem of object detection. For this project, we will focus on cyclists and pedestrians and compare different deep learning algorithms that could possibly solve this task.

Traditionally, object detection is done in three stages: region selection, feature extraction and classification. **Region selection** is the task of scanning the whole image to find the area where objects of different sizes might reside. While scanning the image, windows of all possible sizes have to be considered, which could be computationally expensive. In order to recognize different objects, one has to extract features from an image that could define the semantics of different objects (**feature extraction**). These features could then be used to recognize various objects. For this task, various techniques like SVM [6] and ensemble methods could be applied, but with the development of deep neural networks that don't need manual explicit feature extraction, the field of object detection has seen great progression.

In this paper we compare two types of detectors: region based and single shot object detectors. In region based methods, regions of interest (**RoI**) are generated as starting point. RoIs are different sized bounding boxes which have a high likelihood of containing objects of interest. Next the RoI is classified to an object category after which a refinement of bounding boxes will take place.

For our experiments we have used Faster R-CNN [14], which uses a CNN (RPN) to generate region of interest. We have tried different types of backbones (Vgg16 [15], ResNet50 [5]), which are CNNs for feature extraction. Fully connected layers are used to classify and do bounding box regression.

In region based detectors, region detection and feature extraction are done by different networks. But in single shot detectors, object detection is done in a single pass through the network. This makes them faster than region based networks, but reduces accuracy. There are techniques like FPN (Feature Pyramid Networks [8]) to increase their accuracy. In this category of detectors, we have compared YOLOV3 [13] and focal loss based RetinaNet [9], with different feature extractors (backbones). We talk about these detectors in more detail in further sections.

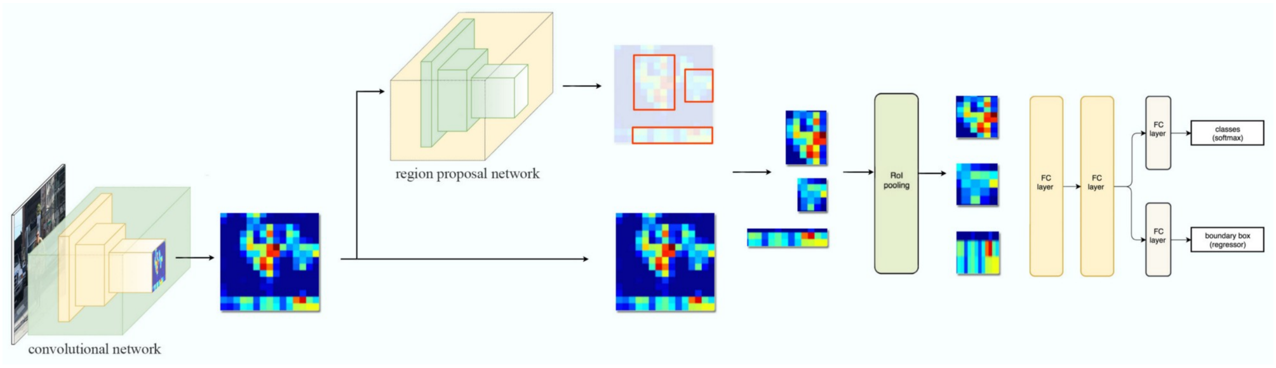

Figure 1: Faster R-CNN Architecture, where external region proposal is replaced by internal deep network. (From [7])

2 Dataset

For training we rely on the MIO-TCD dataset [11], published in 2018 which contains 137,743 photos of traffic scenes with localized cars, trucks, and other traffic participants but most importantly for us: bicycles and pedestrians. There were 4452 images containing 8466 pedestrians + cyclists.

For evaluation of our trained models we tested against the KITTI dataset [2] from 2013, to test different image sizes and because there were more cyclists and pedestrians in this dataset. It contains 15,000 images recorded with a vehicle similar to the Google StreetView car. We tested on 2486 images containing 4487 pedestrians and 1627 cyclists (we only used images in which at least one pedestrian or cyclist was present).

