Universitat Oberta de Catalunya

uoc.edu

Design of an image fusion object detection algorithm for the autonomous driving during adversarial weather conditions

Guillem Tudela Debrigode
University Master's Degree in Data Science
Area 2

Antonio Lozano Bagén
Jordi Casas Roma

January 2021

FITXA DEL TREBALL FINAL

Títol del treball:	*Design of an image fusion object detection algorithm for the autonomous driving during adversarial weather conditions*
Nom de l'autor:	*Guillem Tudela Debrigode*
Nom del consultor/a:	*Antonio Lozano Bagén*
Nom del PRA:	*Jordi Casas Roma*
Data de lliurament (mm/aaaa):	*01/2021*
Titulació o programa:	*Màster Universitari en Ciència de Dades*
Àrea del Treball Final:	*Àrea 2*
Idioma del treball:	*Anglès*
Paraules clau	*image fusion, autonomous driving, object detection algorithm*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

Aquest treball dissenya i implementa un algorisme de detecció d'objectes basat només en imatges amb l'objectiu de ser usat en conducció autònoma davant de situacions climatològiques adverses, tals com la neu, la boira o la pluja.

S'emprarà la base de dades DENSE [4] tant per a l'entrenament com pel testeig dels models.

Basat en l'arquitectura dels SSD [21], en primer lloc s'estudia la precisió obtinguda per cadascun dels sensors per separat, a continuació s'estudien diferents mètodes per a augmentar la precisió del model. Després es dissenyen models basats en cadascun dels tres mètodes de fusió d'imatges: la fusió basada en píxels, la fusió dels sensors i la fusió de característiques. Per a aquest darrer mètode es dissenyaran els submodels de fusió intermèdia i fusió tardana.

Els models són avaluats i comparats entre si, i es conclou que el model amb la millor ràtio entre precisió i rapidesa de detecció és el model basat amb la fusió intermèdia de les característiques.

Abstract (in English, 250 words or less):

During this thesis an image-only object detection algorithm is designed and implemented. Its purpose is to be used on real-time situation as a detection algorithm for autonomous driving under adversarial weather conditions, such as snowstorms, fog or rain.

The DENSE Dataset [4] is used to train and evaluate the model.

Based in the SSD architecture [21], the precision of each sensor is studied. Then other methods are tried to increase the precision of the model.

Then other models are designed, one for each of the image-fusion methods: pixel-fusion, sensor-fusion and feature-fusion. For the last method a halfway fusion and a late fusion model are implemented.

All models are evaluated, and its results are compared. Optimising detection velocity and precision it is concluded, the best model is the one which applies the feature fusion with halfway fusion.

Index

List of figures

List of equations

List of tables

1. Introduction

1.1. Context and justification of the thesis

The interest for autonomous driving has been growing steadily during the past few years up to the point that, under favourable weather conditions, full automated driving is nowadays almost a reality. But it is still far for reaching its fully potential, mainly because there are still situations where an autonomous system is not completely reliable, especially those with adversarial weather conditions. While autonomous cars drive more kilometres, more experience may be gained, but the number of accidents in which an autonomous car is involved increases too, sometimes with fatal consequences such as injuries or even casualties. Despite it is already true that these systems are safer than human-driven ones, the technology still needs years of learning to mature.

One of the most basic tasks of the autonomous driving consists in the detection and classification of objects. Seeing what surrounds the car is an important task because it may be necessary to use the brakes to avoid a collision. This is nowadays mainly achieved using cameras, whose accuracy is improved using radar and LIDAR (Laser Imaging Detection and Ranging) sensors. However, the purchase and installation of LIDAR in serial cars is extremely expensive, which may make it suitable as a standard equipment for premium-class vehicles only, but not for standard ones. Tesla, for instance, builds up its automated driving algorithm by relying only on the data from radar and cameras and aims not to use LIDAR to detect objects in serial vehicles. LIDAR may however remain in use mostly for data collecting and testing purposes.

To train an object detection algorithm, a huge number of images is needed. Open datasets to be used as a basis are available and can be used for research purposes, like the KITTI [1], NuScenes [2] or the dataset from Udacity [3], to cite a few of them, which typically offer visible-spectrum images (RGB) and sometimes LIDAR data. They show interurban and urban scenes, mostly during daytime, and even if some of them include nightlight scenes as well, they mostly rely on clear weather conditions, meaning a sunny, cloud-free day. Perhaps in the sunny-blessed coastline of California, Tesla's homeland, it may not be necessary to train the model in other conditions than sunny ones, but in the rest of the world other weather conditions occur, such as fog, rain, snow or dust storms. An autonomous car needs to be ready to face all kind of weather situations and still be able to detect objects correctly if it wants to be named rightfully as autonomous. Here resides the importance of the selected dataset for the model training, as a biased one could induce to an algorithm which only works on specific conditions, like sunny days, but fails to detect objects during a snowstorm, because it has never been trained to detect objects under this new weather condition.

Taking advantage of the release of the DENSE dataset [4] with urban and interurban images taken under clear and adversarial weather conditions it came the idea to design an object detection algorithm trained using this data. Another major improvement of this dataset, in comparison to those previously mentioned, is the inclusion of images taken by near infrared (NIR) and far infrared (FIR) cameras. The publishers of the DENSE dataset trained an object detection model combining the data from the RGB, the NIR, radar and LIDAR, otherwise for this thesis another approach will be followed, in which only images will be combined: the NIR, the FIR and the RGB spectrums will be fused to build a more reliable model and to compensate the loss of the radar and LIDAR data as inputs.

1.2. Main objectives

The main objective of the thesis is to determine whether the radar and LIDAR input data can be dismissed for the object detection and classification under adversarial weather conditions tasks and if they can be replaced by the NIR and FIR images instead. To achieve it, an object detection algorithm exclusively based on images will be designed, trained and tested.

The calculated precision results of the model will be compared with the ones obtained by the DENSE dataset researchers, a paper published by the University of Ulm, the University of Princeton in cooperation with Mercedes-Benz AG and Algolux researchers and available at the Arxiv portal [5]. As the only self-imposed constraint, the algorithm should work on real-time automated driving applications, hence speed should prevail above accuracy.

The secondary objective of this project is to test which image fusion approach shows the best speed-accuracy ratio and to determine if a model trained using only clear weather images can be reused under other weather conditions as well and still be able to detect objects properly, implying it generalises correctly.

1.3. Strategy and methodology

The starting point is the analysis of the standalone contribution of each one of the image-spectrums sensors to determine which precision can be achieved by each sensor separately. Starting with the RGB-only model, the hyperparameters of the algorithm will be optimised and used for the rest of the models to facilitate the comparison of the results. Afterwards, a NIR-only model and a FIR-only model will be designed and trained.

Once the contribution of each standalone image-spectrum is known, the fusion of the RGB, NIR and FIR cameras can start. There are three different ways to fuse images (explained in detail in the next chapter) and one model for each of the methods will be designed and tested.

The precision of each of the models will be calculated to determine which model achieve the better results.

All the models will be trained from scratch to avoid possible biases of the loaded model.

1.4. Thesis plan

The most time-intensive part of the project is the training of the models, hence a GPU will be used to accelerate the calculations, specifically an 8 GB GeForce GTX1070.

All the images needed to feed the model are taken from the DENSE dataset and no extra images are added. Python has been chosen as the language used for the data preparation, the pipeline and for the model design and test, while Keras is the deep learning framework. Each part of the project has been written in separate Jupyter notebooks.

The object detection algorithm is not designed from scratch. Following the recommendations from M. Elgendy [6] the SSD algorithm has been borrowed from P. Ferrari [7]. This detection algorithm works with images belonging to one source at each time and with three channels only, meaning it is necessary to adapt the code so it can also work with multichannel images and with multiple inputs. For the calculation of the precision the function from R. Padilla [8] is used.

The project has been divided in seven parts. It began with the research phase, the documentation of interesting topics related to the deep learning and automated driving. It continued with the project definition and planning and the state-of-the-art documentation, task that had begun with the research phase. Parallelly the design and implementation phase started: the dataset was downloaded, its data was explored and prepared. The next step was the hyperparameter tuning and the design of the algorithm, task that took two months. During this phase the achievements and the steps descriptions were documented to accelerate the thesis write-up. The final steps were the project presentation and the project defence.

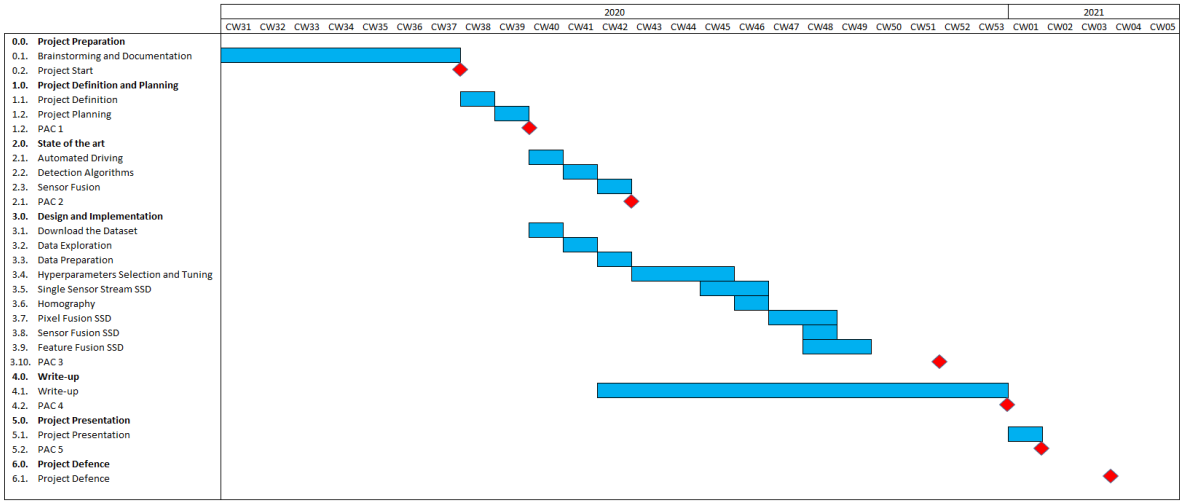


Figure 1: Gant project diagram
Source: self-made

1.5. Summary of the results

During the data exploration phase, it was noticed that a tiny time offset between the FIR images and the other two spectrums existed. The time difference was small but big enough to allow the objects of the image to move to a different position, hence the fusion of images could be affected reducing the model precision. It was decided to dismiss the FIR images for this thesis, remaining only NIR and RGB in use. Only using two sensors, instead of the four used by the researcher's solution could explain why the obtained precisions of this project are not as good as those published at the researcher's paper [5]. If the FIR images have had no offset, this precision gap could have been reduced, but with the currently available dataset is not possible to prove it. This may explain the fact as well that the researcher's paper dismisses the FIR data, although its reason is not mentioned in the paper itself.

The fusion of images has proved to be a good approach to increase the precision of the detection algorithm in comparison to those approaches based on one-sensor-only. The best model has been uploaded to a Github repository as well as the ".h5" file and the instructions to use them [35]. The other trained models will not be uploaded because of save-space limitations. Regarding the images from the DENSE dataset they will be neither uploaded because its licence specifies, they can only be used in scientific papers. Nevertheless, on chapter 3 is explained how to obtain the needed images.

1.6. Summary of the rest of the chapters

The second chapter explains the state of the art of the main topics of this project. First, the different kind of object detection algorithms will be explained, followed by the description of the metrics used to measure the algorithm effectiveness. Secondly, the sensor used for the autonomous driving will be listed. Then the theory about the image and the

wavelength of the light will be explained and, finally, some light will be shed on the three ways to achieve the image fusion.

The third chapter presents the DENSE dataset and goes through the paper “Seeing Through Fog Without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather” published by the researchers [5]. On this chapter the researchers’ object detection algorithm approach will be described, and the achieved precision by model will be shown.

The fourth chapter describes all the activities done previous the start of the algorithm design, then it describes and evaluates the models. The results will be commented in the fifth chapter.

The conclusions and the future work are described in the sixth chapter. Followed by the glossary, the bibliography and the appendix.

2. State of the art

2.1. The object detection algorithms

Computer vision can be defined by its purpose, i.e. to replicate the work of the human eyes and brain to allow computers to see and understand a scene or image [13]. There are many activities from the autonomous driving which can be achieved using computer vision, such as “objection detection, object classification, lane detection and object tracking” [14] but in this thesis the focus will lay on the object detection. As explained in [15] “The problem definition of object detection is to determine where objects are located in a given image (object localization) and which category each object belongs to (object classification)”.

The fundamental structure of the detector algorithm is the so-called backbone, which is tipically a deep learning network such as VGG16 [16] or ResNET [17] which extracts the features of the images, classifies them into one of the possible classes (or object targets) and locates the object within the image [18]. For each of the detections a confidence score is given, meaning how sure is the algorithm to have found an object as well as the class it belongs to.

2.1.1. One stage and two stage detectors

The algorithm detection model can be classified in two groups: the one-stage and the two-stages detectors, depending on the number of phases used to accomplish the detection task.

While one-stage detectors accomplish simultanously the localisation and classification of the object, the two-stage ones generate first regions of interest, i.e. regions where an object may be found, and then extract and classify these [18]. One-stage detectors are tipically faster but achieve lower precision rates than those obtained by two-stages ones.

From the first group, the most well-known detectors are the YOLO [19] (with the fourth actualisation having been recently released [20]) and the SSD [21]. From the second group, the R-CNN [22], the Fast R-CNN [23] and the Faster R-CNN [24] stand out.

The output of the detection algorithm can be either a bounding box surrounding the object location or a mask of the object, being the last option more common for image segmentation tasks, which are out of the scope of this thesis.

2.1.2. Evaluation metrics

The most common metrics of the object detection algorithms are the mean average precision (mAP) and the frame per second (FPS), which

combined determine not only how precise a detection is, but also how fast it can be calculated. For the use case of an object detection algorithm for automated driving purposes, a reasonable high precision using low computation time is needed.

In order to calculate the mAP, the value intersection over union (IoU) needs to be obtained, which measures the overlap between the ground truth and the bounding box, i.e. the real location of the object and the predicted one. The determination of the real bounding boxes, as well as the class the detection belongs to (such as car, pedestrian, track, bicycle...) is a task called labelling which is human-manually performed and provided along each training dataset.

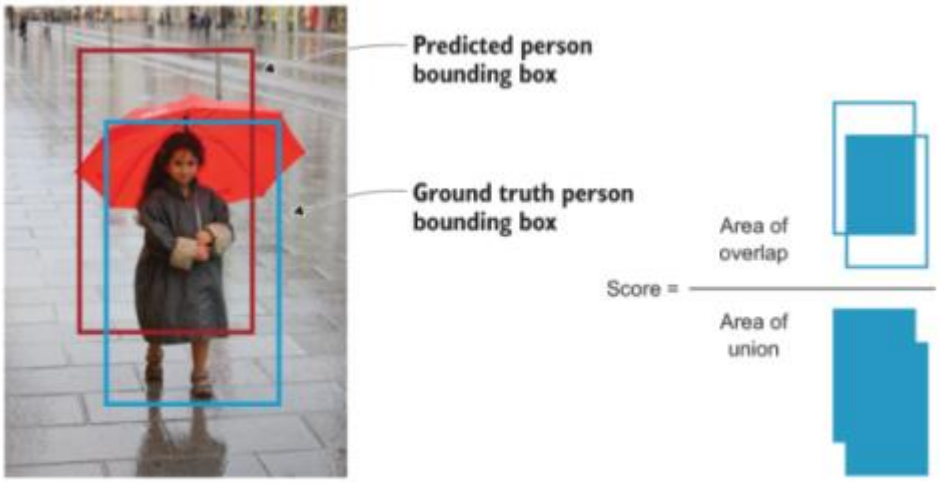


Figure 2: IoU calculation
Source: [6]

The higher the area of overlap, the better the score of the IoU is, and, consequently, the higher precision the detection algorithm will have. The values of the IoU range from zero (no overlap) to one (perfect match between ground truth's and prediction's bounding boxes).

To ensure a successful prediction score it is necessary to set a threshold to dismiss the predictions with a lower IoU, which is typically set to 50%, but higher values can also be used. If an object is correctly predicted, meaning that the prediction matches both ground truth and predicted class, is considered as a true positive (TP). The predicted objects with an IoU value below the threshold are considered as false positives (FP). The false detections, meaning the object has been correctly detected but not the class it belongs, will be considered as FP too. On the other hand, non-detected ground truths are called false negatives (FN). The detection algorithm gives a confidence value for each prediction, meaning how sure is the model that a detected object belongs to a specific class. It is common to dismiss the predictions with a lower confidence value and its consequences can be seen better in the Figure 4 and Figure 5. But to determine the perfect threshold for the model is first necessary to calculate its precision and recall values.

