TRABALHO DE CONCLUSÃO DE CURSO

PARKING LOT AND TRAFFIC SURVEILLANCE USING A DEEP LEARNING APPROACH

Samuel Soares Santos

PROGRAMA DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
FACULDADE DE TECNOLOGIA
UNIVERSIDADE DE BRASÍLIA

FICHA CATALOGRÁFICA

SANTOS, SAMUEL SOARES
PARKING LOT AND TRAFFIC SURVEILLANCE USING A DEEP LEARNING APPROACH [Distrito Federal] 2021.
xvi, 50 p., 210 x 297 mm (ENE/FT/UnB, Engenheiro, Engenharia Elétrica, 2021).
Trabalho de conclusão de curso - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Monitoramento de pessoas e veículos	2. Detecção de objetos
3. Redes neurais	4. Visão computacional
I. ENE/FT/UnB	II. Título (série)

BIBLIOGRAPHY

SANTOS, S.S (2021). *PARKING LOT AND TRAFFIC SURVEILLANCE USING A DEEP LEARNING APPROACH*. Trabalho de conclusão de curso, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 50 p.

CESSÃO DE DIREITOS

AUTOR: Samuel Soares Santos
TÍTULO: PARKING LOT AND TRAFFIC SURVEILLANCE USING A DEEP LEARNING APPROACH.
GRAU: Engenheiro Eletricista ANO: 2021

Samuel Soares Santos
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

RESUMO

O presente trabalho foi realizado na Continental Engineering Services, em Toulouse, França. O objetivo foi desenvolvimento de uma aplicação para monitoramento de pedestres e veículos em tempo real usando redes neurais convolucionais, além do estudo de técnicas para melhorar a rede neural proposta. Uma arquitetura de tipo Centernet foi treinada e implementada. Os testes mostram que a aplicação funciona em tempo real, e é capaz de detectar com boa precisão veículos grandes e carros. Os desafios futuros consistem em melhorar a precisão para objetos pequenos do ponto de vista de uma câmera de segurança, principalmente pedestres e veículos de duas rodas. Além disso, foi mostrado que o uso de imagens geradas artificialmente na etapa de treinamento implica em uma melhora na performance do modelo, desde que tais imagens não sejam dominantes na base de dados.

ABSTRACT

The present work was performed at Continental Engineering Services in Toulouse, France. The objective was to develop a real-time pedestrian and vehicle monitoring application using convolutional neural networks, and to study techniques to improve the proposed neural network. A Centernet model was trained and implemented. Tests show that the application works in real time and is able to detect large vehicles and cars with good accuracy. Future challenges consist mainly in improving the accuracy for small objects from the point of view of a security camera, especially pedestrians and two-wheeled vehicles. Furthermore, it has been shown that the use of artificially generated images in the training stage correlates with improvement in model performance, as long as these images are not dominant in the training set.

SUMMARY

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

These works were developed in the helpful environment of Continental Engineering Services, where my colleagues were always available to help me think and work through problems. I would like to thank Rémi Trichet, Sergey Abrashov, and Paul Mendez. Their help, their teachings, and the different perspectives they provided on the many challenges of this internship were invaluable. They taught me how to think in the engineering aspects of an innovation project and how to adapt the problem statement according to the circumstances, skills that I hope to take with me for life.

I would like to thank my engineering school in France, ENSEIRB-MATMECA. I was received with open arms in a double-degree program from which I learned a lot, technically, professionally and culturally. This school made me feel ready to take on the challenges of being an engineer. I would like to thank Prof. Marc Donias and Prof. Yannick Berthoumieu for their guidance and Prof. Nathalie Deltimple for the help in all the aspects of the double-degree program.

I have a lifetime debt with the University of Brasilia and the Federal District of Brazil, for giving me the opportunity of having a free high-quality education. I would like to leave a special thanks to Prof. Mylène de Queiroz, who introduced me to the image and signals domain and to the adventures of research projects.

Without the funding from CAPES and the BRAFITEC program, I could not have participated in this double-degree program. I am truly grateful for the trust I got from them and for the opportunity of having an international experience that, quite frankly, changed my life.

