NHL STENDEN
computer vision & data science

A Comparison of Deep Learning Methods for Two-wheeler Detection in Images

NHL Stenden Professorship Computer Vision & Data Science

Michel Jelsma, Douwe Pieter Reitsma

Supervisors: Klaas Dijkstra, Lucas Ramos

Abstract— Traffic analysis is currently a laborious task performed manually by human experts. Object detection models could potentially speed up this process significantly. This research explores which object detection models are most suitable for detecting vulnerable road users, such as bicycles, scooters, and pedestrians, in traffic camera footage. In this work, we compared Faster-RCNN, YOLOv5, an ensemble of YOLOv5 models, and a YOLOv5 model trained on the ensemble's output (surrogate model) to see which approach resulted in the best accuracy. The models were trained and validated on the Specialized Cyclist, Berkley Deep Drive 100K, and the MB10000 datasets. Additionally, we created a custom dataset used for fine-tuning and testing. This dataset contains images from a traffic intersection in the province of Utrecht.

Three experiments were conducted to examine which object detection models are most suitable for traffic analysis. Experiment 1 compared Faster-RCNN to YOLOv5s and YOLOv5m to assess the most accurate model. In experiment 2, we analyzed the differences between using and not using pre-trained weights before training. Finally, in experiment 3, a fine-tuned YOLOv5 model was compared to the fine-tuned ensemble and surrogate models. Faster-RCNN had the best accuracy in the first experiment with 0.88 mAP, 0.81 F1-score, 0.77 precision, and 0.85 recall. Experiment 2 demonstrated that pre-training on the external datasets resulted in better scooter and pedestrian detection accuracy but worse performance for cyclist detection. Finally, the third experiment revealed that a fine-tuned model pre-trained on the combined external datasets was more accurate than the ensemble and surrogate models. We concluded that Faster-RCNN and YOLOv5 trained only on our dataset could detect vulnerable road users with high accuracy. Furthermore, it was not beneficial to use ensemble and surrogate models over a model trained on combined external datasets in our case.

Index Terms—Deep learning, Object Detection, Surrogate Learning, Ensemble Model, YOLOv5, Accident detection, Two-wheeler, Cyclist, Scooter, Pedestrian.

1 Introduction

Two-wheelers such as (electric) bicycles and scooters are a vulnerable group on the road, as they are often victims of road accidents. It is estimated that in the Netherlands, over 10000 bicyclists are hospitalized with severe injuries annually (van der Horst et al., 2014). According to Reurings et al. (2012), a large number of factors such as cyclist behavior, traffic environment, choice of route, and weather conditions, can contribute to unsafe traffic situations for cyclists. Therefore, identifying these factors could help locate dangerous traffic points.

Human experts can identify these dangerous traffic points manually by physically standing near a road, keeping track of traffic, or watching video data collected by traffic cameras. The process described above is cumbersome since it is time demanding and prone to human error. The people looking at the traffic videos cannot predict when a dangerous circumstance will occur, which means they have to pick specific periods, which often results in the loss of valuable information.

Computer Vision could be of great aid to identify possibly dangerous circumstances. For instance, a model could be trained to detect such circumstances and report the corresponding timestamp to the user, significantly speeding up the expert's work, who often has to watch days of footage to find the relevant events.

- *Michel Jelsma is an Electrical Engineering student at the NHL Stenden University of Applied Sciences, E-mail: michel.jelsma@student.nhlstenden.com.*
- *Douwe Pieter Reitsma is a Software Engineering student at the Hanze University of Applied Sciences, E-mail: d.p.reitsma@st.hanze.nl.*
- *Klaas Dijkstra is an associate lector at the NHL Stenden Lectoraat in Computer Vision & Data Science, E-mail: klaas.dijkstra@nhlstenden.com.*
- *Lucas Ramos is a researcher at the NHL Stenden Lectoraat in Computer Vision & Data Science, E-mail: lucas.ramos@nhlstenden.com.*

The first step for detecting dangerous situations automatically is identifying all elements involved correctly. Therefore, this paper examines which detection models are most suitable for detecting two-wheeled vehicles in traffic camera footage. Furthermore, we also describe a system for detecting the status of traffic lights in semi-fixed positions (the traffic light can move a few pixels between frames; see Appendix A for a more detailed description).

The primary research question of this paper is: *To what degree of accuracy can two-wheelers be detected in traffic camera footage?*.

The secondary research questions of this paper are:

- *Which object detection model has the best accuracy?*
- *How does an ensemble of models compare to a single all-encompassing model?*
- *How does a surrogate model compare to an ensemble of models?*

2 State of the art

In the state-of-the-art literature, many approaches have been proposed to address the problem of two-wheeled vehicle detection. Traditionally, Sayed et al. (2013) proposed an automated safety diagnosis approach for evaluating traffic conflicts involving vehicles and bicycles. In the paper, computer vision techniques, such as feature extraction, were used to analyze videos automatically and detect traffic conflicts. These conflicts were then ranked by severity using the time-to-collision (TTC) indicator, which refers to the time remaining before an accident when the course and speed of the vehicles involved are maintained (Naseralavi et al., 2013).