Given a certain confidence threshold, the precision of the prediction is calculated as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{all detections}}$$

Equation 1: Precision formula
Source: [8]

On the other hand, the recall of the prediction is calculated with the following formula:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{all ground truths}}$$

Equation 2: Recall formula
Source: [8]

While the precision of the model indicates the percentage of the true positives within all the detections, the recall sets the relationship between the true positives and all the ground truths, hence expressing the quality of the model. The perfect model has both high precision and high recall values.

In order to define the perfect threshold for a certain model the TP and FP values must be calculated, throughout all images and at different confidence thresholds independently for each one of the defined classes. The results can be graphically expressed in the so-called precision-recall curve, which represents the precision and recall of the object detector model at different thresholds. This chart will be drawn for each of the defined classes.

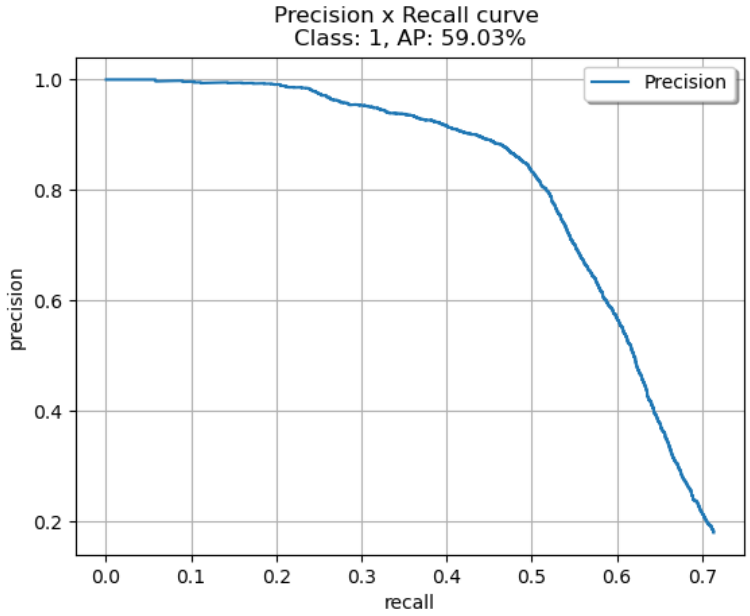


Figure 3: Precision-recall curve

Source: self-made with [8] function

To help to understand the curve and the confidence-threshold concept the following images may help.

Figure 4: Object detections with a threshold of 0.01
Source: self-made with a ground image from [4]

On the previous image, it can be seen what happens when the confidence threshold is set too low, in that case, close to zero. The blue boxes represent the detected objects according to the model while the green ones are the original ground truths. Each of the blue boxes include the predicted object class (PassengerCar, LargeVehicle, Pedestrian or RidableVehicle) as well as the model detection confidence. With a lower confidence threshold all objects will be found, so the recall will achieve the rate of 100% because the FN will drop to zero, but the FP will increase too, reducing consequently the precision of the model, for example, the model sees pedestrian everywhere, even as part of the buildings or the road.

On the contrary, if the threshold is set to a higher number like 0.3, the image gets clearer as the number of FP and TP drops, as only three objects are (correctly) detected while the rest of the objects remain unseen.

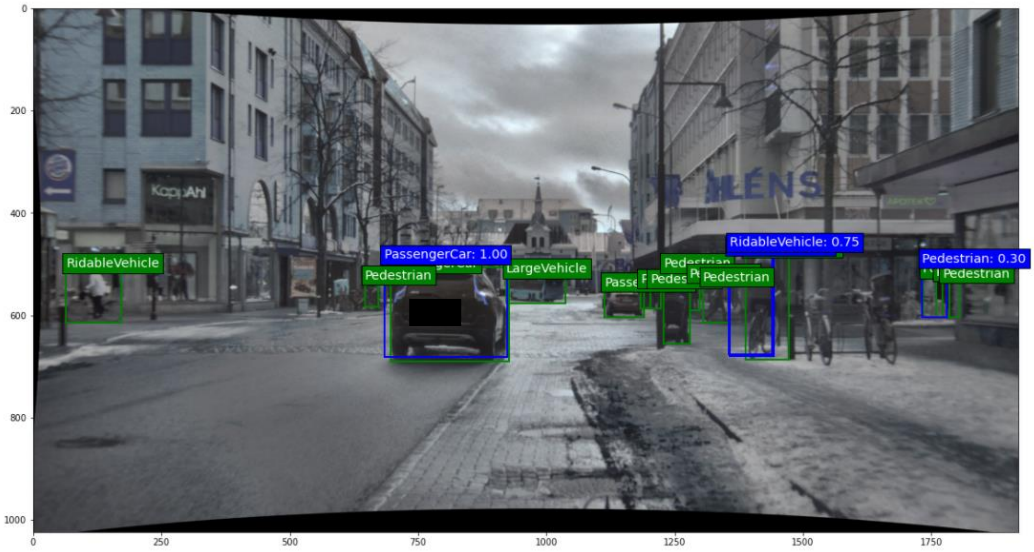


Figure 5: Object detections with a threshold of 0.30
Source: self-made with a ground image from [4]

A summary of TP, FP, Precision and Recall at different thresholds for the class PassengerCar from the last image can be seen in the next table. Of course, its results differ from those from the precision-recall curve previously seen, as the graph was calculated using the data from all images, not only one.

Confidence threshold	TP	FP	FN	Precision	Recall
0.01	2	14	0	0.125	1
0.3	1	0	1	1	0.5
0.5	1	0	1	1	0.5
0.75	1	0	1	1	0.5
1	1	0	1	1	0.5

Table 1: Example detection metrics
Source: self-made

In this case and for this class (PassengerCar), it could be concluded that any confidence threshold starting from 0.3 would be valid.

The average precision (AP) for a specific class can be obtained by calculating the area under the precision-recall curve. If this step is performed for each of the classes the algorithm is designed to detect it will be obtained one of the most important metrics to define an object detection algorithm: the mean average precision (mAP), which indicates how good the algorithm is detecting along all the classes.

The other most important metric is the frame per second (FPS), which shows how fast the algorithm is in detecting objects. The benefits of FPS depend on the use case of the algorithm, but for real time applications, the FPS is a decisive factor: as a perfectly high precision is useless if it provides the results too late. Considering a standard value for video

applications of 24 frames per second, this value will be set as the algorithm's FPS lower-threshold frame rate.

2.1.3. Algorithm comparison and selection

In the Table 2 the comparison of the AP of 20 classes achieved by different object detection algorithms with the PASCAL VOC2007 dataset [25] is shown.

TABLE II
COMPARATIVE RESULTS ON VOC 2007 TEST SET (%).

Methods	Trained on	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN (Alex) [15]	07	68.1	72.8	56.8	43.0	36.8	66.3	74.2	67.6	34.4	63.5	54.5	61.2	69.1	68.6	58.7	33.4	62.9	51.1	62.5	68.6	58.5
R-CNN(VGG16) [15]	07	73.4	77.0	63.4	45.4	44.6	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	35.6	66.8	67.2	70.4	71.1	66.0
SPP-net(ZF) [64]	07	68.5	71.7	58.7	41.9	42.5	67.7	72.1	73.8	34.7	67.0	63.4	66.0	72.5	71.3	58.9	32.8	60.9	56.1	67.9	68.8	60.9
GCNN [70]	07	68.3	77.3	68.5	52.4	38.6	78.5	79.5	81.0	47.1	73.6	64.5	77.2	80.5	75.8	66.6	34.3	65.2	64.4	75.6	66.4	66.8
Bayes [85]	07	74.1	83.2	67.0	50.8	51.6	76.2	81.4	77.2	48.1	78.9	65.6	77.3	78.4	75.1	70.1	41.4	69.6	60.8	70.2	73.7	68.5
Fast R-CNN [16]	07+12	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4	70.0
SDP+CRC [33]	07	76.1	79.4	68.2	52.6	46.0	78.4	78.4	81.0	46.7	73.5	65.3	78.6	81.0	76.7	77.3	39.0	65.1	67.2	77.5	70.3	68.9
SubCNN [60]	07	70.2	80.5	69.5	60.3	47.9	79.0	78.7	84.2	48.5	73.9	63.0	82.7	80.6	76.0	70.2	38.2	62.4	67.7	77.7	60.5	68.5
StuffNet30 [100]	07	72.6	81.7	70.6	60.5	53.0	81.5	83.7	83.9	52.2	78.9	70.7	85.0	85.7	77.0	78.7	42.2	73.6	69.2	79.2	73.8	72.7
NOC [114]	07+12	76.3	81.4	74.4	61.7	60.8	84.7	78.2	82.9	53.0	79.2	69.2	83.2	83.2	78.5	68.0	45.0	71.6	76.7	82.2	75.7	73.3
MR-CNN&S-CNN [110]	07+12	80.3	84.1	78.5	70.8	**68.5**	88.0	85.9	87.8	60.3	85.2	73.7	87.2	86.5	85.0	76.4	48.5	76.3	75.5	85.0	81.0	78.2
HyperNet [101]	07+12	77.4	83.3	75.0	69.1	62.4	83.1	87.4	87.4	57.1	79.8	71.4	85.1	85.1	80.0	79.1	51.2	79.1	75.7	80.9	76.5	76.3
MS-CR [104]	07+12	80.0	81.0	77.4	72.1	64.3	88.2	88.1	88.4	64.4	85.4	73.1	87.3	87.4	85.1	79.6	50.1	78.4	79.5	86.9	75.5	78.6
OHEM+Fast R-CNN [113]	07+12	80.6	85.7	79.8	69.9	60.8	88.3	87.9	89.6	59.7	85.1	76.5	87.1	87.3	82.4	78.8	53.7	80.5	78.7	84.5	80.7	78.9
ION [95]	07+12+S	80.2	85.2	78.8	70.9	62.6	86.6	86.9	**89.8**	61.7	86.9	76.5	**88.4**	87.5	83.4	80.5	52.4	78.1	77.2	86.9	**83.5**	79.2
Faster R-CNN [18]	07	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6	69.9
Faster R-CNN [18]	07+12	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6	73.2
Faster R-CNN [18]	07+12+COCO	84.3	82.0	77.7	68.9	65.7	88.1	88.4	88.9	63.6	86.3	70.8	85.9	87.6	80.1	82.3	53.6	80.4	75.8	86.6	78.9	78.8
SSD300 [71]	07+12+COCO	80.9	86.3	79.0	**76.2**	57.6	87.3	88.2	88.6	60.5	85.4	**76.7**	87.5	**89.2**	84.5	81.4	55.0	81.9	**81.5**	85.9	78.9	79.6
SSD512 [71]	07+12+COCO	**86.6**	**88.3**	**82.4**	76.0	66.3	**88.6**	**88.9**	89.1	**65.1**	**88.4**	73.6	86.5	88.9	**85.3**	**84.6**	**59.1**	**85.0**	80.4	**87.4**	81.2	**81.6**

* '07': VOC2007 trainval, '07+12': union of VOC2007 and VOC2012 trainval, '07+12+COCO': trained on COCO trainval35k at first and then fine-tuned on 07+12. The S in ION '07+12+S' denotes SBD segmentation labels.

Table 2: Comparative AP results on VOC2007 dataset
Source: [15]

Even if theoretically two-stage detectors deliver higher mAPs (see section 2.1.1), Table 2 shows that one-stage SSD detectors achieve bigger mAP values than the two-stage detectors such as the Faster R-CNN or the R-CNN.

When it comes to compare FPS values, as shown in the next table, the best results are as expected for the one-stage detectors, leading the classification the YOLO and the SSD300.

TABLE V
COMPARISON OF TESTING CONSUMPTION ON VOC 07 TEST SET.

Methods	Trained on	mAP(%)	Test time(sec/img)	Rate(FPS)
SS+R-CNN [15]	07	66.0	32.84	0.03
SS+SPP-net [64]	07	63.1	2.3	0.44
SS+FRCN [16]	07+12	66.9	1.72	0.6
SDP+CRC [33]	07	68.9	0.47	2.1
SS+HyperNet* [101]	07+12	76.3	0.20	5
MR-CNN&:S-CNN [110]	07+12	78.2	30	0.03
ION [95]	07+12+S	79.2	1.92	0.5
Faster R-CNN(VGG16) [18]	07+12	73.2	0.11	9.1
Faster R-CNN(ResNet101) [18]	07+12	**83.8**	2.24	0.4
YOLO [17]	07+12	63.4	**0.02**	**45**
SSD300 [71]	07+12	74.3	**0.02**	**46**
SSD512 [71]	07+12	76.8	0.05	19
R-FCN(ResNet101) [65]	07+12+coco	83.6	0.17	5.9
YOLOv2(544*544) [72]	07+12	78.6	0.03	40
DSSD321(ResNet101) [73]	07+12	78.6	0.07	13.6
DSOD300 [74]	07+12+coco	81.7	0.06	17.4
PVANET+ [116]	07+12+coco	**83.8**	0.05	21.7
PVANET+(compress) [116]	07+12+coco	82.9	0.03	31.3

* SS: Selective Search [15], SS*: 'fast mode' Selective Search [16], HyperNet*: the speed up version of HyperNet and PAVNET+ (compress): PAVNET with additional bounding box voting and compressed fully convolutional layers.

Table 3: Comparative FPS results on VOC2007 dataset
Source: [15]

In conclusion and in order to achieve the best of both worlds, the SSD300 is the chosen architecture for this thesis.

2.1.4. SSD Structure

The SSD model uses a VGG16 [16] network as a backbone, truncated before the classification layers. After the truncation, two fully connection

layers are added as well as four convolutional blocks, each of them with two convolutional layers which decreases in size. The output of the last convolutional layers is used to feed the detection layer and finally a non-maximum suppression is applied to output only the best results.

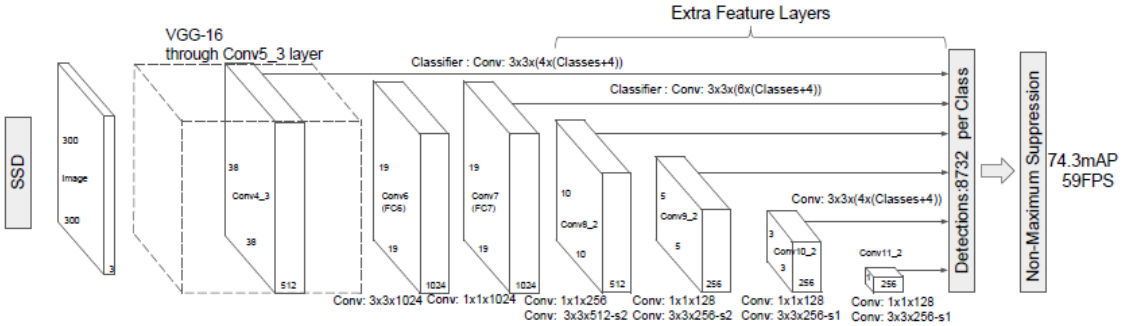
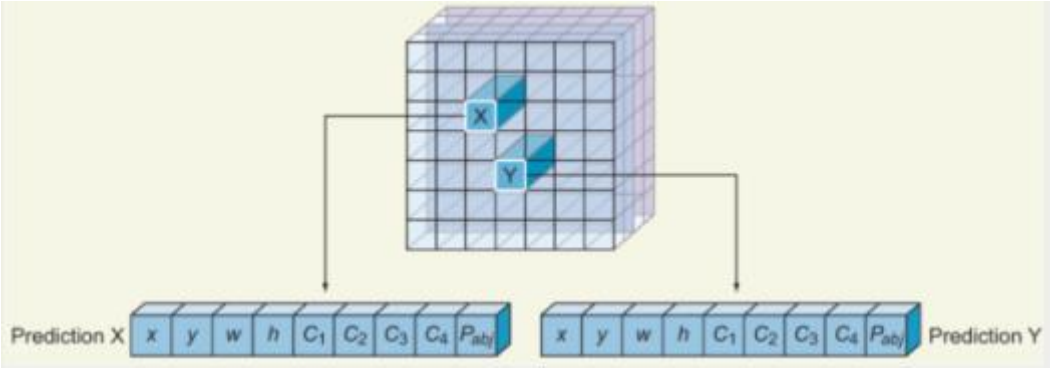


Figure 6: SSD Structure
Source: [21]

The original SSD300 has four bounding boxes[1] at the first and last two detection layers and for the rest of the layers it has six, resulting a total of 8732 predictions per class. To obtain this number the output of each detection layer has to be computed and its result summed, for example, the first detection layer is the result of applying a 3x3 convolutional to a 38x38x512, and as said, this fist layer has four bounding boxes, this means for one class a total of 38x38x4 (5776) bounding boxes will be applied to look for each class specificaly. The rest of the layers will detect 19x19x6 (2166), 10x10x6 (600), 5x5x6 (150), 3x3x4 (36) and 1x1x4 (4) boxes respectivaly, summing the output of the rest of the layers the number 8732 is obtained. The deeper layers are mainly used to detect the bigger objects. This limited number of bounding boxes is the main reason of the SSD model being faster than the two-stage detectors.