3 Background

Since the proposal of R-CNN [4], a great amount of improvement has been made in order to increase accuracy and decrease inference timing. In Fast R-CNN [3] by not repeating the feature extraction on each proposed region we decrease the processing time significantly. Faster R-CNN is further improved by replacing external region proposal techniques with an internal network. YOLO (v3) [13] is a single shot detector which was improved over previous versions by using FPN and changing the cost function. All of them bring different degrees of detection performance improvements and make real-time and accurate object detection become more achievable.

3.1 Faster R-CNN

Faster R-CNN is an integrated method. The main idea is to use shared convolutional layers for region proposal generation and detection. It was discovered that feature maps generated by object detection networks can also be used to generate the region proposals. The fully convolutional part of the Faster R-CNN network that generates the feature proposals is called a region proposal network (RPN). The network flow is similar to that of the Fast R-CNN architecture, but region proposal was replaced by a convolutional network.

A Faster R-CNN network is trained by alternating between training for RoI generation and detection. First, two separate networks are trained. Then, these networks are combined and fine-tuned. During fine-tuning, certain layers are kept fixed and certain layers are trained in turn. The trained network receives a single image as input. The shared fully convolutional layers generate feature maps from the image. These feature maps are fed to the RPN. The RPN outputs region proposals, which are input, together with the said feature maps, to the final detection layers. These layers include a RoI pooling layer and output the final classifications.

Using shared convolutional layers, region proposals are computationally cost effective. Earlier region proposal part was only executed on CPU, but for computing the region proposals on using CNN, GPU can be used. The architecture of Faster R-CNN is shown in Figure 1.

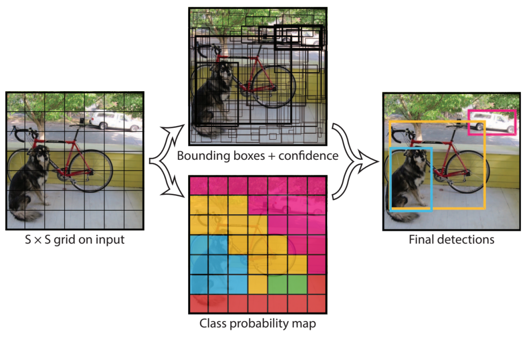

Figure 2: Basic idea of YOLO (From [12])

3.2 YOLOv3

In Faster R-CNN, time spent on handling both components (RPN and detection) becomes a bottleneck in real time application. As we told earlier, one stage or single shot detectors which map directly from pixels to bounding box boundaries and class probabilities reduce this time expense. You Only Look Once (YOLO) [12], treats object detection as a classification task. The basic idea of YOLO is shown in Figure 2. YOLO divides the image into a grid and each grid cell is responsible for predicting an object centered at that cell. Each grid cell predicts bounding boxes and each bounding box has a confidence score. However, this one object per grid cell rule limited the detection of closer objects. Similarly it performed poorly when the size of the object was very small. In YOLOv3 it changed to a more complex backbone for feature extraction and used FPN [8] to detect small objects.

3.3 RetinaNet (Focal Loss)

As we have seen before, Single Shot Detectors trade accuracy for temporal performance. Focal loss aims to find a balance between inference speed and accuracy by specifically reducing the loss for well-trained classes.
The initial authors [9] of the Focal Loss took the ordinary binary Cross Entropy loss as a starting point:

$$CE(p,y) = \begin{cases} -log(p), & \text{if } y = 1 \\ -log(1-p). & \text{otherwise} \end{cases}$$

Here $y \in \{-1, 1\}$ symbolizes the class of the classified object (1 for correctly classified) and $p \in [0, 1]$ the probability of that class.
Now in order to deal with the strong class imbalance, a slight variation of the Cross Entropy loss with a weighting factor has been introduced:

$$CE(p) = -\alpha log(p)$$

(For readability p has been redefined as $1 - p$ in case $y = -1$)
Here the factor α can be set to, for example, the inverse of the number of times a class appears in the dataset, but in some cases this would still not suffice.
Therefore, the authors created a new loss function called Focal Loss:

$$FL(p) = -\alpha(1-p)^{\gamma} log(p)$$

It's called "Focal loss" because the additional factor $(1 - p)^{\gamma}$ (with "focusing parameter" γ) increases the loss for hard training examples (meaning low p) and decreases the loss for easy or well-trained class examples with high p. In other words, misclassified examples get a higher priority of being corrected while well-classified examples are de-prioritized.
The authors have experimented with different values for α and γ and come to find that $\alpha = 0.25$ and $\gamma = 2$ yield the best results in most cases.