Foroozandeh Shahraki (2017) described the use of a convolutional neural network (CNN) to implement a detector called YOLOv2 (Redmon & Farhadi, 2016) that can be used to detect bicycles in footage from traffic cameras. A few different datasets were mentioned in their work, but the model was primarily trained using

the Gavrilla.net dataset (Li et al., 2016b), which contains images of cyclists and pedestrians in different camera angles. Furthermore, to prevent the model from classifying cyclists as pedestrians and vice versa, the model was also trained with images containing either only bicycles or bicycles with the lower half of the cyclist's body. Their approach showed good performance at cyclist detection and high accuracy at cyclist counting. Therefore it is suitable for detecting dangerous traffic situations since counting the number of cyclists on the road could help identifying traffic congestions. Furthermore, the authors stated that the collected bicycle trajectory data, i.e., the change in position of the same bicycle from frame a to frame b, could be used for road safety studies. For example, to see if two bicycle trajectories overlap.

Kausar et al. (2020) compared several single and two-step deep learning models for the detection of bicycles and motorcycles. The models discussed in their study were trained and tested using the Tsinghua-Daimler Cyclist Detection Benchmark (Li et al., 2016a) and the MB7500 (Espinosa et al., 2018) dataset. From these, the Tsinghua-Daimler dataset was split into three sets of easy, moderate, and hard difficulty, respectively. The study showed that a YOLOv3 single-step model based on the Darknet-53 architecture tested on the easy, moderate, and hard sets from the cyclist dataset resulted in a mean average precision of 62.9%, 46.4%, and 44.0%, respectively. Furthermore, when tested on the motorcycles dataset, the YOLOv3 model scored a mean average precision of 89.0%. Their study also showed that applying the pre-processing steps mentioned in their paper (Gaussian blur, histogram equalization, and unsharp-masking) increased the mAP by 2% & 3% for motorcycle and bicycle detection, respectively.

3 Materials and Methods

This section provides an overview of the materials and methods that were utilized in this research.

3.1 Datasets

During our research, we used several datasets to pre-train our models (see Table 1). These models were then fine-tuned using the Traffic Intersection Dataset (TID) we created ourselves (see Section 3.1.4). The test set in the TID dataset will be used for as the test set in all the experiments.

The classes used in our models were: *bicycle*, *scooter*, *pedestrian* and *two-wheeler* (other two-wheeled vehicles). Bicycles and scooters have a separate class, because these two-wheelers are most common on the bicycle path, and we want to distinguish between them. Table 2 displays the total number of boxes per class in the datasets used to pre-train our models.

Dataset	Used Classes	Train	Validation	Total
BDD100K	pedestrian	24554	3555	28109
SCD	bicycle	7205	1271	8476
MB10000	scooter	8429	1487	9916
Total		40188	6313	46501

Table 1: Distribution of labeled images in the datasets used for pre-training the models.

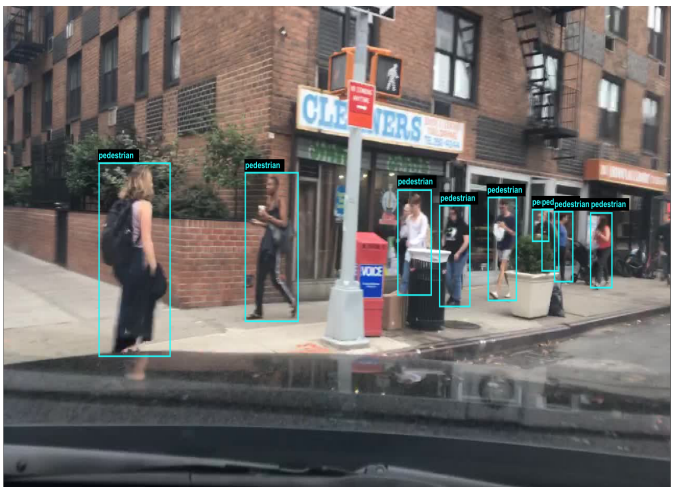
3.1.1 BDD100k

The Berkeley Deep Drive dataset (BDD100K) is a dataset containing 100000 annotated images collected from videos shot from a driving car (Yu et al., 2018). From these images, 24554 and 3555 are reserved for training and validation, respectively.

Figure 1 contains a sample image containing pedestrians from the BDD100k dataset.

Dataset	Class	Train	Validation	Total
BDD100K	pedestrian	66724	9476	76200
SCD	bicycle	7932	1361	9293
MB10000	scooter	45161	8050	53211
Total		119817	18887	138704

Table 2: Number of bounding boxes containing the corresponding class in the datasets used for pre-training the models.



Source: Berkley Deep Drive

Figure 1: Sample image from the Berkley Deep Drive 100k dataset.

3.1.2 Specialized Cyclist Dataset

Since the cyclist class is severely under-represented in the BDD100K dataset, it could prove useful to include a dataset specifically for cyclists, namely the Specialized Cyclist Dataset (SCD). The SCD is a dataset containing 62297 images of roads with approximately 18200 instances of cyclists (Masalov et al., 2019). As with the BDD100K dataset, the images from the SCD are preprocessed to contain only the relevant classes and are then split into training and validation sets of sizes 7205 and 1271, respectively.

Figure 2 demonstrates a sample image with cyclists from the Specialized Cyclist Dataset.