Each detection describes the location of the bounding box, gives a confidence value of being an object and also a probability value for belonging to each of the classes.



[1] Technically is one bounding box with four different scaling factors, i.e. a new box after reducing or increasing the original width's and high's box. The bounding boxes are also known as anchor boxes.

Figure 7: Raw prediction output
Source: [6]

As for each location different scaling factors are applied, the same object can be multiple times detected, or may have been partially contained in the contiguous box too. Some of this detections may detect a half of the object, or only a small part of it. To eliminate those predictions and to keep only the best of them the non-maximum supression is applied. This step consist in sorting all the predictions and keeping those with the top scores only if in the same region no other prediction for the same class has been previously selected (because it could mean both predictions belong to the same object). The chosen threshold can affect the detection rate in images in which lot of objects are close to each other, for example, a face detection algorithm in a (pre-Covid-19) concert image or a car detection algorithm in a parking's mall.

2.2. Autonomous driving and sensors

Autonomous driving systems relies mostly on data from cameras to detect objects, but other sensors can be helpful as object detectors and other tasks as well. Radars, nowadays a standard equipment sensor, are used to determine the exact distance to the surrounding objects and, if necessary, activate the emergency break. There are proximity sensors whose data feeds the parking assistance function and the GPS allows the position determination and the trajectory planning. Recently the use of the LIDAR has gained ground among some car companies. This sensor generates a 360 degrees points-cloud, which is a 3D reproduction of the surroundings of the car, a helpful information for object detection and for trajectory planning too.

This thesis focusses on the object detection task, a well-studied field of autonomous driving that performs one of its basic activities: replace the human eyes and see what is in front of the car to drive safely. While under clear weather conditions this task can be easily done, detecting objects in adversarial weather conditions, like those experienced during a snow or dust storm, under heavy rain, with dense fog or at night, is a more difficult activity. Under those circumstances the camera, radar and LIDAR do not work in proper conditions and its data-output is degraded, as summarised in the next table.

Sensor	Recognition technology	Sun glare	Rain	Fog	Snow
LiDAR	-Self-localization -Traffic participant recognition -Static/dynamic objects estimation		-Reflectivity degradation -Reduction in recognition distance -Shape change due to splash	-Reflectivity degradation -Reduction in measurement distance	-Change in peripheral shape -Road surface occlusion -Noise due to snow fall
MWR	-Self-localization -Static/dynamic objects estimation		-Reduction in measurement distance	-Reduction in measurement distance	-Mild noise due to fallen snow
Camera	-Self-localization -Traffic light detection -Traffic participant recognition -Static/dynamic objects estimation	-Whiteout of objects	-Visibility degradation	-Visibility degradation	-Visibility degradation -Road surface occlusion

Table 4: Influences of adversarial weather conditions
Source: [26]

Under adversarial weather circumstances, the visibility of the camera is generally degraded. Radars, on the other hand, can return noise as it confuses the snowflakes with objects, giving the wrong idea of an immediate collision; rain and fog can reduce the measurement distance of the sensor. The LIDAR output is also hard affected by those adversarial conditions, the reflectivity is degraded and the water splash and the snowflakes produce noise too, increasing the chance of false detections.

Based on the previous information it can be concluded no sensor works in perfect conditions all the time; hence, it seems a good idea not to rely in one sensor only, but to use more sensors and with different features. There are multiple possible combinations but as cost is one decision-making factor for automotive manufacturers, and as cameras are the cheapest of them, this thesis will work with the hypothesis of using only cameras to feed the object detection algorithm, concretely the designed model will be based on the information from light visible and the infrared cameras.

2.3. Theory of the image

Before starting with the image fusion explanation, it is necessary to stop first in the image itself. An image is a combination of pixels, which are commonly represented by a number from 0 to 255 depending on its pixel intensity, a number which can be stored using 8 bits (2^8). The closer the number to zero is, the more intense the colour and vice versa. Mathematically an image can be represented as a three-dimensional matrix, being the rows the width and the columns the height of it, while the third dimensional corresponds to the number of channels. A colour image has three channels, also known as red, green and blue channels or for its acronym RGB; on the contrary, a grayscale image has only one channel. Another way of representing the colours is the HSL or the HSB colour space, in the first case each pixel will be described by its hue, saturation and lightness while in the second case, its pixels will be described by the hue, saturation and brightness [9].

From the physics point of view the visible light, hence what humans can see, is an electromagnetic radiation whose wavelength lies between 390 nm and 700 nm [10]. The smaller values correspond to the blues while the bigger ones are the orange and reds, in between there are the rest of the colours the human eyes can see. The electromagnetic spectrum region above the visible light is called the infrared region, which can also be divided in three subregions: named near infrared, mid-wave infrared and far infrared, depending on its wavelenght values [11]. The other electromagnetic spectrums can be seen in the next figure.

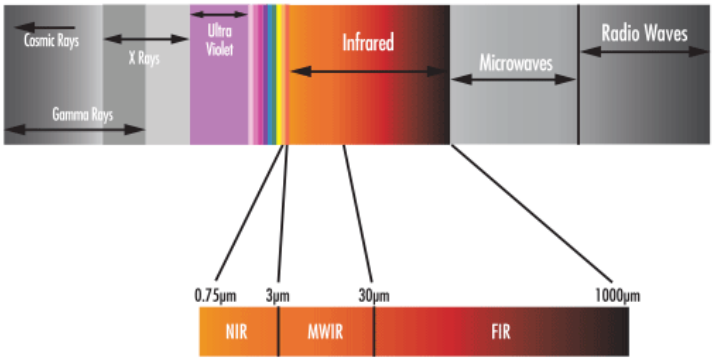


Figure 8: The electromagnetic spectrum
Source: [11]

While RGB cameras do work fairly well during daytime hours, the infarred ones are commonly used for night-vision, so its combination with image fusion techniques could produce a robust model, which could work in both clear and adversarial weather conditions.

2.4. Image fusion

In order to deliver precise results for autonomous driving under adversarial weather conditions, there are two alternatives to the image fusion, which are the domain adaptation and the data pre-processing [5]. The first alternative consists in using style-transferring techniques to convert, for example, an image taken on a sunny day to a foggy one, while the second alternative focus on removing the adversarial condition of the image, i.e. removing the droplets, the snowflakes or even the fog. Nevertheless, this project will focus on the image fusion and will not try those alternatives.

The pedestrian detection is one of the mainly investigation lines of the multispectral image fusion, mostly involving the fusion of visible and infrared spectrums [27-32]. One main problem before the image fusion can be applied is the differences in the field of views (FoV) of the cameras, i.e. how much a camera can see, and that cameras are not placed in the exact same position, hence some angles and shapes may look different. To fix the first problem there is no other solution but to reduce the bigger FoV to equal the lowest one. As for the second problem, homography techniques are applied, i.e. the use of a transformation matrix to adapt the image plane.

There are three methods to fuse the images. The first one is called sensor fusion and it consist in working separately with each of the sensors. That means, each sensor will be trained in its own detection network and the predictions of all of them will be used to produce a more robust model. In this kind of methods is also possible to assign variable weights, for example, RGB outputs could be downgraded during nightlight scenes and, on the contrary, during daylight scenes they would be preferred instead of the NIR or FIR outputs.

The second method is known as pixel fusion. As explained in the previous chapter, the RGB image has three channels, so the main idea of this method is to add more information adding extra channels. With a perfect homography transformation the RGB and infrared images match pixel-wise perfectly, therefore the name of “pixel fusion”. Other possibilities are adding the magnitude gradient or the histogram of oriented gradient; some studies [27] show the combination of the ACF+T+THOG[2] channels increase the prediction power of the model.

The last method is called the feature fusion and it is based on the fusion of each image stream inside the network itself, meaning the fusion of the feature maps. The SSD300 must be reworked to accept two inputs instead of one and to concatenate the streams as well. For that purpose, a concatenation layer is added as well as a Network in Network (NiN) layer [32]. This new layer after the concatenation is used to reduce the number of dimensions by stacking the output of multiple convolutional layers, as there have not been a second stream, favouring the reuse of the weights from pretrained models for the last layers of the network, thus accelerating the training. Depending on in which moment the streams are fused, the fusion will be baptised as early fusion, halfway fusion or late fusion.

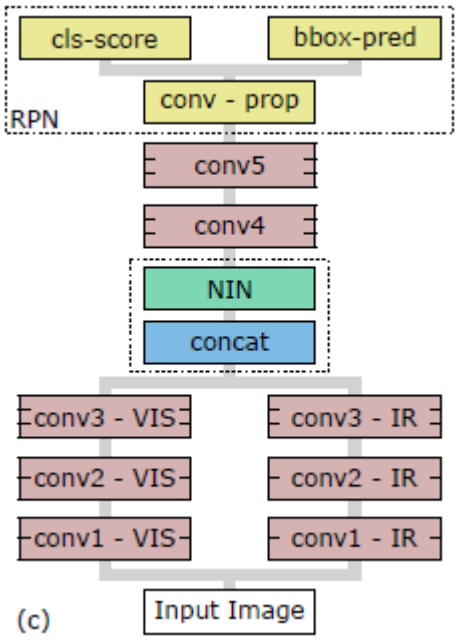


Figure 9: Two stream feature fusion model (halfway fusion)
Source: [27]

[2] ACF: the three channels of the CIELUV colour space, plus the magnitude gradient (M) and the six-gradient histogram (O).
T: thermal channel
THOG: gradient histogram of the thermal channel

3. The DENSE Dataset

3.1. Seeing through fog without seeing it

The DENSE Dataset was released along with the paper “Seeing Through Fog Without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather” in 2019 and is the product of more than 10000 km of a two-months-driving across the northern part of Europe, starting from Germany and ending up in Scandinavia. The most important feature of the dataset, and what makes it different form other available datasets, is the inclusion of scenes in adversarial weather conditions, such as dense fog or snowing conditions. It consists of a multimodal dataset and it includes data from radar, LIDAR, and cameras with RGB, NIR and FIR spectrums. To download it is necessary to fill a registration form using the following link [12]. For this thesis is not necessary to download the whole 600 Gb dataset, but a small portion of it, concretely the following folders: cam_stereo_left_lut, fir_axis, gated0_rect8, gated1_rect8, gated2_rect8 and gt_labels. The first folder contains the RGB images, the second the FIR images, the next three folders include the NIR images and the last one contains the ground truth labels for the RGB and NIR images, unfortunately the FIR’s ground truths are not available. The dataset is split between train, validation and test, as well as its image weather indication, and it can be found in the researcher’s Github repository [36].

3.2. The sensors

The vehicle was equipped with two RGB cameras, with a 1920×1024 resolution and a FoV of 39.6°×21.7°; the NIR camera, here called gated, has a resolution of 1280×720 while its FoV is 31.1°×17.8° and captures the scenes in three different depth-ranges, adjusting the flash to illuminate the desired depth; the FIR camera has a resolution of 640×480. The car is equipped with radar, LIDAR, a weather station and a road-friction sensor.

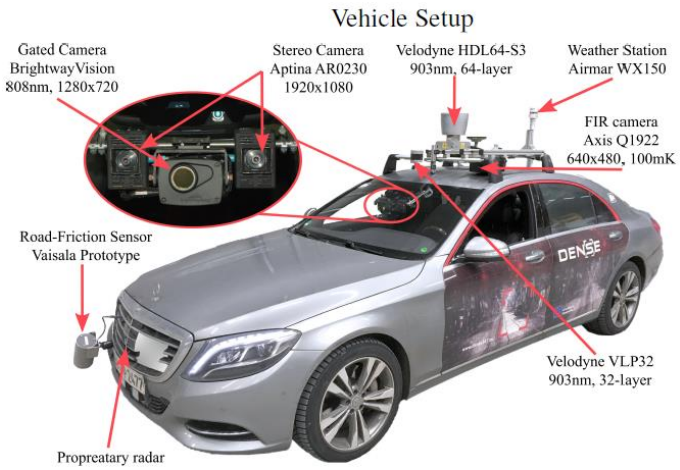


Figure 10: Vehicle setup
Source: [5]

3.3. The approach

The model uses the data from radar, LIDAR and the RGB and NIR cameras, while the FIR data is put aside. The RGB images see its FoV reduced to match it with the NIR one, and afterwards homography is applied to transform the NIR into the same RGB planar image.

The backbone of the model is a reworked VGG network, cut on the fourth convolutional block. Six layers are added to be used for the SSD detection layers, each of them decreasing in size. Each sensor has its own separate stream, and the entropy of the sensor is provided to steer the sensor-feature fusion. At each layer, the extracted features of the sensors are fused in the LIDAR-stream.

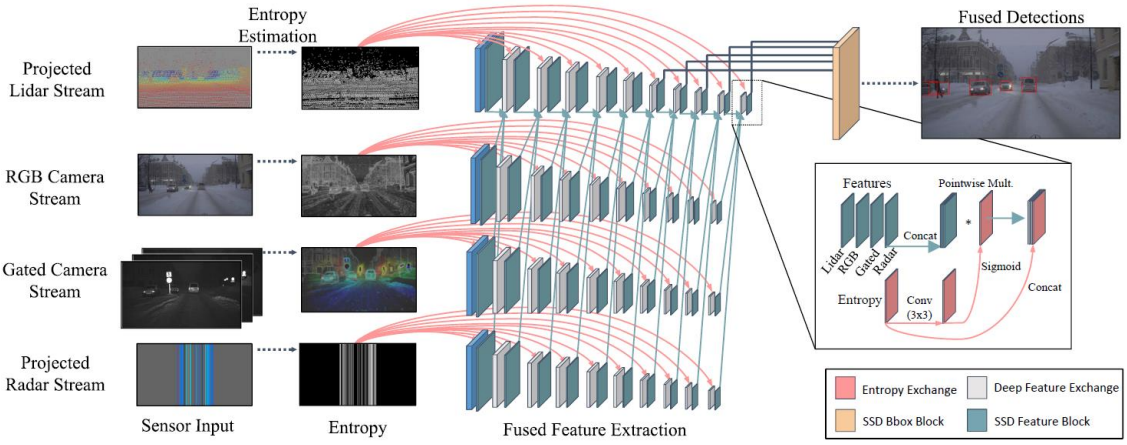


Figure 11: DENSE architecture
Source: [5]

3.4. Other tested solutions

The model was trained using only clear weather images and tested on clear and adversarial weather condition scenes. The dataset was split into train, validation and test assigning 3510, 779 and 8743 images to each group, respectively. To compare the obtained results other models were trained too, which maintain the same network architecture but use the data of one single sensor at each time. The average precisions for the car's class under different weather conditions are shown in the table below.

Weather	clear			light fog			dense fog			snow/rain		
Difficulty	easy	mod.	hard	easy	mod.	hard	easy	mod.	hard	easy	mod.	hard
Deep Entropy Fusion (this work)	89.84	**85.57**	**79.46**	90.54	**87.99**	**84.90**	87.68	**81.49**	**76.69**	88.99	**83.71**	**77.85**
Deep Fusion (this work)	**90.07**	80.31	77.82	**90.60**	81.08	79.63	86.77	77.28	73.93	**89.25**	79.09	70.51
Fusion SSD	87.73	78.02	69.49	88.33	78.65	76.54	74.07	68.46	63.23	85.49	75.28	67.48
Concat. SSD	86.12	76.62	68.61	87.98	78.24	70.17	77.99	69.16	67.07	83.63	73.65	66.26
ADDA [61][1]	85.27	70.51	67.86	87.83	78.68	70.38	87.64	78.12	74.37	84.17	74.25	66.86
CyCada [29][1]	88.50	77.84	69.56	89.08	79.36	75.58	87.24	77.04	73.38	85.56	74.80	67.22
Image-only SSD	85.43	75.75	67.79	87.76	78.52	70.43	**87.89**	78.25	74.96	84.33	74.38	67.01
Gated-only SSD	77.10	61.95	58.27	80.65	69.64	61.75	75.16	66.76	61.68	77.32	61.31	57.23
Lidar-only SSD	73.46	57.32	54.62	68.43	54.82	51.91	28.98	25.24	24.56	67.50	52.26	46.83
Radar-only SSD	10.26	8.54	8.23	16.92	13.24	12.66	16.33	13.57	13.00	12.94	10.95	10.40
AVOD-FPN [36]	66.47	58.71	51.63	60.40	52.51	51.92	33.95	26.29	26.17	59.55	51.91	50.54
Frustum PointNet [49]	80.06	75.89	67.70	84.06	76.88	75.44	76.69	73.62	68.49	78.34	74.34	66.52

Table 5: AP for cars on different weather conditions and algorithms
Source: [5]

The designed model outperforms the single-sensor models in almost all-weather conditions. Providing the sensor entropy at each layer increases the precision of the model as well, especially in the harder conditions, like scenes with dense fog. In most of the weather situations the precision exceeds the 80% precision-threshold, hence the researchers have built a model which detects a high quote of TP, while maintaining the FP and FN low.