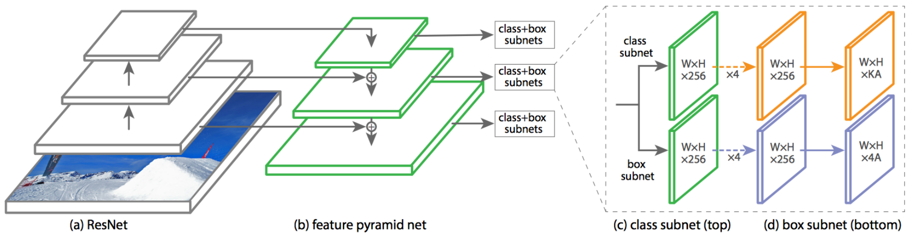

Figure 3: RetinaNet Architecture

Detector	Backbone	Loss Function	Optimizer	Learning Rate	Platform	Language
Faster R-CNN	Vgg16	Classifier: Cross Entropy Regression: Smooth L1	Adam	0.00005	Keras/Tensorflow	Python
Faster R-CNN	ResNet50	Classifier: Cross Entropy Regression: Smooth L1	Adam	0.00005	Keras/Tensorflow	Python
RetinaNet	Vgg16	Regression: Smooth L1 Classifier: Focal	Adam	0.00005	Keras/Tensorflow	Python
RetinaNet	ResNet50	Regression: Smooth L1 Classifier: Focal	Adam	0.00005	Keras/Tensorflow	Python
RetinaNet	ResNet101	Regression: Smooth L1 Classifier: Focal	Adam	0.00005	Keras/Tensorflow	Python
YOLOv3 (Darknet)(pretrained)	DarkNet53	Custom YOLO	Adam	0.001	Darknet	C

Table 1: Training details for different detectors.

The architecture of RetinaNet takes a Feature Pyramid Network as a backbone (built on top of ResNet50 or Resnet 101 in our case) and attaches two task-bound subnets to it: One for classification and one for bounding box regression (See 3. This has the advantage that objects of all sizes are approximately equally probable to be classified correctly, whereas Single Shot Detectors such as YOLOv3 struggle with smaller objects).

4 Experimental setup

A major problem with training of deep learning models is the availability of high computing power. To tackle this, we ran the training and evaluation code on the Peregrine cluster [1] provided by the RUG. The CPU and GPU nodes of this cluster are equipped with Intel Xeon 2.5 GHz cores and Nvidia K40 GPU cards respectively.

In addition to computing power, deep convolutional networks require a large amount of training data. Since our task was to only detect bicyclist and pedestrian from a given image, we only trained our models on these two classes.

We trained different combinations of feature extractors with our detectors. For training we used pre-trained weights supplied with these backbones. For Faster R-CNN [14] we used VGG-16 [15] and ResNet50 [5] as backbones. For RetinaNet [9] we used VGG-16 [15], ResNet50 [5] and ResNet101 [5] as backbones. The pretrained weights were trained on the COCO [10] dataset. Further details of the experimental setup are given in Table 1.

4.1 Difficulties regarding YOLOv3

We did not manage to train YOLOv3 using either its standard DarkNet implementation nor using a conversion to a Keras implementation due to many crashes that were encountered only after days of training time. Therefore, we predicted the objects in our test data based on the original YOLOv3weights. One problem with this was that the YOLOv3weights are based on a slightly different categorization. Where the KITTI and MIO-TCD dataset contain labels for cyclists and pedestrians, YOLOv3 recognizes bicycles and persons. We solved this problem by looking at the coordinates of the persons and the bicycles and converting those to cyclists when the person was on top of the bicycle. We decided that this would be the case when their predicted boxes overlap. (See Figure 4).