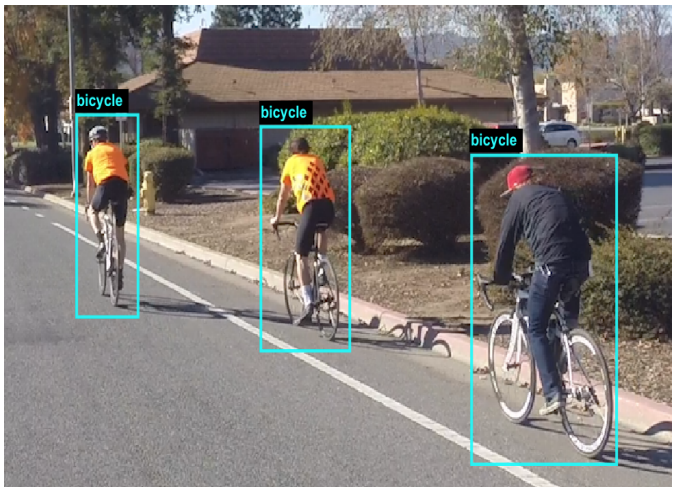
3.1.3 MB10000

Since the BDD100K dataset did not have any classes specifically for scooters, the MB10000 dataset was added to the list of datasets. The MB10000 dataset contains 10000 images of motorcycles in traffic which were shot from a drone with a camera (Espinosa et al., 2018). It is used as input for the model to increase the accuracy of motorcycle (scooter) detection. The images with the relevant classes are split into training and validation sets of sizes 8429 and 1487, respectively.

Figure 3 demonstrates a sample image with motorcycles from the MB10000 dataset.

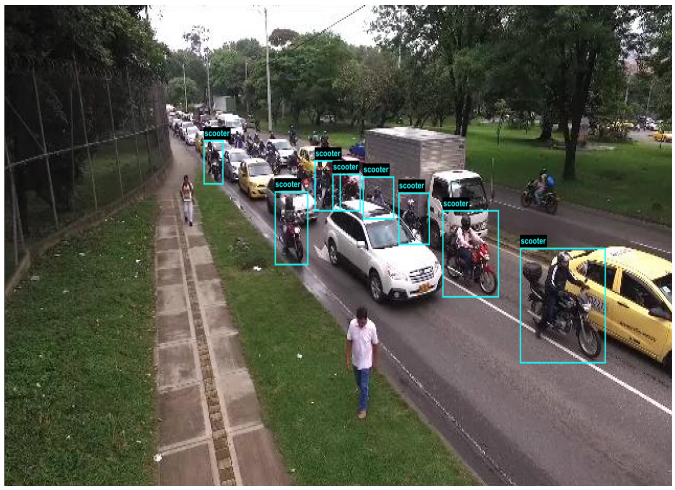
3.1.4 Traffic Intersection Dataset

In addition to the datasets mentioned above, we also created our own dataset, the Traffic Intersection Dataset (TID), to fine-tune our pre-trained models. This dataset was created to assess whether object detection techniques are suitable for traffic analysis. The TID was constructed using traffic camera footage shot from a static camera at an intersection in the province of Utrecht in the Netherlands that was collected over two weeks. 2043, 821, and 263 frames were reserved for training, validation, and testing, respectively. Furthermore, the test-set from this dataset is used for all the experiments.



Source: Karlsruhe Institute of Technology

Figure 2: Sample image from the Specialized Cyclist dataset.



Source: MB10000

Figure 3: Sample image from the MB10000 dataset.

Figure 4 illustrates a sample image with cyclists from the Traffic Intersection dataset. Furthermore, Tables 3 and 4 display the number of boxes per class and the number of unique instances per class, i.e., different objects, respectively.

3.2 Preprocessing

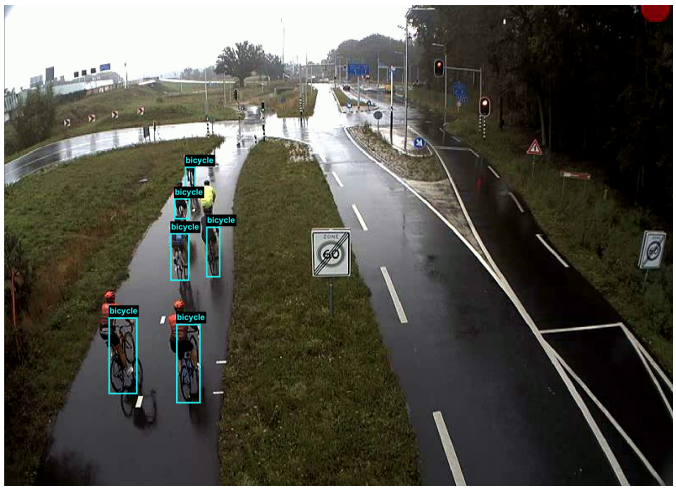
Since the datasets contained images that are sometimes unclear due to noise or blurriness, we apply the following preprocessing step before training, testing[1], and validating our data:

Given I is an RGB image of $height \times width \times channels$, we first normalize the image, after which we apply an unsharp masking filter to get the processed image (see below for a more detailed description). Figure 5 demonstrates an example of an input image before and after applying the preprocessing step.

3.2.1 Algorithm

Step 1: normalize the input image by applying a normalization function f to the RGB channels of every pixel at position i, j in the image I:

[1]The preprocessing step is also applied to the test set since we noticed that this increased the accuracy of the models in our case.



Source: Province of Utrecht & Meetel B.V.

Figure 4: Sample image from the traffic intersection dataset.

Class	Boxes	% of Total
bicycle	5559	74.48
scooter	1369	18.34
two-wheeler	128	1.72
pedestrian	408	5.47
Total	7464	100

Table 3: Number of boxes per class in the traffic intersection dataset.

$$I_{norm}(I) = f(I_{i,j}) \tag{1}$$

where

$$f(c) = \frac{c - c_{min}}{c_{max}} \tag{2}$$

with c being a color channel matrix, and c_{min} and c_{max}, representing the minimum and maximum values of the color channels. $c_{min} = 0$ and $c_{max} = 255$, in case of RGB.