Although the model had been trained using clear weather images only, there are almost no AP differences between the weather splits, proving that the model generalises sufficiently to predict unseen weather conditions.

Those precision values, specially the RGB-only (referenced in the Table 5 as Image-only SSD), the NIR-only (known as Gated-only SSD) and the Deep Fusion model, will be used as a reference to compare the own proposed model. Unfortunately, the publishers did not mention which hyperparameters have been used to obtain such results, it is even unknown which input size for the SSD have been used, hence AP differences are likely to occur, only its known, that the algorithm uses 21 anchor boxes, but not the chosen aspect ratio.

4. Design and test of an image-only object detector algorithm

4.1. The strategy

The first step before building the model is the exploration of the dataset, cleaning the labelled data if necessary, and exploring the class distribution. Another important step is the ground truth size exploration: as mentioned in section 3.4, the publishers of the DENSE dataset did not provide the hyperparameters of its proposed model, so it is necessary to understand how the ground truths look like to design a better bounding boxes combination for the SSD model.

Regarding the split between train, validation and test, this project will use the same split as the proposed on the paper, i.e. the model will be trained and validate using clear-weather images only. Additionally, the test split will be divided into daylight and nightlight scenes too, but the easy-moderate-hard split shown in Table 5 will not be followed, as is not clear which parameters have been used to define those categories.

Afterwards, the FoV between the three image spectrums will be equalled and homography transformations will be applied to match pixel-wise the RGB, NIR and FIR images. Finally, the first SSD model will be built to set the hyperparameters (which will be shared for all the models). Once they are settled, a single-sensor SSD model will be trained to know the precision each sensor can achieve separately. Then the three fusion models will be designed, and its results compared to determine which of the approaches show the best results.

4.2. Exploring the dataset

4.2.1. The classes

According to the paper [5] only four classes are available, but it was discovered there are some more. As part of the data preparation the list of classes was simplified to four.

	ID	TOTAL
0	DontCare	5296
1	LargeVehicle	5016
2	LargeVehicle_is_group	1
3	Obstacle	2343
4	PassengerCar	52580
5	PassengerCar_is_group	238
6	Pedestrian	40067
7	Pedestrian_is_group	1136
8	RidableVehicle	2431
9	RidableVehicle_is_group	4
10	Vehicle	1866
11	Vehicle_is_group	7
12	person	3
13	train	1

Table 6: Number of classes before the data cleaning
Source: self-made

In the following image it can be seen the class distribution of the RGB and NIR images. Its output differs because the NIR images have a reduced field of view, meaning the NIR image sees less scene as the RGB, hence less objects, but in both cases the homogeneity of the classes is maintained, being two classes overrepresented and two of them clearly underrepresented. In the distribution it can be seen as well, how the number of objects during the nightlight scenes is lightly reduced, compared to daylight images.

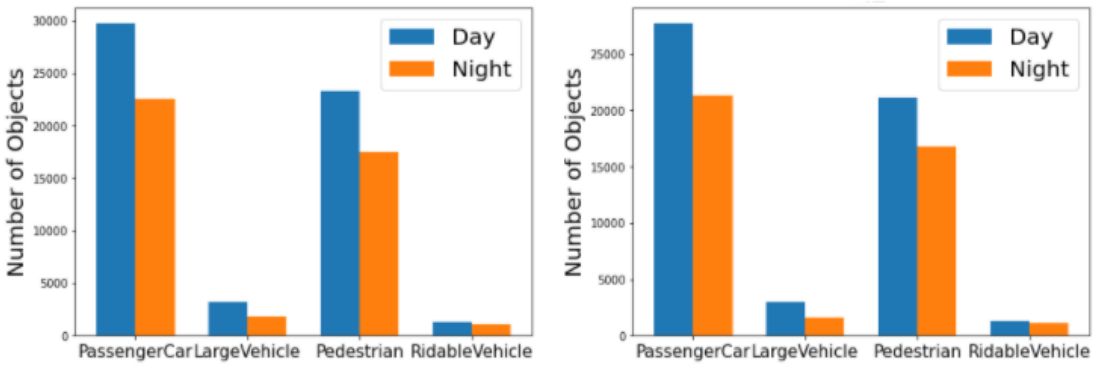


Figure 12: Class distribution for RGB (left) and NIR (right) images
Source: self-made

A detailed class distribution for each of the weather conditions can be found at the appendixes.

4.2.2. RGB, NIR and FIR images

The cameras of the vehicle have a different resolution and FoV depending on the type of the output as it had been explained on section 3.2. The only way to make the fusion of the three spectrums work

consists in reducing the shape of the images to the smaller one and then applying the homography transformation.

During the image exploration, it has been discovered the FIR images have not been taken in the same instant as the RGB and NIR were. Looking closely in Figure 13, a minor time offset can be appreciated, which invalidates the premise of the three-spectrum pixel-wise fusion, as the homography will never be able to transform the images to the same planar, because the objects are not in the same location.

Figure 13: Extract of the same image on FIR, RGB and NIR spectrum
Source: self-made with a ground image from [4]

On the previous image, it can be noticed how the legs of the man crossing the road show a different position depending on the image spectrum. While between RGB and NIR images there is no observable difference, the FIR and the rest show an offset. Therefore, regrettably the FIR spectrum will not be used.

It has been previously commented in the section 3.2, the NIR camera had a flash to focus on different depths so at night, with low visibility, each of the images will illuminate a different part of the scene: the so-called gated0, will focus the nearer objects, the gated1 the intermediate

and the gated2 the distant ones, in the Figure 41 in the appendix 9.7 it can be seen an example of how this variable focus looks like.

Afterwards, all the images were resized to a 300×300 resolution, as this is the necessary input's shape of the SSD300, and it can increase the model train velocity because the resize operation is saved.

4.2.3. The ground truths

As previously mentioned in section 2.1.4, the SSD algorithms does not make a massive exploration of all kind of bounding boxes everywhere, but it looks a limited number of them, hence, to increase the prediction score of the model it seems wise to know how exactly the ground truths of the dataset look like and then tune consequently the hyperparameters.

But SSD300 models, as stays implicit in its name, use 300×300 images as input. That means there is no need of calculating the original ground truths because they are not really what the model will have as input, but something more deformed. The reason is the reduction the image goes through to pass from a 1920×1024 to a 300×300 in case of the RGB images (the NIR transformation goes from 1280×720 to 300×300). That means a RGB is reduced almost six times in its x-axis and four times in its y-axis and, as the reduction is not equal in both axes, the aspect ratios of the ground truths will be affected too, becoming a thinner version of them, of course, width and height will be reduced accordingly too.

Visually is easier to understand how the shape reduction affects the image and the objects it contains.

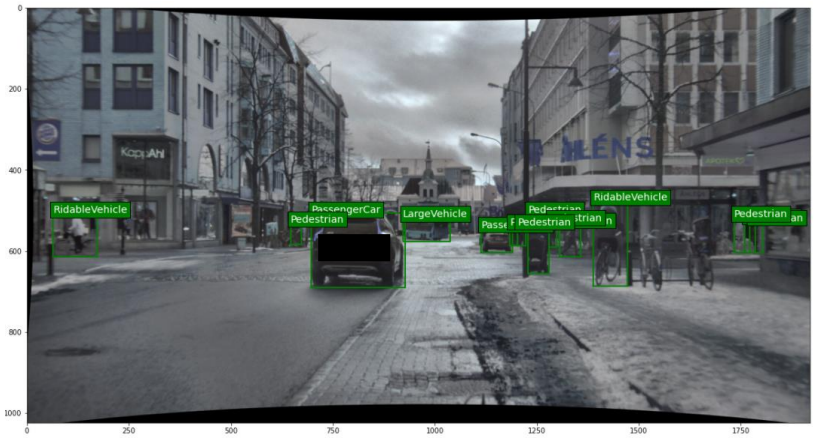


Figure 14: Image deformation before resizing
Source: self-made with a ground image from [4]

Figure 15: Image deformation after resizing
Source: self-made with a ground image from [4]

Focusing on a specific it is noticeable the deformation of the ground truths because of the image shape reduction.

Figure 16: Image deformation
Source: self-made with a ground image from [4]

In the following image is shown the ground truth sizes for each class. As each of them represent a different concept, the size of the ground truth varies too.

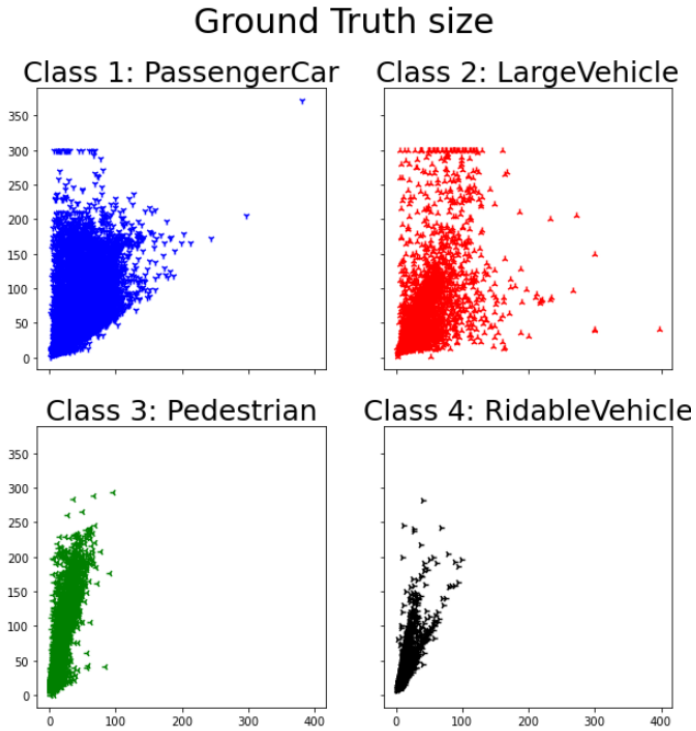


Figure 17: Ground truth sizes
Source: self-made

From the previous image it can be guessed the aspect ratios varies depending on the class. To see it more clearly the histogram of the ground truths has been represented.

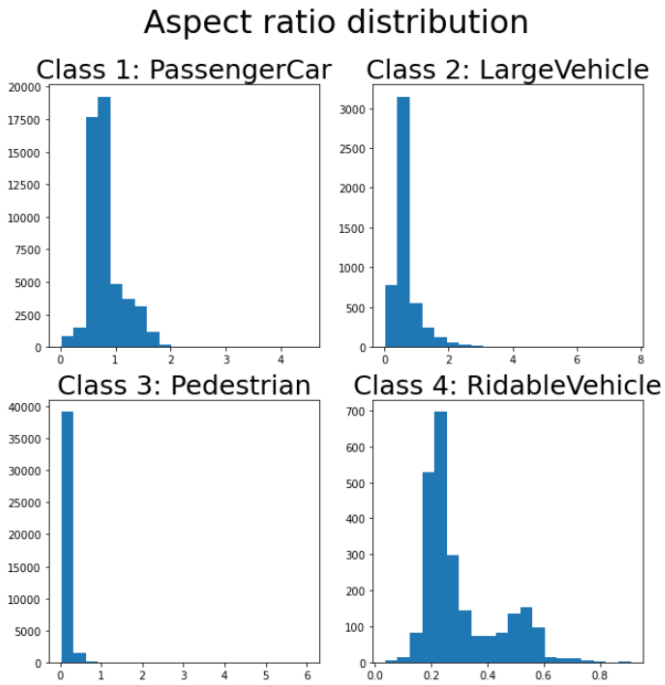


Figure 18: Aspect ratio distribution
Source: self-made

Vehicles, like passenger cars or large vehicles, have mainly a squared ground truth or a wider than taller one, but almost never twice the width-height relation, sometimes its aspect ratio is below one, but remaining in close to it. On the other hand, pedestrians and ridable vehicles have always an aspect ratio below one, mostly with values below 0.5, like 0.3 or 0.2, so the model needs to look to objects that are five times taller than wider to detect those targets.

As part of the data cleaning, it was discovered some ground truth had a width or a height equal to zero, so they were removed from the dataset.

4.3. Homography

Instead of using a manually point matching between the RGB and NIR images, as done by the original paper and explained in the supplement material [34], to obtain the transformation matrix the library OpenCV has been used with an automatic point selection. Then the RANSAC optimisation has been applied, as the original paper does.

Although this action has been repeated several times with different images to optimise the homography results, a perfect pixel-wise matching has never been obtained, and in some regions of the image the offset is visible.

Figure 19: Pixel-wise fusion of RGB and NIR after homography
Source: self-made with a ground image from [4]

In the previous image the area around the centre like the car, the bus or the church show no visible offset, the same can be said on the left part of the image. On the other hand, on the top part of the scene, like the buildings, or on the right, like the person cycling or the shop-sign this offset between the RGB and the NIR image is noticeable. This noise can cause a reduction of the predictions, but it is concentrated in small regions or places where no object or fewer objects are expected, therefore this project will continue even knowing a perfect homography has not been obtained.

The homography will affect the ground truths and the aspect ratio distribution too, as the reduction from the original image to the 300×300 input will now be not as severe as it used to be for the original RGB images, because the image has a size of 1240×690[3] instead of 1920×1024.

[3] The NIR size is 1280×720, but it had a black-margin offset which has been trimmed, hence the new size of 1240×690

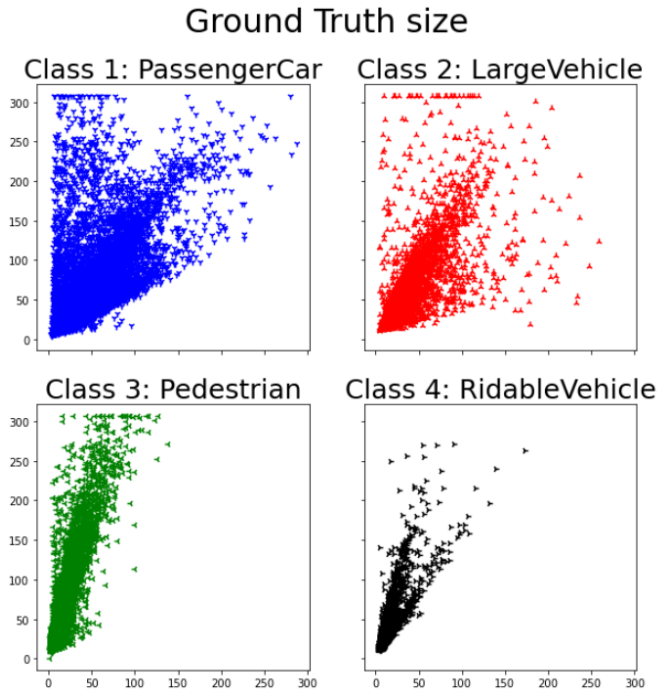


Figure 20: Ground truth sizes after homography
Source: self-made

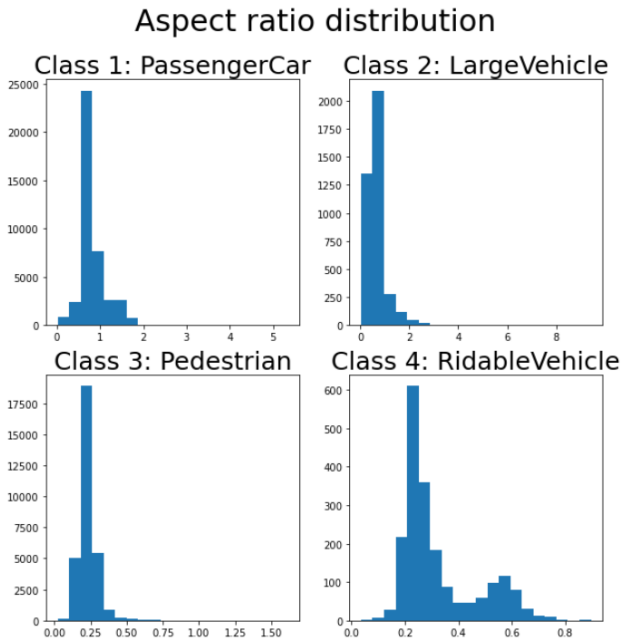


Figure 21: Aspect ratio distribution after homography
Source: self-made

4.4. Setting the hyperparameters

The model has been trained without using any kind of image pre-processing, the images have been taken from the dataset directly as they are. The following data augmentation techniques have been applied: random change of brightness, contrast and hue, but without adding new images, so the class distribution remains unaltered.

To set the hyperparameters several trials have been attempted, mainly regarding changes of the aspect ratios and the steps, i.e. aspect of the bounding boxes and the number of pixels between the centre of two consecutives boxes, respectively. This activity has been one of the most time-consuming tasks of the project, as with the available hardware the training time of the model, even with GPU, took more than 24 h.