Thank you to Océane Lambert, who was my support during hard times, helped me understand the nuances of the French culture, and much more.

Finally, I would like to thank my family and friends, both in Brazil and Europe, for the emotional support. They were always there for me, to celebrate or to help.

1 INTRODUCTION

Video surveillance is becoming more present in they daily live. A wide range of cameras and drones is used for real-time surveillance of outdoors zones like traffic intersections, parking lots, public parks, streets, entrances of commerce, and the list goes on. However, treating this huge amount of raw data and providing useful feedback in real-time is humanly impossible, and most of it stays stocked as raw data.

The use Artificial Intelligence, especially Deep Learning techniques, for analysing this data has been increasing recently due to advances in hardware, power consumption, and Neural Networks performance. Real-time processing of a video sequence captured by a surveillance camera is imperative for some applications, which makes processing speed a constraint to the choice of algorithm. However, the general behaviour is for deeper and slower neural networks to present higher accuracies, configuring a trade-off.

Academic research on the field of video surveillance includes parking space counting((12), (11)), traffic flow counting((13), (14)), pedestrian tracking((15), (16), (1)), and many more ((17)). This problem is different than object detection from a car driver's point of view because the scene changes slowly and the camera's perspective is fixed, as it is shown in figure 1. Besides, the incentive for the development of autonomous vehicles has created a demand for research on this subject, increasing the availability of public available datasets ((18), (19), (20)), possibly with licenses allowing commercial use. This is not the case for surveillance-like applications, which have tighter licensing constraints.

The goal of Continental Engineering Services (CES) is to develop a new product based for environment motoring, that is, for outdoors surveillance applications.

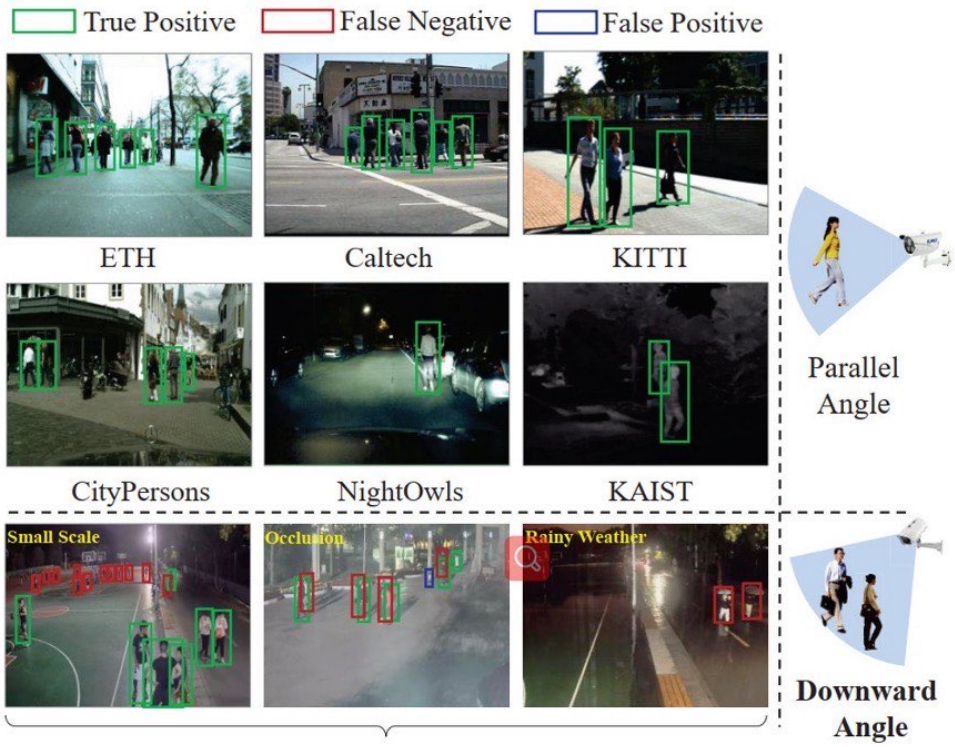


Figure 1: Comparison between images taken from a parallel angle and from a downward angle (1).