Step 2: remove blurriness from the image by applying an unsharp masking operation $h(u,v)$ to the blurry image I (Kausar et al., 2020):

$$I_{sharp}(I) = I \cdot h(u,v) \tag{3}$$

where

$$h(u,v) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{u^2+v^2}{2\sigma^2}\right] e^{-\frac{u^2+v^2}{2\sigma^2}} \tag{4}$$

with u and v representing the horizontal and vertical distance from the origin and σ being the standard deviation.

Step 3: combine the steps mentioned above to the input image I to get the processed image:

$$I_{processed} = I_{sharp}(I_{norm}(I)) \tag{5}$$

Class	Instances	% of Total
bicycle	113	80.14
scooter	22	15.6
two-wheeler	2	1.42
pedestrian	4	2.84
Total	141	100

Table 4: Unique instances per class in the traffic intersection dataset.

Figure 5: Input image before (left) and after (right) applying the preprocessing step.

3.3 Data Augmentations

During the development of our models, we discovered that training only on plain datasets, without any augmentations, resulted in poor generalizability. This is because models that use convolutional neural networks (CNN's) are prone to overfitting on certain features, such as angle and color. For this reason, we apply the following image augmentations (see Figure 6) from the Albumentations library (Buslaev et al., 2020):

- ChannelShuffle: randomly shuffle the RGB channels of the image.
- HorizontalFlip: randomly flip the image around the x-axis.
- Rotate: randomly rotate the image.

3.4 Models

The resolution of the images from the TID is not very high (800x608 pixels), and objects further in the frame are nearly unidentifiable even by human observers. Therefore, a model with high accuracy is essential to correctly detect and calculate the metrics of the road participants that are further away in the image. Furthermore, processing speed is not the primary concern since the model is not used for performing real-time detections. However, we still aim to develop the most time-efficient solution possible, since traffic monitoring videos can be week-long.

We experiment with YOLOv5 variants and Faster R-CNN to discover the most accurate detection model among them for detecting two-wheelers. This choice of models is motivated by the research of Kausar et al. (2020) that demonstrated state-of-the-art object detection accuracy for Faster-RCNN with RESNET50 compared to YOLOv3, but did not test YOLOv3's successor YOLOv5, since it was not released yet at the time of writing.

3.4.1 Learning Rate Selection

The learning rate is a vital hyperparameter used while training deep neural networks. A learning rate that is too large could mean that the optimization algorithm never finds the minimum point of the loss function. On the other hand, a learning rate that is too low could mean that the optimization algorithm takes a long time to minimize the loss function, thus increasing the time it takes to train the model or, in the worst case, causing the model never to converge.

To choose the optimal starting learning rate for our models, we utilize a method called cyclical learning rates, first described by Smith (2017). This method finds the optimal learning rate by letting the learning rate cyclically vary between a minimum and a maximum boundary value. The optimized learning rate is defined using the training and validation sets before the start of the main training loop.

3.4.2 Ensemble Model

Since we selected multiple datasets for training our models, we also needed to consider that some datasets might contain images with unannotated objects, resulting in false negatives during detection. For example, the MB10000 dataset also contained instances of pedestrians which were not annotated, causing the loss to increase (see Figure 8). To prevent this from happening, we employed an approach called ensemble learning. With ensemble learning, multiple models can be combined to make predictions (Polikar, 2006). In our case, we are using three models, each responsible for detecting one class (bicycle/scooter/pedestrian). By performing inference with all three models on an image, all the desired classes can be detected.

The ensemble pipeline (see Figure 7) consists of the following steps:

1. Use a preprocessed image as input for the detection models.
2. Each detection model (see Section 4.3.2) returns predictions for the input image.
3. Use non-max suppression to select the best predictions.
4. Return the predicted bounding boxes for the given input image.

3.4.3 Surrogate Learning

One caveat of the ensemble learning method is that it is relatively slow since it has to pass the input image through each detection model separately. Ideally, we would pass the input image only through one detection model, but we chose to use ensemble learning since this is not possible. However, there is an alternative to the

Figure 6: Data augmentations.

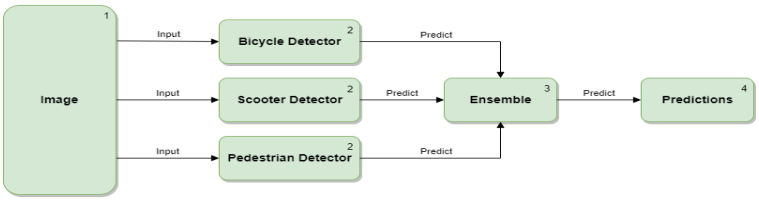
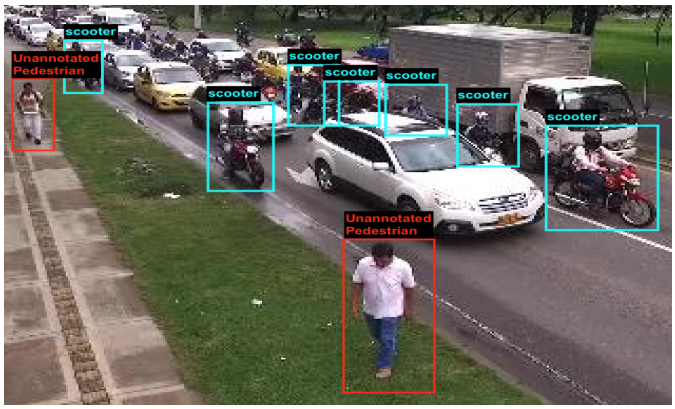


Figure 7: Ensemble pipeline.