Finally, the values have been settled as follows.

```
aspect_ratios = [[1.0, 2.0, 0.5, 3.0, 1.0/3.0, 0.25, 0.125],
                 [1.0, 2.0, 0.5, 3.0, 1.0/3.0, 0.25, 0.125],
                 [1.0, 2.0, 0.5, 3.0, 1.0/3.0, 0.25, 0.125],
                 [1.0, 2.0, 0.5, 3.0, 1.0/3.0, 0.25, 0.125],
                 [1.0, 2.0, 0.5, 3.0, 1.0/3.0, 0.25, 0.125],
                 [1.0, 2.0, 0.5, 3.0, 1.0/3.0, 0.25, 0.125]]
steps = [8, 16, 32, 64, 100, 300]
scales = [0.1, 0.2, 0.37, 0.54, 0.71, 0.88, 1.05]
```

Figure 22: Hyperparameters for the SSD300 model
Source: self-made

The Adam function has been used as optimizer, instead of the SGD, because it was too unstable. The learning rate value has been settled to 0.001, and its value have been reduced to one tenth once the model had stopped improving the validation loss. The number of epochs has been settled to 400, introducing an early stopping with 50 epochs non-improvement margin.

This model has more bounding boxes as the original one, hence the expected FPS will be lower as more bounding boxes will be used for the object detection, concretly with this hyperparameters a total of 13580 different bounding boxes will be detected for each class, quite above the original SSD300 threshold. But even with this bounding box increase it is thought the 24-FPS-threshold can be exceeded.

The anchor box neutral size is a square with 300×300 pixels, and the scale ratio of the first detection layer is 0.1, meaning the first bounding box will be a square with a size of 30×30 pixels, then the aspect ratios will be applied to increase the number and the forms of the bounding box.

4.5. Analysing the contribution of each sensor

4.5.1. The RGB

The first RGB model has been trained using the original RGB images, i.e. without the homography transformation, and resizing the images from 1920×1024 to 300×300. The pipeline structure can be seen in the next image.

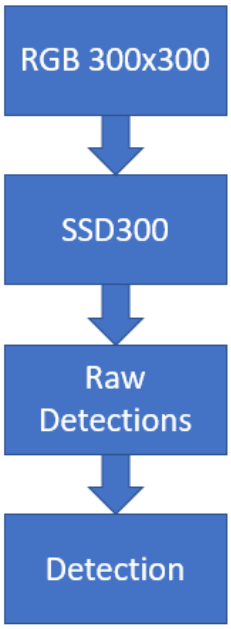


Figure 23: RGB-only SSD300 model pipeline
Source: self-made

With the hyperparameters shown in Figure 22 the model obtained the best loss values at the 339th epoch:

- Best epoch: 339 of 389
- Train loss: 3.132
- Validation loss: 3.217

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	59.03	61.20	66.22	62.38	65.79	65.24	58.58	62.88	**62.67**
Large vehicle	41.00	37.09	40.63	36.69	47.23	28.18	37.55	40.03	**38.55**
Pedestrian	55.20	58.23	10.14	55.94	35.99	47.67	59.17	58.55	**47.61**
Ridable vehicle	34.59	15.71	0.00	28.72	52.33	3.10	22.29	24.18	**22.62**
mAP	**47.46**	**43.06**	**29.25**	**45.93**	**50.33**	**36.05**	**44.40**	**46.41**	**42.86**

Table 7: AP and mAP results of the RGB SSD300 model
Source: self-made

The precision of the classes follows suspiciously the same order as the class distribution shown in Figure 12. The detection of the cars scores the best precision values, while the ridable vehicle class mainly obtains the lower precisions, with some weather-exception. From the results is observable, this model generalises perfectly, as it can detect objects in unseen weather conditions and even scoring better results on adversarial weather conditions than on clear weather. One of the explanations the researchers of the DENSE dataset guessed [5], related the better precisions with the lower number of objects present during night scenes or foggy situations. Nevertheless, the precision values are below those

obtained by the researches and shown in Table 5 (the image-only SSD) even taking into consideration the hard-weather condition only, the obtained precision still relies seven to ten points below the researchers' model.

4.5.2. The NIRs

The NIR model has been trained three times, each of them using a different depth-focus, i.e. near, intermediate and distant, it follows the same pipeline structure as shown in Figure 23. As each NIR illuminates different objects is expected to obtain different precision values. The resulted train and validation loss differ to each other too, as shown in the next table.

	Near-NIR	*Inter-NIR*	*Distant-NIR*
Best epoch	318 of 368	363 of 400	399 of 400
Train loss	3.588	3.471	3.869
Validation loss	3.714	3.547	3.788

Table 8: Train parameters for NIR images
Source: self-made

The three NIR precisions are shown in the next tables: the near in the Table 9, the intermediate precision in Table 10 and the distant-NIR results in Table 11.

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	50.93	50.94	62.02	51.58	68.18	61.90	52.15	51.53	**56.15**
Large vehicle	37.96	39.99	41.60	39.15	48.78	19.68	35.52	38.04	**37.59**
Pedestrian	46.91	45.19	12.80	42.72	41.35	62.30	43.71	42.04	**42.13**
Ridable vehicle	42.40	28.92	2.78	40.58	41.83	23.43	31.22	33.06	**30.53**
mAP	**44.55**	**41.26**	**29.80**	**43.51**	**50.03**	**41.83**	**40.65**	**41.17**	**41.60**

Table 9: AP and mAP results of the near-NIR SSD300 model
Source: self-made

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	54.48	57.64	64.73	59.17	71.52	68.54	55.60	59.29	**61.37**
Large vehicle	33.19	47.18	38.58	46.94	46.55	26.59	34.37	45.93	**39.92**
Pedestrian	47.27	51.58	15.01	50.56	42.60	66.87	44.90	47.44	**45.78**
Ridable vehicle	45.78	26.55	0.00	48.45	48.27	20.00	29.05	31.36	**31.18**
mAP	**45.18**	**45.74**	**29.58**	**51.28**	**52.23**	**45.50**	**40.98**	**46.01**	**44.56**

Table 10: AP and mAP results of the intermediate-NIR SSD300 model

Source: self-made

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	51.69	53.25	60.71	54.13	68.64	62.94	52.55	54.80	**57.34**
Large vehicle	32.71	42.27	37.39	34.78	43.80	19.65	31.60	40.03	**35.28**
Pedestrian	42.13	36.07	7.85	30.92	28.25	13.02	38.79	28.18	**28.15**
Ridable vehicle	32.99	7.34	0.00	29.02	37.89	8.90	18.55	8.43	**17.89**
mAP	**39.88**	**34.73**	**26.49**	**37.21**	**44.65**	**26.13**	**35.37**	**32.86**	**34.67**

Table 11: AP and mAP results of the distant-NIR SSD300 model
Source: self-made

The distant NIR was still improving before it achieved the maximal epoch-threshold of 400 and the train and validation loss lye above the other NIRs, hence is not a coincidence it shows the worst precision values of the three NIR, as it needed more time to train properly.

The intermediate NIR show the best precision values between the three NIR divisions, even better values than the previous RGB model are obtained, especially in night scenes and within the underrepresented classes. The ridable vehicle class, for example, obtains between 10 and 20 points above the RGB precision of Table 7.

Comparing the results with those obtained by the researchers “gated-only SSD” model at Table 5, the intermediate-NIR equals the values of the hard-scenes and even those from the moderate ones in some weather circumstances.

4.6. Precision improvement

The mAP values previously shown look quite disappointing, especially those of the RGB spectrum, as they lye quite below the obtained precisions of the single-sensors models.

There are some ways to improve the precision rate before starting with the fusion techniques: increasing the number of epochs, artificially increasing the size of the dataset to balance the class distribution, adding more bounding boxes, using the SSD512 instead of the SSD300, and so on, all those methods could increment the final precision of the model. In this section they will be checked and commented.

4.6.1. Increasing the input shape: SSD512

From the researchers’ paper is known the SSD algorithm is used, but not if is a SSD300, a SSD512 or even an *ad hoc* SSD model with a bigger input shape. An increment of the algorithm input image shape decreases

the FPS, hence the model could eventually not work in real time conditions as wished.

The SSD512 model has an extra detection layer, therefore the total number of detected bounding boxes per class increase too, as the training time do. The model kept improving until the 400th epoch, obtaining a train loss of 2.682 and a validation loss of 2.812.

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	60.43	62.36	67.45	63.13	70.04	66.10	61.26	64.53	**64.41**
Large vehicle	48.78	49.71	42.78	46.62	48.65	48.57	44.02	51.37	**47.56**
Pedestrian	62.58	65.35	19.93	61.06	48.97	55.65	66.37	65.01	**55.62**
Ridable vehicle	46.78	30.53	1.81	48.57	60.26	21.51	22.46	32.14	**33.01**
mAP	**54.64**	**51.99**	**32.99**	**54.85**	**56.98**	**47.96**	**48.53**	**53.26**	**50.15**

Table 12: AP and mAP results of the RGB SSD512 model
Source: self-made

The precision results of the previous table are clearly above those from Table 7 with the SSD300 model, and the difference would have been even bigger, if more time had been given to the model to be properly trained.

4.6.2. Reducing the FoV

Another possibility to increase the precision might be related to the object deformation due to the image reduction, using a smaller image, like the one obtained after applying the homography, instead of the original 1920×1024 image could increase the precision too. Not only because the reduction of the objects is less severe, as it has been seen in the chapter 4.3, but because less objects are visible due the field of view reduction.

This model has been trained again with a SSD300 using the images resulting of the homography, meaning with an input image shape of 1240×690. From now on, all the models will use this image size as the standard input (and then applying the 300×300 resizing).

This model obtained the best results at the 240th epoch, early stopping 50 epochs later, and it obtained a train loss of 3.275 and a validation loss of 2.523.

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	56.41	58.14	66.18	60.47	67.50	68.52	57.12	61.89	**62.03**
Large vehicle	44.08	46.83	42.39	45.05	55.83	61.20	42.73	45.06	**47.90**

Pedestrian	51.07	53.62	6.87	50.00	39.80	61.00	55.38	51.96	**46.21**
Ridable vehicle	42.21	18.74	0.00	45.82	33.29	15.83	25.42	22.07	**25.42**
mAP	**48.44**	**44.33**	**28.86**	**50.33**	**49.11**	**51.64**	**45.16**	**45.24**	**45.39**

Table 13: AP and mAP results of the RGB FoV SSD300 model
Source: self-made

Generally, this model shows a better precision results than those with the original 1920×1024 image size, hence the FoV reduction seems a good method to increase the model precision.

4.6.3. Other options

There are other methods which could increase the precision, one of them is the increase of the number of epochs, a clear example is the model SSD512 and the distant-NIR SSD300, which were still improving just before the 400-epoch threshold was achieved. The other models never shown a sign of overfitting, hence a combination of an increment of epochs and a reduction of the learning rate might have helped.

Adding more bounding boxes (via increasing the number of aspect ratios or reducing the number of pixels between two consecutive bounding boxes) could affect positively the precision while reducing the FPS too.

Artificially increasing the number of images, mainly adding more objects of the underrepresented classes could have a positive effect on the precision value of those classes. Other options would be incrementing the model depth, increasing the number of the detection layers, or even using another network backbone instead of the chosen VGG16. Still, none of these methods will be applied as they are out of the scope of the thesis.

4.7. Pixel Fusion

4.7.1. RGB and gradient magnitude (M)

Inspired by the papers which focus on pedestrian detection, like [27, 30] it was decided to use more channels to increase the information given to the model, as an easy way of implementing the pixel fusion.

The first step consists in adding a fourth channel to the RGB image, called the gradient magnitude. This new channel returns the silhouette of the objects, obtained calculating the pixel derivative in both axis. A bigger slope is produced by a bigger pixel-colour change, hence is likely to be related to the end of an object and the beginning of another one.

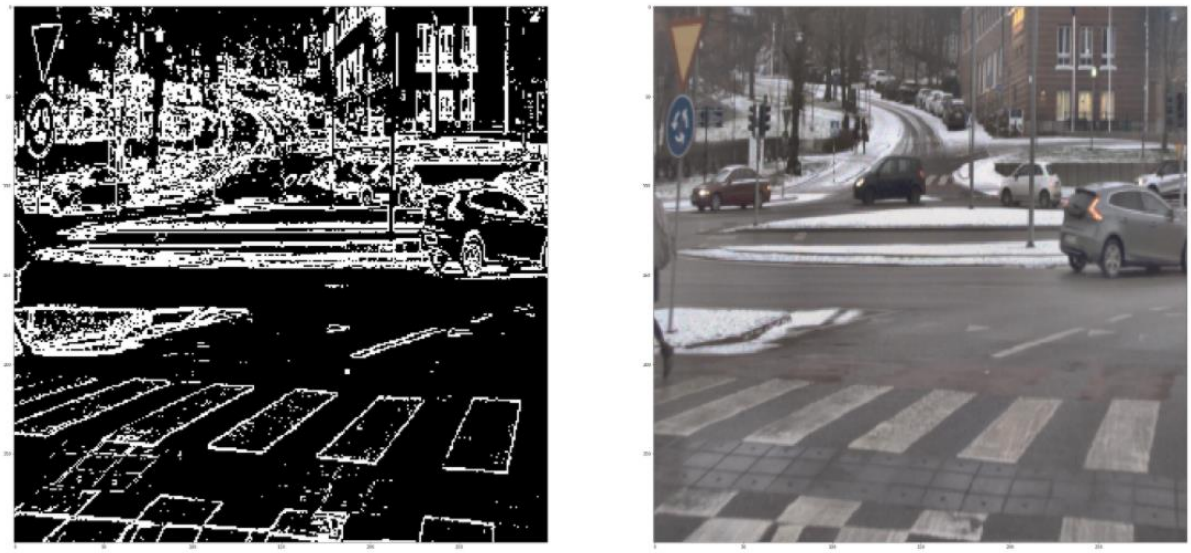


Figure 24: Gradient magnitude (left), corresponding RGB image (right)
Source: self-made with a ground image from [4]

The pipeline structure of this model differs from the one shown in Figure 23 because of the channel-merging extra step.

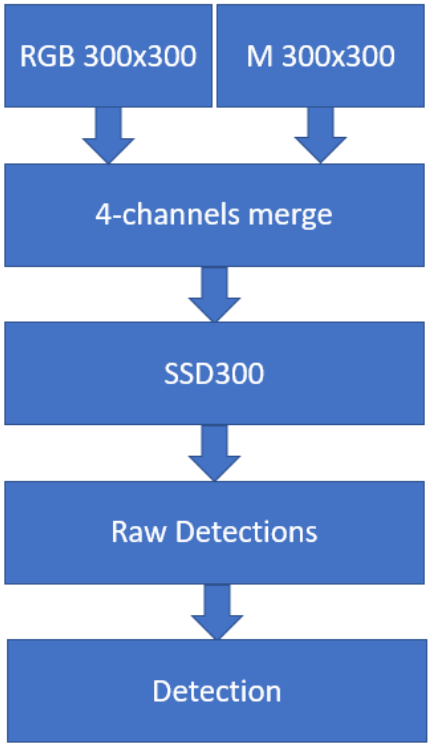


Figure 25: RGB + M FoV SSD300 model pipeline
Source: self-made

This model started overfitting at the 59th epoch and obtained a train loss of 3.145 and a validation loss of 5.450.

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	52.42	50.05	55.95	54.62	51.24	48.19	49.14	52.97	**51.82**

Large vehicle	25.84	14.32	14.88	13.92	21.29	3.20	19.32	17.64	**16.30**
Pedestrian	41.09	41.38	5.97	32.89	13.97	22.58	43.67	39.30	**30.11**
Ridable vehicle	20.00	2.87	0.03	10.82	19.20	0.50	6.45	5.71	**8.20**
mAP	**34.84**	**27.16**	**19.21**	**28.06**	**26.42**	**18.62**	**29.64**	**28.90**	**26.61**

Table 14: AP and mAP results of the RGB + M FoV SSD300 model
Source: self-made

It seems that this fourth channel has only added noise to the model and the precision has consequently dropped. As the obtained precision has not given the expected results, this line of investigation will be abandoned, and no more gradient-related channels will be added. In the following image it can be seen why the precision has dropped so much: this model sees incorrectly cars in the bushes at the left part of the image.

Figure 26: Output of the RGB + M Pixel Fusion model
Source: self-made with a ground image from [4]

4.7.2. The three NIRs

The second pixel fusion model is based on the fusion of the three NIR images, i.e. the near, intermediate and distant, its pipeline structure can be seen in the Figure 27, and is similar to the RGB + M pipeline. The main reason of this pixel fusion model is the creation of a model which sees the three ranges at the same time.