1.1 Problem Discussion

As discussed, the most suitable tools for developing this new project are Convolutional Neural Networks(CNNs). However, training a deep learning model from scratch is a resource-intensive and time-consuming task. It is the role of an engineer to keep in mind the time and resources his work demands, especially considering its economic and environmental repercussions. In addition, the use of alternatives already developed by other teams speeds up the integration processes and is common practice in the industry. Therefore, the use of pre-trained deep learning models and, whenever possible, the use of task-specific algorithms as a starting point is preferred to the from scratch approach.

Robustness is a constraint for algorithms to be integrated into products. In the case of a general-purpose surveillance algorithm, that means it should work with a variety of camera heights, angles, and resolutions, as well as different available resources. The closest available product in CES being a detector from a driver's point of view, one approach is to adapt this detector to a surveillance point of view. By keeping the performance both in driver's point of view and stationary downward camera's point of view, one would ensure that the algorithm is robust enough to work with different camera configurations.

The definition of the target categories to be detected is tied to the application. The goal being the development of a general purpose surveillance algorithm, we are mostly interested in vehicles and pedestrians in the legal term. For example, a truck or bus have similar requirements for parking, as do vans and cars. In additions, joining categories is, as will be discussed later, a way improving performance by loosening the constraint of distinguishing intra-class variance. Therefore, our target categories are:

- car: passenger car, van, pickups, and similar vehicles;
- heavy: buses, mini buses, trucks and similar vehicles;
- two-wheels: mainly motorcycles and bicycles;
- person: individual pedestrian.

As will be discussed in the Data Management section, the availability of datasets whose license allows its commercial use on the task of environment monitoring is low. Therefore, one of the objectives is to generate synthetic data and evaluate its suitability.

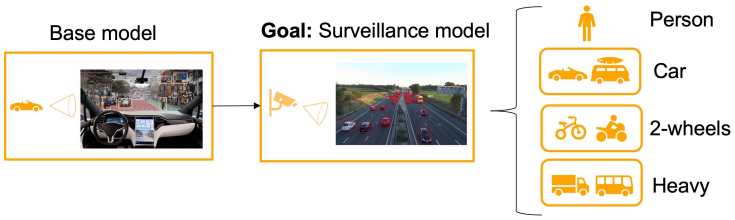
The use of surveillance camera footage implies that some objects might be small and under sampled in the dataset, especially the person and two-wheels categories. In this scenario, we want to evaluate how we can improve performance for under sampled small objects.

It is important to state that these considerations were made during the project, implying the adaption of the problem statement during the development. This is expected in the beginning of a new project's cycle, and these remarks are therefore part of the development itself.

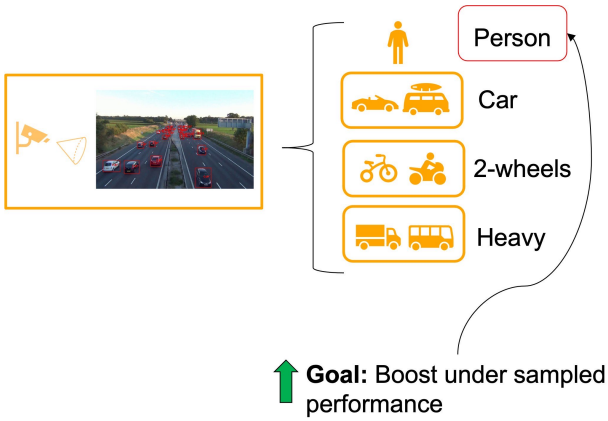
1.2 Objectives

All the considerations in this introduction lead to the definition of the following objectives:

1. adapt a model suitable for real-time detection from a driver's point of view to a surveillance camera's point of view while keeping good performance in both tasks;
2. study how to improve the performance of the model on under sampled categories and small objects;
3. determine if the use of synthetic data for training purposes can improve performance;
4. develop an application, proving the concept of this new project at Continental Engineering Services.