Source: MB10000

Figure 8: Example of an image containing an unannotated class (pedestrian).

ensemble approach, namely surrogate learning. Surrogate learning is a machine learning approach in which a model is trained using the output from other models, i.e., the bounding boxes returned by our object detectors.

The surrogate learning pipeline (see Figure 9) works as follows:

1. The missing classes in the external datasets are annotated with predictions from multiple models. Each model is used to annotate a missing class from a dataset.
2. The surrogate model is trained on the all external datasets combined, which now have annotations for the missing classes.
3. During inference, a preprocessed image is put into the trained surrogate model.
4. Return the predicted bounding boxes for the input image.

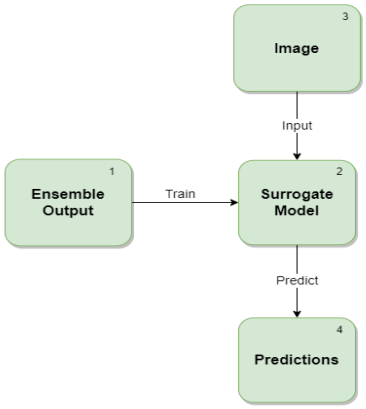


Figure 9: Surrogate pipeline.

4 Experiments & Results

During our research, multiple experiments were set up to assess the performance and accuracy of the models. This section describes the experiments we set up and the corresponding experimental results. For all the experiments, the learning rate was found using the learning rate finder described in Section 3.4.1. Furthermore, augmentations were used during training in all the experiments as described in section 3.3. Testing was always performed using the Traffic Intersection Dataset (TID) mentioned in Section 3.1.4.

4.1 Experiment 1: Comparing Object Detection Models

For the first experiment, different models, such as Faster-RCNN and YOLOv5 variants small and medium, were compared to make a substantiated choice on which model to use for creating the ensemble and surrogate models. The YOLOv5 models were pre-trained on the COCO dataset, whereas the Faster-RCNN used the pre-trained weights from ImageNET. The models were trained on the TID for 300 epochs. The class 'two-wheeler' was not used because there were few samples to achieve an unbiased test result.

The results for experiment 1 are demonstrated in Table 5. Where Faster-RCNN showed a higher mAP, F1-score, and recall, followed by YOLOv5s and YOLOv5m. These results were similar to the results from (Kausar et al., 2020), where Faster-RCNN also yielded better results than YOLOv5's predecessor YOLOv3.

Although the Faster-RCNN model had the best accuracy in this experiment, the following experiments will only utilize YOLOv5s. Since YOLOv5 has a faster inference time compared to Faster-RCNN, we chose to use this model, since inference time is important when processing week-long videos, to get a quick result.

4.2 Experiment 2: Pre-training on Specialized External Datasets

In experiment 2, we tested if pre-training on specialized external datasets (BDD100k, SCD & MB10000 for pedestrians, bicycles & scooters, respectively) is beneficial for increasing model accuracy. Concretely, we trained YOLOv5s models with and without using pre-trained weights as a base and compared the test results. Furthermore, each model focussed on one specific class, which means we filtered out the other labels from the datasets before training. So, for instance, the models that focused on pedestrians only used the "pedestrian" class from the TID and BDD100k datasets.

Table 6 shows that the YOLOv5s pedestrian detection model with no pre-training on the BDD100k dataset yielded no detections. However, with pre-training, the model returned slightly better results than without, with 0.1 mAP, 0.07 F1-score, 0.13 precision, and 0.05 recall. Furthermore, pre-training the YOLOv5s bicycle detection model on the SCD resulted in decreased accuracy compared to no pre-training, with 0.64 mAP, 0.71 F1-score, 0.83 precision, and 0.62 recall. Finally, it can also be noticed that pre-training the YOLOv5s scooter detection model on the MB10000 dataset resulted in a slight improvement compared to no pre-training, with 0.76 mAP, 0.8 F1-score, 0.79 precision, and 0.81 recall.

4.3 Experiment 3: Ensemble and Surrogate learning

In the final experiment, we compared a fine-tuned YOLOv5s model pre-trained on a combination of the SCD, BDD100K, and the MB10000 datasets to fine-tuned ensemble and surrogate models. This experiment aims to evaluate if a surrogate model will perform better than a model trained on concatenated datasets that might be missing labels for certain objects. We elaborate on the definitions of ensemble and surrogate in Section 4.3.2 & 4.3.3.

4.3.1 Concatenated Dataset

First, a model was trained using combined external datasets (SCD + MB10000 + BDD100k) to assess whether surrogate learning is better than simply training on concatenated external datasets. Testing the model without fine-tuning resulted in no detections. Afterward, this model was fine-tuned on the TID and tested again, resulting in a considerable improvement with 0.68 mAP, 0.76 F1-score, 0.8 precision, and 0.73 recall (see Table 7).

4.3.2 Ensemble Model

The ensemble model consisted of three YOLOv5s object detection models. Each model detects one separate class, i.e., pedestrians, cyclists, and scooters. First, these models were trained separately on the BDD100K, SCD, and MB10000 datasets, respectively. After which, they were tested with and without fine-tuning on the TID to analyze if a decent score could be achieved without the need for fine-tuning on the TID.