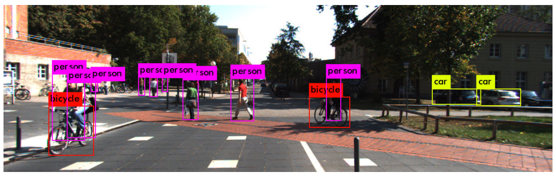

Figure 4: Darknet Yolo prediction

5 Results

Intersection over union (IoU) [16] is a measure often used in object detection: it is the intersection of the ground-truth box and the predicted box divided by the union of the ground-truth box and the predicted box. High overlap results in a high IoU. Generally, an IoU threshold of 0.5 is used for true positive measure. There is only one true positive match for each ground-truth. If several detections match the same ground truth, the box with the highest likelihood is selected as the true positive match and the other detections are marked as false positives. Detections which match with no ground truth box are marked false positives. The ground truth boxes with no matching detections are called false negatives.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

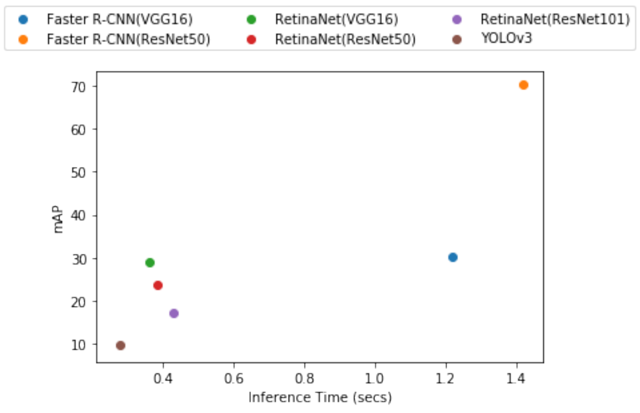

Figure 5: Speed Vs Accuracy Tradeoff. Note that for Faster R-CNN accuracy is highest and for YOLOv3 speed is best but accuracy is poor. For Faster R-CNN(Vgg16) RetinaNet(Vgg16) accuracy scores are similar, but inference time is significantly different. (Note: Time is measured on GPU nodes, except for YOLOv3)

The performance of the object detection algorithm is evaluated by calculating the precision-recall curve and the interpolated average precision of the detections for each class. For each image, the average precision of each class is the average of the area under the precision recall curve. We rank the detection in order of their predicted likelihood and then calculate precision recall value pairs in the same order. If we would plot these points and take the total area under this plot and average it by total number of samples, this gives us the average precision (AP) of each class per image. Then we calculate the mean average precision (mAP) per image, which is the average over all APs. Total mAP of the model is the mean of all mAP values. Table 2 shows mAP values for different detectors with different backbones.

As shown in Figure 5 single shot detectors have better inference timing but region based detectors have better accuracy. Some images of example predictions are shown in Figure 6.

Detector	IoU = 0.4			IoU = 0.5			IoU = 0.6		
	AP pedestrian	AP cyclist	mAP	AP pedestrian	AP cyclist	mAP	AP pedestrian	AP cyclist	mAP
FRCNN-resnet50	**65.29**%	**88.05**%	**76.72**%	**61.71**%	**78.67**%	**70.21**%	**53.86**%	**78.19**%	**66.03**%
FRCNN-vgg16	24.38%	41.28%	32.90%	22.92%	38.02%	30.42%	20.07%	39.34%	29.73%
Retinanet-resnet50	34.91%	17.68%	30.32%	27.92%	11.83%	23.64%	5.12%	17.13%	13.93%
Retinanet-vgg16	41.14%	17.35%	34.81%	34.63%	13.94%	29.13%	23.65%	5.82%	18.91%
Retinanet-resnet101	24.94%	11.56%	21.38%	20.53%	8.74%	17.39%	13.54%	3.8%	10.97%
YOLOv3 (DarkNet)	20.05%	0.53%	10.29%	18.67%	0.03%	9.35%	16.94%	0.00%	8.47%
YOLOv3 (Keras)	21.21%	1.13%	11.17%	19.18%	0.26%	9.72%	16.66%	0.00%	8.33%

Table 2: mAP results of different architecture combination on our training data.