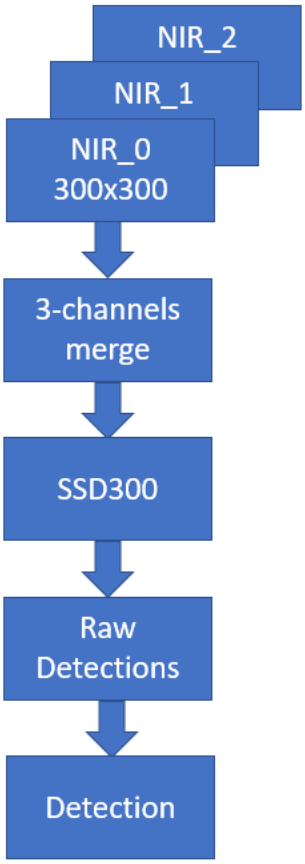


Figure 27: NIR FoV Pixel Fusion SSD300 model pipeline
Source: self-made

Each NIR input is in grayscale, so mathematically it can be described as one single channel. The fusion of the three NIR distances produce a three-channel image and is equivalent to assigning the red channel to the near-NIR, the green channel to the intermediate-NIR and the blue channel to the distant-NIR. Therefore, the output image will be grayscaled in those regions visible in the three channels but will be coloured if its pixels derived from only one or two of the NIR channels, being the colour red, blue, green or a combination of two of them depending of the input channel.

The model obtains the best results in the 191th epoch, with a train loss of 3.941 and a validation loss of 2.862.

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	51.67	55.22	63.71	57.33	70.30	68.46	53.58	58.30	**59.82**
Large vehicle	36.18	57.21	47.73	46.18	48.74	31.28	35.61	49.68	**44.08**
Pedestrian	42.06	47.51	9.68	48.89	34.62	57.23	37.45	42.16	**39.95**
Ridable vehicle	41.01	21.36	0.00	45.00	42.64	21.78	23.71	29.54	**28.13**
mAP	**42.73**	**45.33**	**30.28**	**49.35**	**49.08**	**44.69**	**37.59**	**44.92**	**43.00**

Table 15: AP and mAP results of the NIR FoV Pixel Fusion SSD300 model
Source: self-made

Except for the large vehicle class, the rest of the precisions lye below the obtained results of the NIR channels alone, so this fusion has not shown any improvement as expected.

4.7.3. RGB and NIR

An interesting approach is the fusion of both spectrums, the RGB and the three-channel NIR, to build a robust model which works in daylight and nightlight conditions, whose pipeline structure can be seen in the next image.

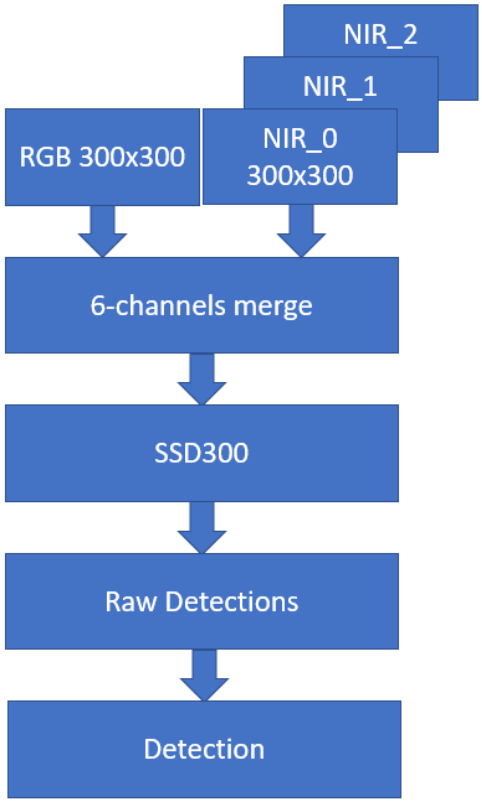


Figure 28: RGB+NIR FoV Pixel Fusion SSD300 model pipeline
Source: self-made

This model has an input of six channels, three from the RGB and three additional ones from the NIR and has obtained the best loss values in the 256th epoch, with a train loss of 2.878 and a validation loss of 2.372.

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	57.18	59.08	68.39	60.93	70.25	70.24	58.99	63.18	**63.53**
Large vehicle	43.80	47.30	34.42	39.80	50.26	45.93	37.53	43.05	**42.76**
Pedestrian	49.82	53.33	8.17	50.11	39.02	58.05	54.51	52.88	**45.74**
Ridable	38.39	13.33	0.00	36.84	51.22	18.84	21.81	22.68	**25.39**

vehicle									
mAP	**47.30**	**43.26**	**27.74**	**46.92**	**52.69**	**48.26**	**43.21**	**45.45**	**44.35**

Table 16: AP and mAP results of the RGB+NIR FoV Pixel Fusion SSD300 model
Source: self-made

The results from the previous table show only an improvement with the passenger car class, almost equalling the value of the SSD512 model. The rest of the classes do not show a precision improvement. It is possible that the imperfect homography and the consequent pixel-offset has prevented the model to be trained to its full potential.

4.8. Sensor Fusion

The sensor fusion approach fusions the raw predictions of the models, i.e. the precision before the non-maximum suppression step is applied. Two models will be designed to check the quality of this method: the fusion of the three NIR channels and the fusion of the RGB and the NIR images.

4.8.1. NIRs

An alternative variant of the model explained in chapter 4.7.2 is the sensor fusion in which, instead of fusing the three NIR and feeding the model with this triple channel image, the precisions obtained separately by each model will be fused, as shown in the next figure.

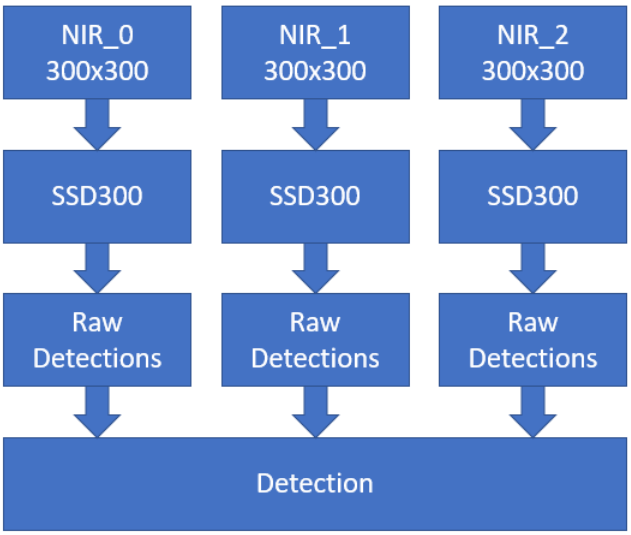


Figure 29: NIRs Sensor Fusion SSD300 model pipeline
Source: self-made

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	52.16	55.53	61.22	56.54	70.23	65.74	53.24	56.80	**58.93**
Large vehicle	38.62	52.08	46.04	52.18	51.75	30.18	40.33	51.79	**45.37**

Pedestrian	46.47	48.88	11.71	46.42	40.74	58.72	42.61	42.99	**42.32**
Ridable vehicle	46.53	30.91	1.19	49.91	46.03	25.97	32.12	33.61	**33.28**
mAP	**45.95**	**46.85**	**30.04**	**51.26**	**52.19**	**45.16**	**42.08**	**46.30**	**44.98**

Table 17: AP and mAP results of the NIRs Sensor Fusion SSD300 model
Source: self-made

This method has outperformed the NIR pixel fusion variant, seen in the chapter 4.7.2, as well as simple models with the separate NIR channels from chapter 4.5.2.

4.8.2. RGB and NIR

The sensor fusion of the RGB and NIR results, as an alternative to the pixel fusion model designed at chapter 4.7.3, show again a better precision rate compared to the pixel fusion variant. Its structure differs from the one shown in Figure 28.

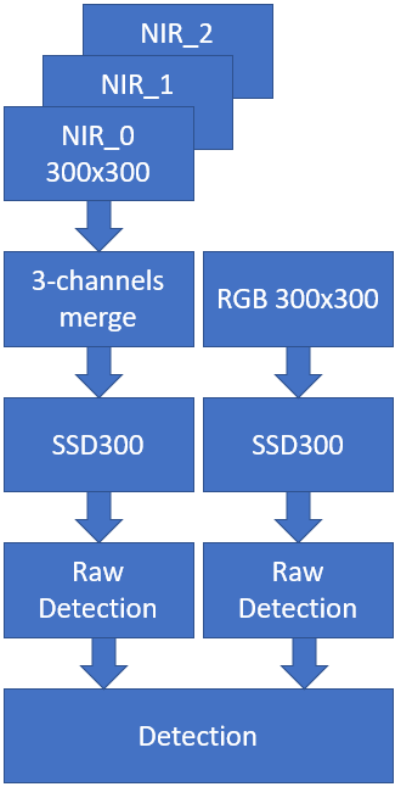


Figure 30: RGB+NIR FoV Sensor Fusion SSD300 model pipeline
Source: self-made

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	53.64	56.55	63.76	59.55	72.51	70.35	55.16	60.50	**61.50**
Large vehicle	43.86	59.97	50.91	49.73	53.53	57.10	42.02	52.67	**51.22**

Pedestrian	45.00	49.67	7.22	49.30	41.04	61.54	43.15	46.08	**42.88**
Ridable vehicle	45.24	24.47	0.00	48.94	32.68	21.81	28.14	30.92	**29.03**
mAP	**46.93**	**47.66**	**30.47**	**51.88**	**49.94**	**52.70**	**42.12**	**47.54**	**46.16**

Table 18: AP and mAP results of the RGB+NIR FoV Sensor Fusion SSD300 model
Source: self-made

The underrepresented classes, i.e. large and ridable vehicles, see its precision increased, while the other two classes loss its precision rate three points, but the general result remains positive.

4.9. Features Fusion

The third fusion method is the feature fusion. This version fuses the image spectrum, nor in the input adding more channels, neither in the output fusing the raw predictions, but inside the network. Two variations have been designed: the first one, called halfway fusion, fuse the features in the intermediate layers of the network, while the second model, called late fusion, fuse high-level features in the deeper layers. Its pipeline structure can be seen in the next image and it defines both, halfway and late fusion.

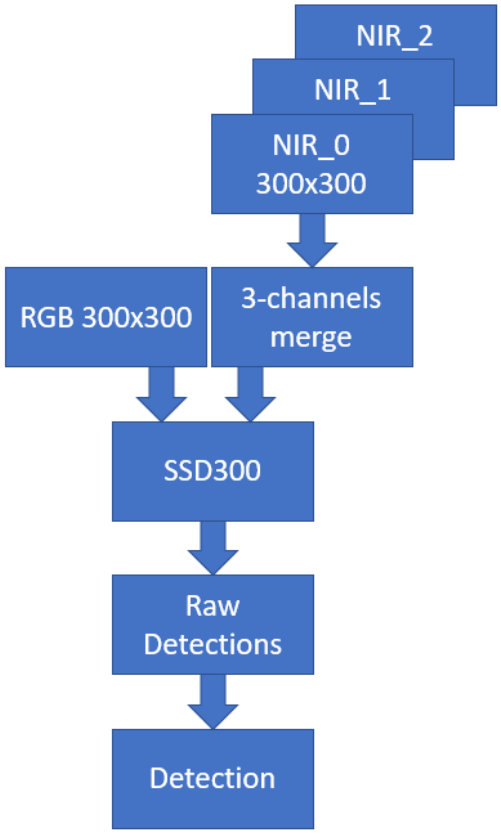


Figure 31: RGB+NIR FoV Feature Fusion SSD300 model pipeline
Source: self-made

4.9.1. Halfway Fusion

The SSD network receives two streams, the first belonging to the RGB pipeline and the second to the NIR images. The first layers are duplicated until both streams are concatenated and its features fused, subsequently the network follows the same architecture as the original SSD300, with only one stream. The model architecture can be seen in the appendix in the Figure 37.

This model obtains its best precision during the 283th epoch, with a train loss of 2.963 and a validation loss of 2.237.

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	56.09	58.60	67.18	61.01	69.71	70.18	58.61	62.62	**63.00**
Large vehicle	41.95	44.97	32.99	40.41	49.84	54.53	36.24	43.68	**43.08**
Pedestrian	49.69	53.53	8.51	49.14	38.26	59.16	55.02	51.87	**45.65**
Ridable vehicle	42.03	18.31	7.03	39.89	44.71	11.07	22.13	24.08	**26.16**
mAP	**47.44**	**43.85**	**28.93**	**47.61**	**50.63**	**48.74**	**43.00**	**45.56**	**44.47**

Table 19: AP and mAP results of the RGB+NIR FoV Halfway Fusion SSD300 model
Source: self-made

The obtained precisions are equivalent with those obtained using other fusion methods outstanding specially in the passenger car's class detection.

4.9.2. Late Fusion

To reduce the influence of the homography offset between the RGB and NIR images, a second feature fusion model has been designed with a later feature fusion. The idea behind is to fuse the map features at "a stage where spatial information is less relevant" [29]. Another reason to try this variation is to study the consequences of adding a new detection layer. As the stream concatenation is placed after the fifth convolutional block and the output of the fourth concatenation block is used to feed the first detection layer, this model will have a specific detection layer per each stream, while the other layers will be feed with the result of the feature concatenation of both streams. The model diagram can be seen in the appendix as Figure 39.

This model obtains its best precision during the 273th epoch, with a train loss of 4.155 and a validation loss of 3.708, clearly above the previous designed models.

	CLEAR		*LIGHT FOG*		*DENSE FOG*		*SNOW*		
	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	DAY	NIGHT	*AP*
Passenger car	46.05	45.82	57.11	46.54	46.72	58.42	47.55	50.83	**49.88**

Large vehicle	11.86	19.29	11.79	21.44	8.20	35.84	18.16	16.52	**17.89**
Pedestrian	44.62	47.14	17.28	46.36	33.12	30.81	49.26	47.44	**39.50**
Ridable vehicle	15.32	12.12	11.65	21.38	3.17	17.62	6.05	9.64	**12.12**
mAP	**29.46**	**31.09**	**24.46**	**33.93**	**22.80**	**35.67**	**30.25**	**31.11**	**29.85**

Table 20: AP and mAP results of the RGB+NIR FoV Late Fusion SSD300 model
Source: self-made

The designed late-fusion model has not produced the expected precision improvement, the one-spectrum-only detection layer seems to have boycott the detection performance of the whole algorithm.

5. Results

This chapter summarises the results seen in the previous chapter. It also provides relevant information regarding the model training time and its detection FPS value.

Model name	*Epoch training time*	*Best epoch*	*Train loss*	*Validation loss*
RGB	170 s	339	3.132	3.217
Near-NIR	145 s	318	3.588	3.714
Intermediate-NIR	145 s	363	3.471	3.547
Distant-NIR	145 s	399	3.869	3.788
RGB SSD512	510 s	400	2.682	2.812
RGB FoV	155 s	240	3.275	2.523
Pixel Fusion RGB+M	110 s	59	3.145	5.450
Pixel Fusion NIR	150 s	191	3.941	2.862
Pixel Fusion RGB+NIR	180 s	256	2.878	2.372
Sensor Fusion NIR	-	-	-	-
Sensor Fusion RGB+NIR	-	-	-	-
Halfway Fusion RGB+NIR	320 s	283	2.963	2.237
Late Fusion RGB+NIR	430 s	273	4.155	3.708

Table 21: Model summary
Source: self-made

There is a significant difference within the value of the training time per epoch and the network architecture, as well as the image input, plays an important role, as it can be seen in the Table 21. The model SSD512 needs the triple of time per epoch to be trained compared to its SSD300 counterpart; more complex architectures, like those with feature fusion, needs the double of time to be trained compared to a model with pixel fusion. Therefore, it can be seen how time-consuming the training of the model can be and why is so important to use a GPU to accelerate the calculations.

In the next table the AP, mAP and FPS of the model will be shown.

Model name	*Passenger car*	*Large vehicle*	*Pedestrian*	*Ridable vehicle*	*mAP*	*FPS*
RGB	62,67	38,55	47,61	22,62	42,86	33
Near-NIR	56,15	37,59	42,13	30,53	41,60	33
Intermediate-NIR	61,37	39,92	45,78	31,18	44,56	36
Distant-NIR	57,34	35,28	28,15	17,89	34,67	36
RGB SSD512	64,41	47,56	55,62	33,01	50,15	14

RGB FoV	62,03	47,90	46,21	25,42	45,39	33
Pixel Fusion RGB+M	51,82	16,30	30,11	8,20	26,61	28
Pixel Fusion NIR	59,82	44,08	39,95	28,13	43,00	43
Pixel Fusion RGB+NIR	63,53	42,76	45,74	25,39	44,35	24
Sensor Fusion NIR	58,93	45,37	42,32	33,28	44,98	10
Sensor Fusion RGB+NIR	61,50	51,22	42,88	29,03	46,16	17
Halfway Fusion RGB+NIR	63,00	43,08	45,65	26,16	44,47	32
Late Fusion RGB+NIR	49,88	17,89	39,50	12,12	29,85	24

Table 22: AP, mAP and FPS summary
Source: self-made

The model which achieves the higher precision among all the classes is the one trained with the SSD512 architecture and using only the RGB spectrum as input. But this model has a low FPS rate of only 14 FPS, clearly below the desired threshold of 24 frames per second to habilitate the model to work on real time. Other models with high precision rates like the sensor fusion variations have also an FPS below the 24 FPS threshold.