The results without fine-tuning on the TID (see Table 7) demonstrate that an ensemble of three models performed better than the model that was trained on the concatenated dataset, which did not detect anything at all. However, the score was still too low to be considered useful.

With fine-tuning, the model using concatenated datasets outperforms the ensemble model considerably with higher mAP, F1-score, precision, and recall.

4.3.3 Surrogate Model

The ensemble model was used to generate annotations for the missing classes in the external datasets. These annotations were then used to generate the annotations of the external datasets with the missing classes while preserving the original annotations. A YOLOv5s model without fine-tuning was then trained using these updated annotations.

The resulting model performed worse than the ensemble model (with and without fine-tuning) but better than the model without fine-tuning using concatenated datasets (see Table 7). However, with fine-tuning, the surrogate model performed better than the ensemble model (with and without fine-tuning) and slightly worse than the fine-tuned model using concatenated datasets.

5 Discussion

In this section, the results of the experiments are discussed and interpreted.

5.1 Experiment 1

The Faster-RCNN model performed better than the YOLOv5 models in the first experiment, which might be caused by the YOLOv5 model having difficulty in detecting objects that are small and close to each other (Gandhi, 2018). Faster-RCNN, on the other hand, was better at detecting small objects since its classifier is possibly better able to adapt to small objects (Eggert et al., 2017).

5.2 Experiment 2

Fine-tuning a model on our own dataset (TID) using pre-trained weights with a model trained on the COCO + External datasets (BDD100k, SCD, MB10000), resulted in worse overall performance than using a model that was only trained on the TID. The same can be

Model	Pre-Trained	Training Set	Testing Set	mAP	F1	Precision	Recall
Faster-RCNN	ImageNET	TID	TID	**0.88**	**0.81**	0.77	**0.85**
YOLOv5s	COCO	TID	TID	0.73	0.78	**0.83**	0.74
YOLOv5m	COCO	TID	TID	0.72	0.74	0.82	0.67

Table 5: Comparing object detection models.

Model	Pre-Trained	Training Set	Testing Set	mAP	F1	Precision	Recall
YOLOv5s	COCO	TID (pedestrians)	TID (pedestrians)	0.0	0.0	0.0	0.0
YOLOv5s	COCO & BDD100K	TID (pedestrians)	TID (pedestrians)	**0.10**	**0.07**	**0.13**	**0.05**
YOLOv5s	COCO	TID (bicycles)	TID (bicycles)	**0.68**	**0.76**	**0.85**	**0.68**
YOLOv5s	COCO & SCD	TID (bicycles)	TID (bicycles)	0.64	0.71	0.83	0.62
YOLOv5s	COCO	TID (scooters)	TID (scooters)	0.71	**0.81**	**0.91**	0.74
YOLOv5s	COCO & MB10000	TID (scooters)	TID (scooters)	**0.76**	0.8	0.79	**0.81**

Table 6: Pre-training on specialized external datasets.

Model	Pre-Trained	Training Set	Testing Set	mAP	F1	Precision	Recall
YOLOv5s	COCO	TID	TID	**0.73**	**0.78**	**0.83**	**0.74**
Concatenated	COCO	External	TID	0.0	0.0	0.0	0.0
Ensemble	COCO	External	TID	0.03	0.01	0.02	0.01
Surrogate	COCO	External	TID	0.002	0.0	0.0	0.0
Concatenated	COCO & External	TID	TID	0.68	0.76	0.8	0.73
Ensemble	COCO & External	TID	TID	0.18	0.51	0.57	0.46
Surrogate	COCO & External	TID	TID	0.67	0.73	0.8	0.68

Table 7: Ensemble and surrogate learning.

observed for the model pretrained on the SCD and fine-tuned on the TID (only bicycle class). It is likely that the SCD dataset is too different from the TID (bicycle class only). The images taken in the SCD are taken horizontally, compared to the traffic camera which has an overview from higher up. Another difference is the SCD has annotated the entire bicycle (bicycle + person), whilst we have only annotated the bicycle itself. Datasets that are too different would cause the model to forget certain features during training, thus decreasing the model's accuracy.

5.3 Experiment 3

Combining all models into an ensemble lead to worse results when compared to experiments 1 and 2. This is likely due to the low accuracy of the model trained for pedestrian identification, which lowered the score of the model. These bad performing models would have created false annotations for the surrogate model, which would also lower the performance of the surrogate model. In the end, it was better to not perform surrogate learning, with the help of the ensemble models, since concatenating the dataset without extra annotations performed better.

For the models without fine-tuning, the surrogate model performed worse than the ensemble model, which was expected. The ensemble models used for creating the annotations did not generalize to the other datasets, which were very different from which they were trained on. Models trained on the SCD and MB10000 dataset, have created false annotations for the remaining classes as seen in Figure 11, which could cause lower scores. The model trained on the BDD100k dataset generalized well and have committed fewer mistakes for the other datasets. In Figure 10 shows pedestrians that have been annotated which were not annotated before (Figure 8).

The images in the MB10000 dataset were all very similar to each other, they all have the same camera angle, time of day and weather. This resulted in a model that did not generalize well. The BDD100k dataset all had different types of images in different scenarios and was able to create a general model which could annotate the other datasets with a good precision.