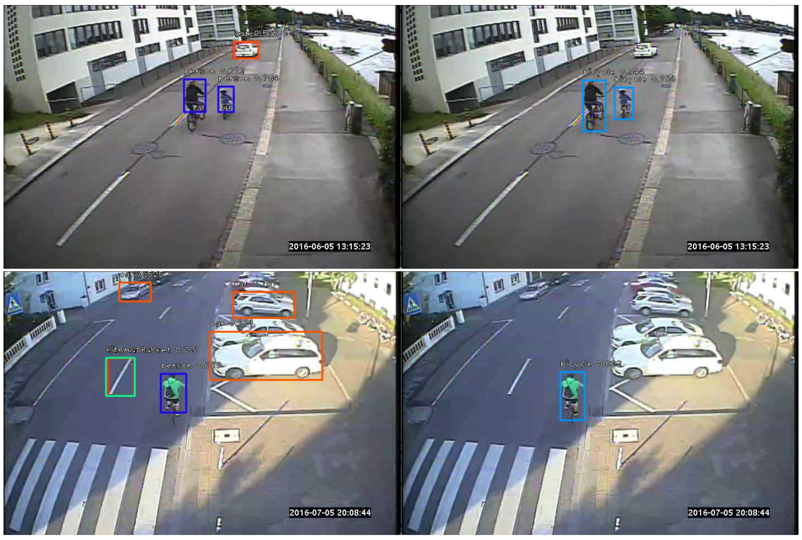

Figure 6: Pretrained weights (left) vs. MIO-TCD re-trained weights (right)

6 Discussion

We found that training only the upper layers of pre-trained models with freezed backbones not only saves training time, but can also result in better precision overall.

We have found that region based detectors have performed well on the bicycle class compared to pedestrian while single shot detectors have performed well on the pedestrian class. While FRCNN-resnet50 proved to be the very best when looking at its accuracy score, the YOLOv3 Keras implementation seems the most promising when taking detection speed into account. Running it on GPU could not be achieved due to crashes and queuing problems on the computer cluster, but we were positively surprised by its performance on CPU. We also believe that the accuracy scores of this implementation could have been improved significantly when properly trained.

Finally, we believe that we could have done better on selecting data for training and testing. The MIO-TCD data set was very different from the KITTI dataset, with only few objects per image while the KITTI images contained many. If our implementations would have been trained and tested on more similar data sets, accuracy scores would have been likely to be improved significantly.

References

[1] RUG Center for Information Technology. Peregrine hpc cluster. https://www.rug.nl/society-business/centre-for-information-technology/research/services/hpc/facilities/peregrine-hpc-cluster.

[2] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[3] Ross Girshick. Fast r-cnn. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1440–1448, Washington, DC, USA, 2015. IEEE Computer Society.

[4] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] Marti A. Hearst. Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28, July 1998.

[7] Jonathan Hui. What do we learn from region based object detectors (faster r-cnn, r-fcn, fpn)?, 2013. [Online; accessed Jan 23, 2019].

[8] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[9] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[10] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[11] Zhiming Luo, B Frederic, Carl Lemaire, Janusz Konrad, Shaozi Li, Akshaya Mishra, Andrew Achkar, Justin Eichel, Pierre-Marc Jodoin, et al. Mio-tcd: A new benchmark dataset for vehicle classification and localization. *IEEE Transactions on Image Processing*, 2018.

[12] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[13] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 91–99, Cambridge, MA, USA, 2015. MIT Press.

[15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[16] C. Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV 2014*, 2014.