The passenger car class is with huge difference the class with the better detection precision, there are mainly two possible reasons: the first one is because is the most represented class of the dataset, so the model has been able to generalise and understand how a car looks like, and the second possibility is its aspect ratio is more homogenous. If data augmentation techniques to increase artificially the representation of the other classes were used, the precision of them would increase too. Therefore, instead of comparing the mAP value of all classes it will be used the AP value of the passenger car class to determine with which model the best results are obtained.

Ignoring those models with an FPS below the real-time threshold, the detection algorithms with the higher precision are the pixel fusion method and the feature fusion with halfway fusion, both using the RGB and NIR spectrums, with an AP above the 63%. To calculate the FPS value of the pixel fusion model the pre-processing step, i.e. the merging of the channels, has been ignored, so its value is in reality lower than those shown in Table 22. The halfway fusion version, however, seems to be the most promising method, because with an FPS rate of 32, there is still some margin to apply precision-improving techniques, like increasing the number of bounding boxes and/or the image input shape, two methods known known to affect positively the precision rate, until the 24 FPS-threshold is obtained.

In the next image can be seen how the halfway-fusion model detects correctly almost all objects of the image in a nightlight urban scene with many objects.

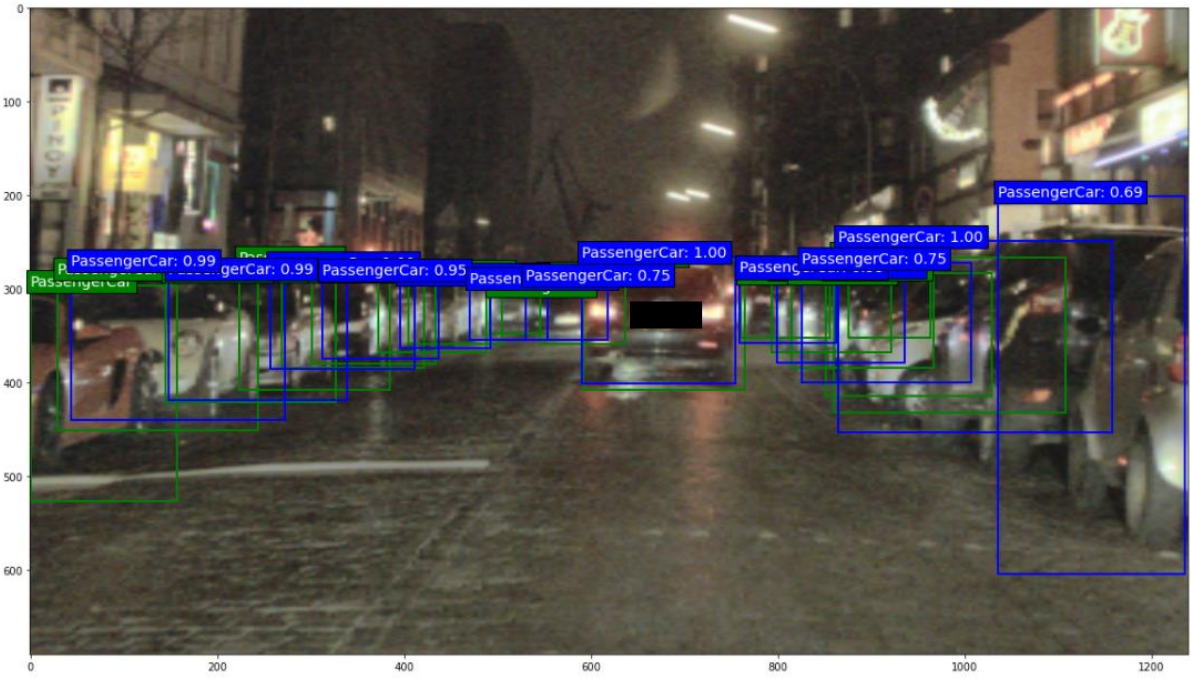


Figure 32: Ground truths (green boxes) and detections (blue boxes) of the Halfway Fusion model
Source: self-made with a ground image from [4]

6. Conclusions and future work

During this thesis an image-only object detection algorithm has been designed. This algorithm could be used on real-time automated driving, no matter the weather conditions experienced, a nice surprise considering it has been trained with clear weather images only. Three fusion techniques have been applied to study the impact in the precision rate.

The models designed following the sensor fusion approach achieve the best prediction results but are unable to work on real time conditions, as they need more time to process and fuse the predictions. Pixel fusion and feature fusion models (with halfway fusion) show a similar precision rate, but the last method detects faster.

The precision rates remain below those obtained by the researchers but ignoring all the hyperparameters used by the researchers' model, like the FPS rate or even the input image of the SSD algorithm, a comparison between both models is impossible. The impossibility of using the FIR channel had possibly had a negative impact on the final precisions as well, as only two of the three image spectrums were used (while the researchers' approach used the sensor data of four different sensors).

During this project it has been noticed the importance of a well-balanced class distribution of the dataset in order to allow the model to fully learn the features of the objects. In that case, the classes with a lower representation have usually obtained a lower precision rate than the overrepresented classes.

As a logical next step for this project is the proposal of an artificial increment of the dataset to balance all classes, this task is expected to increase the precision rate. The setting of the hyperparameters and the model training were the most time-consuming task, as all models were trained from scratch, hence it has been impossible to train all the desired architectures. Therefore, another step could be to try other feature fusion architectures, for example placing the streams-concatenate layer in other positions and experimenting with the deletion of the NiN layer.

7. Glossary

Anchor box: predefined bounding box used by the detection algorithm to detect objects in an image in specific locations
Aspect ratio: width-high image rate
AP: Average Precision
Bounding box: box surrounding an object in an image
FIR: Far Infra-Red
FN: False Negative
FoV: Field of view
FP: False Positive
FPS: Frame per Second
GPU: Graphics Processing Unit
gt: ground truth, input label to train a detection algorithm, it provides the object location and its class assignment
LIDAR: Laser Imaging Detection and Ranging
mAP: mean Average Precision
NiN: Network in Network
NIR: Near Infra-Red
RGB: Red-Green-Blue
SSD300: single shot detector with 300×300 image shape input
SSD512: single shot detector with 512×512 image shape input
TP: True Positive

8. Bibliography

[1] KITTI Dataset. Available: http://www.cvlibs.net/datasets/kitti/
[2] NuScenes Dataset. Available: https://www.nuscenes.org/
[3] Udacity Dataset. Available: https://github.com/udacity/self-driving-car
[4] DENSE Dataset. Available: https://www.uni-ulm.de/en/in/driveu/projects/dense-datasets/
[5] M. Bijelic *et al.*, "Seeing Through Fog Without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather" *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, Accessed: Dec. 06, 2020
[6] M. Elgendy, *Deep Learning for Vision Systems*. Manning Publications, 2020
[7] Pierluigi Ferrari Github Repository. Available: https://github.com/pierluigiferrari/ssd_keras
[8] Rafael Padilla Github Repository. Available: https://github.com/rafaelpadilla/Object-Detection-Metrics
[9] Colour Spaces explanation. Available : http://www.huevaluechroma.com/093.php
[10] Colour Theory explanation. Available: https://en.wikipedia.org/wiki/Color
[11] Infrared explanation. Available: https://www.edmundoptics.de/knowledge-center/application-notes/optics/the-correct-material-for-infrared-applications/
[12] DENSE Dataset registration form. Available: https://www.uni-ulm.de/en/in/driveu/projects/dense-datasets/dense-registration-form/
[13] T. S. Huang, «Computer Vision: Evolution and Promise,» 1996.
[14] N.S.Manikandan, K.Ganesan. "Deep Learning Based Automatic Video Annotation Tool for Self-Driving Car", 2019
[15] Z. Zhao, P. Zheng, S. Xu, X. Wu. "Object Detection with Deep Learning. A Review", 2019
[16] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, R. Qu. "A Survey of Deep Learning-based Object Detection", 2019
[17] Karen Simonyan, Andrew Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014
[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition, 2015
[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", in CVPR, 2016.
[20] A. Bochkovskiy, C. Wang, H. Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection", 2020
[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector", in ECCV, 2016
[22] R. Girshick, J. Donahue, T. Darrell, J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation,", 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587, June 2014.
[23] R. Girshick, "Fast r-cnn," in 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448, Dec 2015.
[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, pp. 1137–1149, June 2017.

[25] PASCAL VOC. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2007/
[26] K. Yoneda, N. Suganuma, R. Yanase, M. Aldibaja. “Automated driving recognition technologies for adverse weather conditions”, 2019
[27] D. König, M. Adam, C. Jarvers, G. Layher, H. Neumann, M. Teutsch. “Fully convolutional region proposal networks for multispectral person detection” 2017
[28] J. Liu, S. Zhang, S. Wang, D. N. Metaxas. “Multispectral deep neural networks for pedestrian detection” 2016
[29] J. Wagner, V. Fischer, M. Herman, S. Behnke. “Multispectral Pedestrian Detection using Deep Fusion Convolutional Neural Networks” 2016
[30] S. Hwang, J. Park, N. Kim, Y. Choi, I. S. Kweon “Multispectral pedestrian detection: Benchmark dataset”, 2015
[31] Y. Caoa, D. Guanb, W. Huangc, J. Yanga, Y. Caoa, Y. Qiao. “Pedestrian detection with unsupervised multiespectral feature learning using deep neural networks” 2019
[32] J. Cao, Y. Pang, J. Xie, F. S. Khan, L. Shao. “From Handcrafted to Deep Features for Pedestrian Detection: A Survey”, 2020
[33] M. Lin, Q. Chen, S. Yan. “Network in Network”, 2014
[34] M. Bijelic et al., “Seeing Through Fog Without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather (Supplemental Material)” The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, Accessed: Dec. 06, 2020
[35] Guillem Tudela Github Repository. Available: https://github.com/GuillemTD/SeeingThroughFog_HalfwayFusion
[36] Seeing Through Fog Github Repository. Available: https://github.com/princeton-computational-imaging/SeeingThroughFog/tree/master/splits