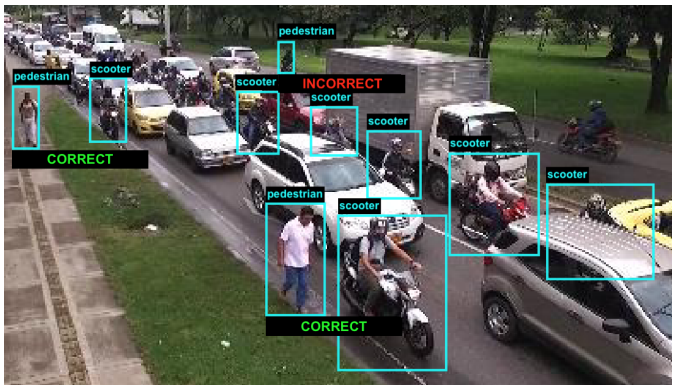


Figure 10: Correct annotation from the ensemble model in the MB10000 dataset.

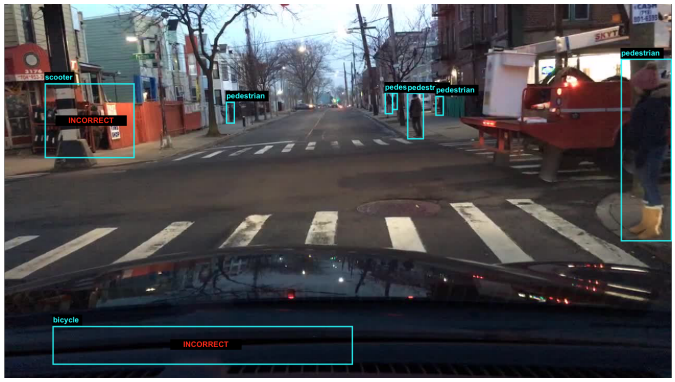


Figure 11: Incorrect annotation from the ensemble model in the BDD100K dataset.

6 Conclusion

We have shown that Faster-RCNN is better than YOLOv5 when detecting bicycles and scooters in our TID dataset. Furthermore, creating an ensemble led to worse results, since some models did not generalize well to all classes. Creating an ensemble model could still be viable when few sub-models are used and inference time is not a concern.

The surrogate model trained on the combined datasets, annotated by the ensemble model, performed better than the ensemble model but worse than a model that was only trained on our dataset. However, the concept of a surrogate model could still prove useful when better models are used, because surrogate models are able to create annotations without the need for humans.

Finally, pre-training a model on external datasets (SCD, BDD100K, and MB10000) did not always yield better results. For pedestrians, it did perform better due to pedestrians being an underrepresented class in the TID. For the bicycle and scooter classes, it performed slightly worse.

7 Future Work

The Faster-RCNN network scored well in experiment 1 but was not explored further in experiments 2 and 3, so it could be interesting to repeat the experiments shown in the study with Faster-RCNN.

When developing ensemble models for surrogate learning, it is important that all models that are included in the ensemble show good performance on the other datasets, otherwise under-performing models can significantly hamper the results. The Traffic Intersection Dataset had an underrepresented pedestrian class, resulting in bad model performance. The number of pedestrians in the TID dataset will need to be increased to improve the model's accuracy.

Finally, to train a well-generalized model, we recommend using datasets of bicycles and scooters with more variety in place, time, and weather, since we have seen that models trained on datasets that lack variety, often performed poorly in our experiments.

Acknowledgements

This project is financially supported by Regieorgaan SIA (part of NWO) under the KIEM SI project STATS.

Regieorgaan SIA PROVINCIE UTRECHT MEETEL

REFERENCES

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. *Information*, *11*(2). Retrieved from https://www.mdpi.com/2078-2489/11/2/125 doi: 10.3390/info11020125

Eggert, C., Brehm, S., Winschel, A., Zecha, D., & Lienhart, R. (2017). A closer look: Small object detection in faster r-cnn. In *2017 ieee international conference on multimedia and expo (icme)* (pp. 421–426).

Espinosa, J. E., Velastin, S. A., & Branch, J. W. (2018, aug). Motorcycle detection and classification in urban Scenarios using a model based on Faster R-CNN. *arXiv:1808.02299 [cs]*. Retrieved 2018-08-13, from http://arxiv.org/abs/1808.02299 (arXiv: 1808.02299)

Foroozandeh Shahraki, F. (2017, 08). *Cyclist Detection, Tracking, and Trajectory Analysis in Urban Traffic Video Data* (Tech. Rep.). doi: 10.34917/11156724

Gandhi, R. (2018, 12). *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms.* Retrieved from https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

Kausar, A., Jamil, A., Nida, N., & Yousaf, M. H. (2020, 08). Two-wheeled vehicle detection using two-step and single-step deep learning models. *Arabian Journal for Science and Engineering*, *45*. doi: 10.1007/s13369-020-04837-4

Li, X., Flohr, F., Yang, Y., Xiong, T., Braun, M., Pan, S., ... Gavrila, D. (2016a, 06). A new benchmark for vision-based cyclist detection. In (pp. 1028–1033). doi: 10.1109/IVS.2016.7535515

Li, X., Li, L., Flohr, F., Wang, J., Xiong, H., Bernhard, M., ... Li, K. (2016b). A unified framework for concurrent pedestrian and cyclist detection. *IEEE transactions on intelligent transportation systems*, *18*(2), 269–281.

Masalov, A., Matrenin, P., Ota, J., Wirth, F., Stiller, C., Corbet, H., & Lee, E. (2019). Specialized cyclist detection dataset: Challenging real-world computer vision dataset for cyclist detection using a monocular rgb camera. In *2019 ieee intelligent vehicles symposium (iv)* (pp. 114–118). doi: 10.1109/IVS.2019.8813814

Naseralavi, S., Nadimi, N., Saffarzadeh, M., & Mamdoohi, A. R. (2013, 10). A general formulation for time-to-collision safety indicator. *Proceedings of the ICE - Transport*, *166*, 294–304. doi: 10.1680/tran.11.00031

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, *6*(3), 21–45.