9. Appendix

9.1. Class distribution original RGB

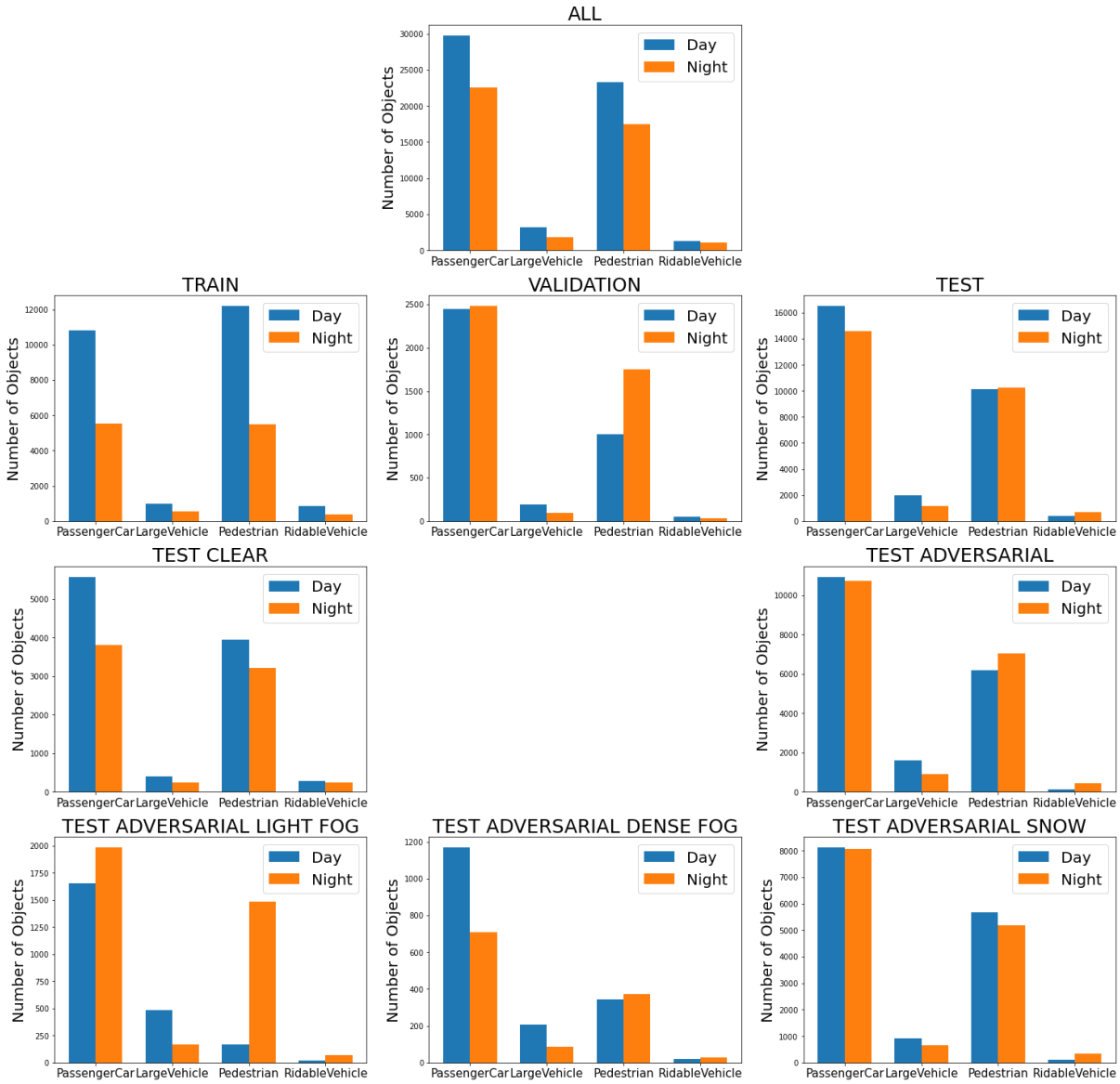


Figure 33: Object distribution in RGB images
Source: self-made

9.2. Class distribution original NIR

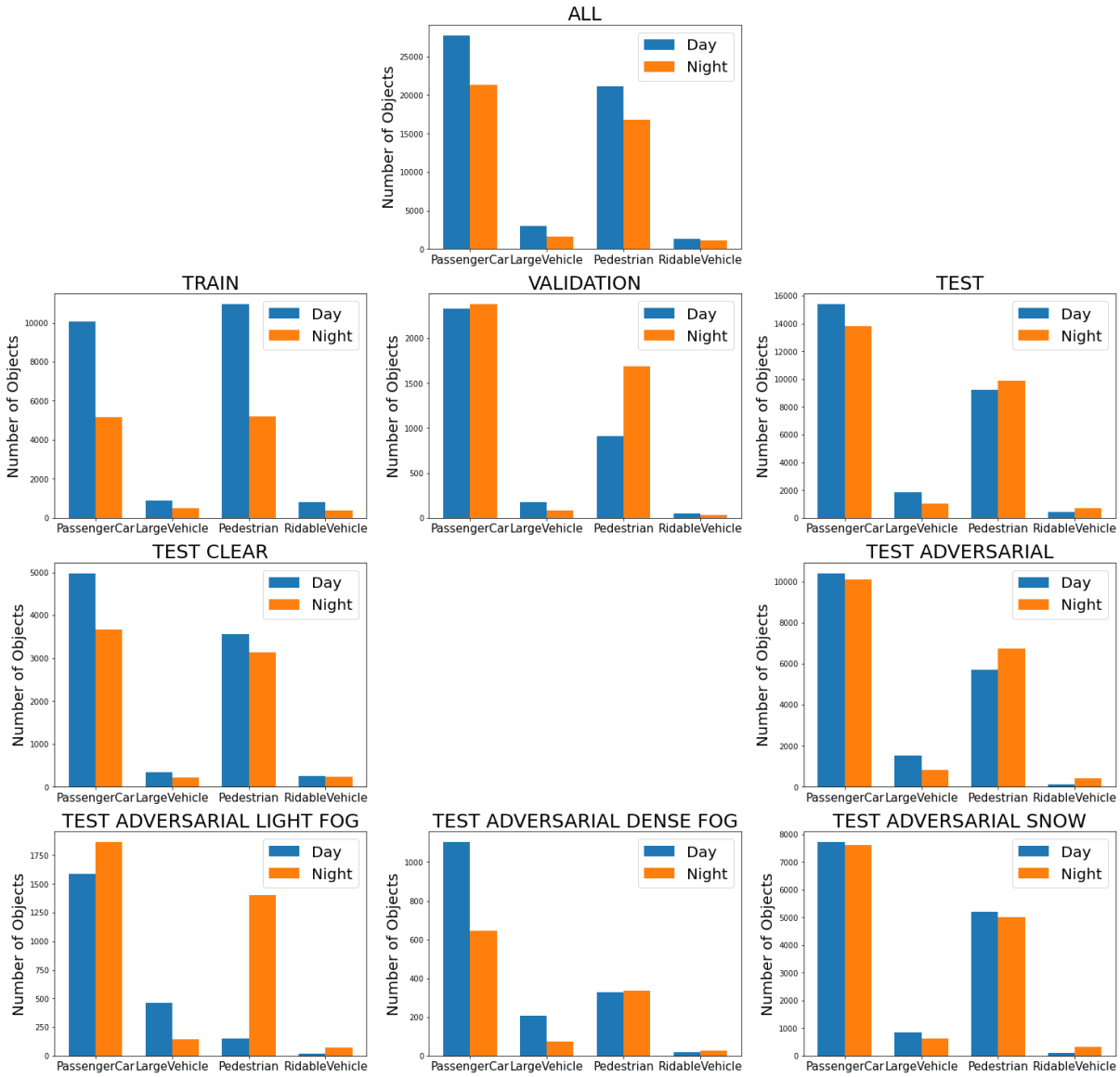


Figure 34: Object distribution in NIR images
Source: self-made

9.3. Model SSD300

```
Model: "SSD300"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 300, 300, 3)       0
_________________________________________________________________
conv1_1 (Conv2D)             (None, 300, 300, 64)      1792
_________________________________________________________________
conv1_2 (Conv2D)             (None, 300, 300, 64)      36928
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 150, 150, 64)      0
_________________________________________________________________
conv2_1 (Conv2D)             (None, 150, 150, 128)     73856
_________________________________________________________________
conv2_2 (Conv2D)             (None, 150, 150, 128)     147584
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 75, 75, 128)       0
_________________________________________________________________
conv3_1 (Conv2D)             (None, 75, 75, 256)       295168
_________________________________________________________________
conv3_2 (Conv2D)             (None, 75, 75, 256)       590080
_________________________________________________________________
conv3_3 (Conv2D)             (None, 75, 75, 256)       590080
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 38, 38, 256)       0
_________________________________________________________________
conv4_1 (Conv2D)             (None, 38, 38, 512)       1180160
_________________________________________________________________
conv4_2 (Conv2D)             (None, 38, 38, 512)       2359808
_________________________________________________________________
conv4_3 (Conv2D)             (None, 38, 38, 512)       2359808
_________________________________________________________________
pool4 (MaxPooling2D)         (None, 19, 19, 512)       0
_________________________________________________________________
conv5_1 (Conv2D)             (None, 19, 19, 512)       2359808
_________________________________________________________________
conv5_2 (Conv2D)             (None, 19, 19, 512)       2359808
_________________________________________________________________
conv5_3 (Conv2D)             (None, 19, 19, 512)       2359808
_________________________________________________________________
pool5 (MaxPooling2D)         (None, 19, 19, 512)       0
_________________________________________________________________
fc6 (Conv2D)                 (None, 19, 19, 1024)      4719616
_________________________________________________________________
fc7 (Conv2D)                 (None, 19, 19, 1024)      1049600
_________________________________________________________________
conv8_1 (Conv2D)             (None, 19, 19, 256)       262400
_________________________________________________________________
conv8_padding (ZeroPadding2D (None, 21, 21, 256)       0
_________________________________________________________________
conv8_2 (Conv2D)             (None, 10, 10, 512)       1180160
_________________________________________________________________
conv9_1 (Conv2D)             (None, 10, 10, 128)       65664
_________________________________________________________________
conv9_padding (ZeroPadding2D (None, 12, 12, 128)       0
_________________________________________________________________
conv9_2 (Conv2D)             (None, 5, 5, 256)         295168
_________________________________________________________________
conv10_1 (Conv2D)            (None, 5, 5, 128)         32896
_________________________________________________________________
conv10_padding (ZeroPadding2 (None, 7, 7, 128)         0
_________________________________________________________________
conv10_2 (Conv2D)            (None, 3, 3, 256)         295168
_________________________________________________________________
conv11_1 (Conv2D)            (None, 3, 3, 128)         32896
_________________________________________________________________
conv11_padding (ZeroPadding2 (None, 5, 5, 128)         0
_________________________________________________________________
conv11_2 (Conv2D)            (None, 2, 2, 256)         295168
=================================================================
Total params: 22,943,424
Trainable params: 22,943,424
Non-trainable params: 0
```

Figure 35: SSD300 model representation
Source: self-made

9.4. Model SSD512

```
Model: "SSD512"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 512, 512, 3)       0
_________________________________________________________________
conv1_1 (Conv2D)             (None, 512, 512, 64)      1792
_________________________________________________________________
conv1_2 (Conv2D)             (None, 512, 512, 64)      36928
_________________________________________________________________
pool1 (MaxPooling2D)         (None, 256, 256, 64)      0
_________________________________________________________________
conv2_1 (Conv2D)             (None, 256, 256, 128)     73856
_________________________________________________________________
conv2_2 (Conv2D)             (None, 256, 256, 128)     147584
_________________________________________________________________
pool2 (MaxPooling2D)         (None, 128, 128, 128)     0
_________________________________________________________________
conv3_1 (Conv2D)             (None, 128, 128, 256)     295168
_________________________________________________________________
conv3_2 (Conv2D)             (None, 128, 128, 256)     590080
_________________________________________________________________
conv3_3 (Conv2D)             (None, 128, 128, 256)     590080
_________________________________________________________________
pool3 (MaxPooling2D)         (None, 64, 64, 256)       0
_________________________________________________________________
conv4_1 (Conv2D)             (None, 64, 64, 512)       1180160
_________________________________________________________________
conv4_2 (Conv2D)             (None, 64, 64, 512)       2359808
_________________________________________________________________
conv4_3 (Conv2D)             (None, 64, 64, 512)       2359808
_________________________________________________________________
pool4 (MaxPooling2D)         (None, 32, 32, 512)       0
_________________________________________________________________
conv5_1 (Conv2D)             (None, 32, 32, 512)       2359808
_________________________________________________________________
conv5_2 (Conv2D)             (None, 32, 32, 512)       2359808
_________________________________________________________________
conv5_3 (Conv2D)             (None, 32, 32, 512)       2359808
_________________________________________________________________
pool5 (MaxPooling2D)         (None, 32, 32, 512)       0
_________________________________________________________________
fc6 (Conv2D)                 (None, 32, 32, 1024)      4719616
_________________________________________________________________
fc7 (Conv2D)                 (None, 32, 32, 1024)      1049600
_________________________________________________________________
conv8_1 (Conv2D)             (None, 32, 32, 256)       262400
_________________________________________________________________
conv8_padding (ZeroPadding2D (None, 34, 34, 256)       0
_________________________________________________________________
conv8_2 (Conv2D)             (None, 16, 16, 512)       1180160
_________________________________________________________________
conv9_1 (Conv2D)             (None, 16, 16, 128)       65664
_________________________________________________________________
conv9_padding (ZeroPadding2D (None, 18, 18, 128)       0
_________________________________________________________________
conv9_2 (Conv2D)             (None, 8, 8, 256)         295168
_________________________________________________________________
conv10_1 (Conv2D)            (None, 8, 8, 128)         32896
_________________________________________________________________
conv10_padding (ZeroPadding2 (None, 10, 10, 128)       0
_________________________________________________________________
conv10_2 (Conv2D)            (None, 4, 4, 256)         295168
_________________________________________________________________
conv11_1 (Conv2D)            (None, 4, 4, 128)         32896
_________________________________________________________________
conv11_padding (ZeroPadding2 (None, 6, 6, 128)         0
_________________________________________________________________
conv11_2 (Conv2D)            (None, 2, 2, 256)         295168
_________________________________________________________________
conv12_1 (Conv2D)            (None, 2, 2, 128)         32896
_________________________________________________________________
conv12_padding (ZeroPadding2 (None, 4, 4, 128)         0
_________________________________________________________________
conv12_2 (Conv2D)            (None, 1, 1, 256)         524544
=================================================================
Total params: 23,500,864
Trainable params: 23,500,864
Non-trainable params: 0
```

Figure 36: SSD512 model representation
Source: self-made

9.5. Model SSD300 Halfway Fusion

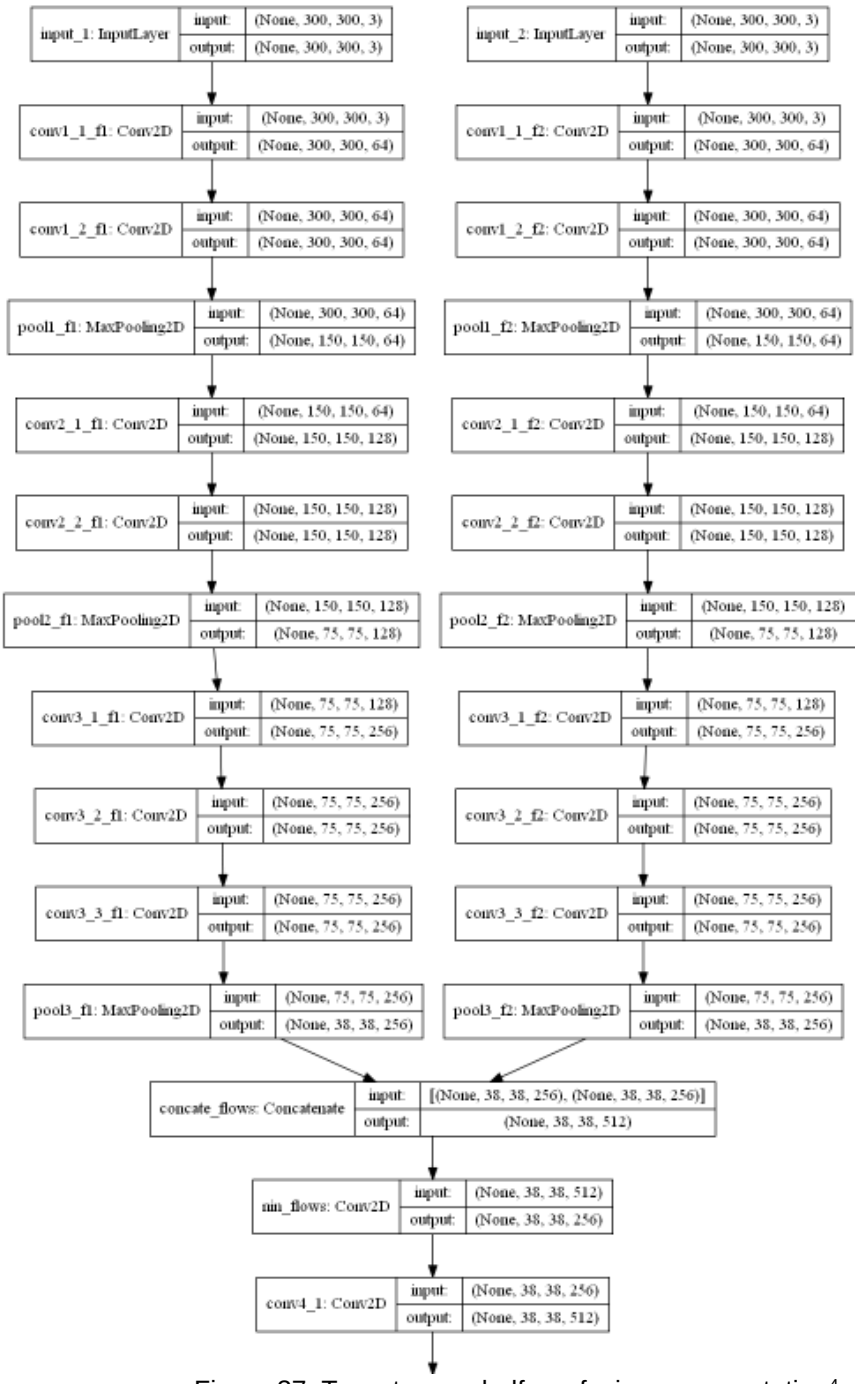


Figure 37: Two streams halfway fusion representation[4]
Source: self-made

```
Total params: 24,810,240
Trainable params: 24,810,240
Non-trainable params: 0
```

Figure 38: Total number of parameters of the halfway fusion model
Source: self-made

[4] From conv4_1 onwards follows the same structure as the SSD300 model and it will not be represented

9.6. Model SSD300 Late Fusion

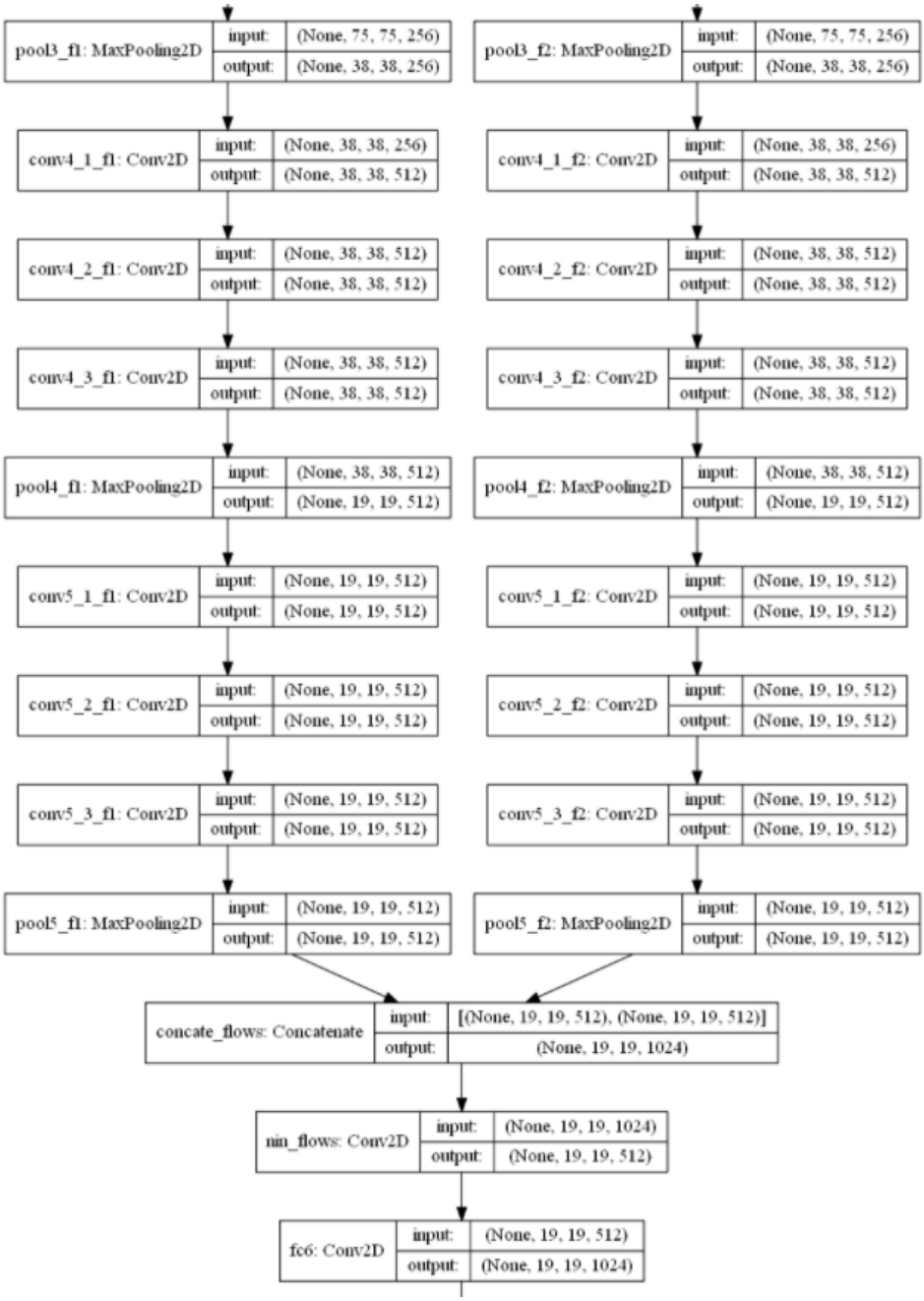


Figure 39: Two streams late fusion representation[5]
Source: self-made

```
Total params: 38,182,912
Trainable params: 38,182,912
Non-trainable params: 0
```

Figure 40: Total number of parameters of the late fusion model
Source: self-made

[5] Before pool3_f1 and pool3_f2 its representation is equivalent to the previously showed SSD300 Halfway Fusion model, from fc6 onwards is equivalent to the SSD300 model and it will not be represented

9.7. The three NIRs

Figure 41: From top to bottom, gated0, gated1 and gated2
Source: [4]