Redmon, J., & Farhadi, A. (2016). *Yolo9000: Better, faster, stronger.*

Reurings, M. C. B., Vlakveld, W. P., Twisk, D. A. M., Dijkstra, A., & Wijnen, W. (2012, Aug 23). *Van fietsongeval naar maatregelen: kennis en hiaten; inventarisatie ten behoeve van de nationale onderzoeksagenda fietsveiligheid (noaf)* (Tech. Rep.). Leidschendam. (R-2012-8)

Sayed, T., Zaki, M. H., & Autey, J. (2013). Automated safety diagnosis of vehicle–bicycle interactions using computer vision analysis. *Safety Science*, *59*, 163–172. Retrieved from https://www.sciencedirect.com/science/article/pii/S0925753513001240 doi: https://doi.org/10.1016/j.ssci.2013.05.009

Smith, L. N. (2017). *Cyclical learning rates for training neural networks.*

van der Horst, A. R. A., de Goede, M., de Hair-Buijssen, S., & Methorst, R. (2014). Traffic conflicts on bicycle paths: A systematic observation of behaviour from video. *Accident Analysis & Prevention*, *62*, 358–368. Retrieved from https://www.sciencedirect.com/science/article/pii/S0001457513001401 doi: https://doi.org/10.1016/j.aap.2013.04.005

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. *CoRR*, *abs/1805.04687*. Retrieved from http://arxiv.org/abs/1805.04687

A Traffic Light State Detection

A.1 Introduction

Passing a red light can be considered a dangerous situation, therefore our system should be able to extract relevant traffic light information for this analysis. In the Netherlands, there is always one light on at a time. In the traffic light the top light is red (stop), the middle light is orange (stop if possible), the bottom light is green (clear to pass).

A.2 Methodology

The steps below describe how to detect the traffic light state from a traffic light in an input image.

1. Input an image.
2. Realign the image (optional).
3. Select the traffic light's Region of Interest (ROI).
4. Split the ROI vertically in three sections (red, orange, green).
5. Calculate and compare the average brightness in each section.

A.2.1 Input an Image

For the input image, sequential frames were extracted from an input video, with a frame-rate of 5FPS.

A.2.2 Realigning the Image

In our case, the ROI of the traffic light had to be pixel-precise since small movements can cause the ROI to shift outside of the traffic light since the resolution of the traffic light is low (3x10 pixels). For example, a shift of 1 - 2 pixels side to side, or 3 pixels up and down, can break the state detection.

Therefore, despite having a fixed position camera, the traffic light has to be re-identified sometimes. On the first frame of a video, the homography is calculated using a known reference image that was compared to the first frame of the video. The calculated homography can then be applied to all the future frames without the need to recalculate.

Since the traffic light state does not change every frame, the state is only read every five frames. With a 5 FPS video, this will result in a detection every second.

A.2.3 Selecting the Traffic Light's ROI

Given a fixed position camera, the traffic light can manually be selected with a Region Of Interest (ROI) with a set of X, Y coordinates in the video frames. The ROI (see Figure A1) will be as big as the traffic light itself, so 3x9 pixels (width x height) in this case.

A.2.4 Splitting the ROI Vertically in Three Sections

The state of the traffic light can be detected based on color, location of brightness, or both. In the daytime, colors are harder to see because of bright sunlight and are more washed out than nighttime videos where the colors are clearly visible. Based on this premise, the choice was made to detect the state based on the brightness. First, the frame is converted to gray-scale to make it easier to process the brightness. Then the traffic light is split into three vertical sections, all with the same dimensions. In Figure A2 the three regions are visible, with the bottom region being brighter than the two regions on top. This indicates the green light will be considered activate.

A.2.5 Calculating the Average Brightness in Each Section

A score of each section can be calculated with an average brightness of all the pixels in this section. The region with a higher score wins and is considered 'on'. The score is calculated as follows:

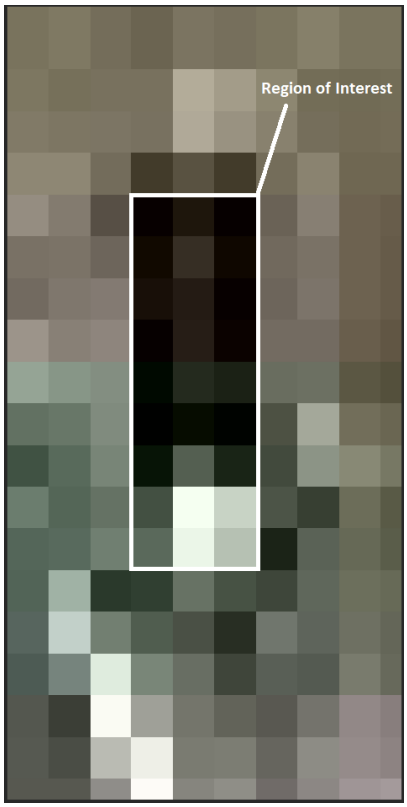


Figure A1: Selecting the traffic light's ROI.

Figure A2: Splitting the ROI vertically in three sections.

$$score = \frac{\sum pixel\ values\ in\ region}{Number\ of\ pixels} \tag{6}$$