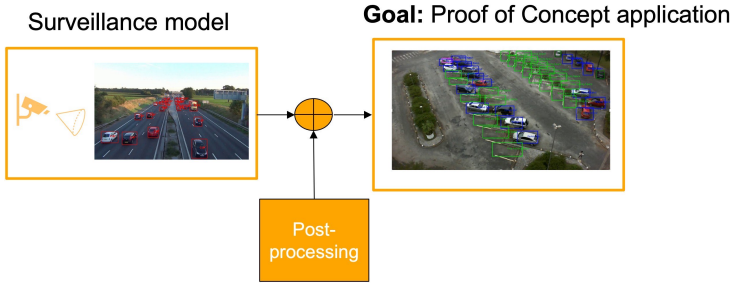


(a) Illustration of objective 1.



(b) Illustration of objective 2.

(c) Illustration of objective 3.



(d) Illustration of objective 4.

Figure 2: Illustration of the objectives of the project.

2 BIBLIOGRAPHICAL OVERVIEW AND STATE OF THE ART

The core problem of the internship is object detection, which has been dominated by Deep Learning (DL) approaches in the last years. The goal of this section is to present a general overview of the state of the art in object detection, justifying the choices made through the project. The metrics used will also be explained, as well as applications close to ours.

2.1 Object detection

Liu *et al.* (2) defines object detection as the task of finding if there are any instances of objects from given categories in an image, as well as the spatial location and extent of each object instance. Alternatively, semantic segmentation is the task of finding binary masks of dimension $H \times W \times C$, where H is the height of the image, W its weight, and C is the number of channels ($C = 3$ for color RGB images), translating the pixel-wise annotation of the category of interest. The task of instance segmentation adds complexity by demanding the labelling of each instance separately.

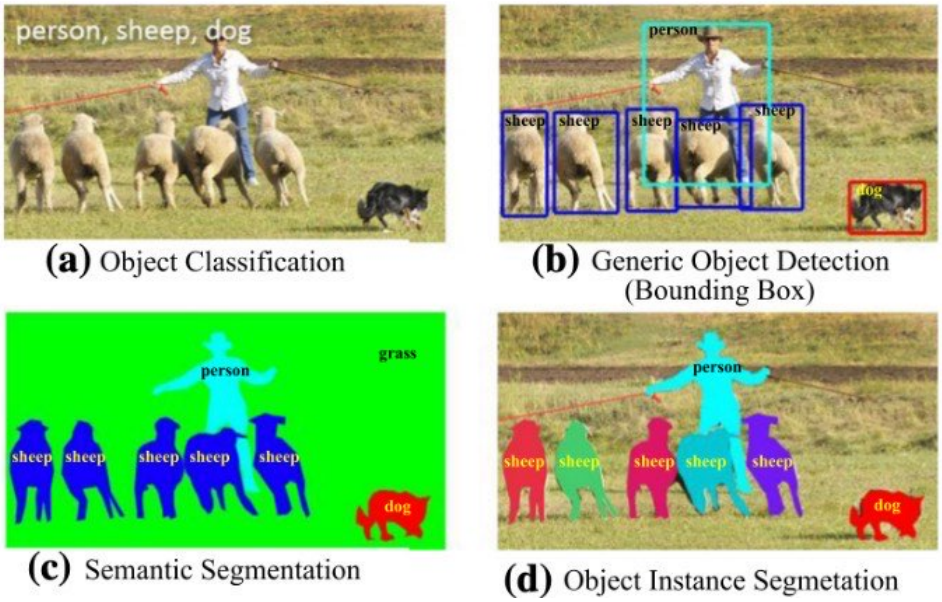


Figure 3: Object detection task overview (2).

The objectives of this project are better attained with Generic Object Detection, that is, the task of finding the $H^{'} \times W^{'}$ patches of the images containing the target objects (with the restriction that $H^{'} < H$ and $W^{'} < W$). From now on, these patches will be denoted as *bounding boxes*.

The object detector is therefore an algorithm that receives an image as input, and outputs N vectors of the form:

$$\begin{bmatrix} x_0 \\ y_0 \\ x_1 \\ y_1 \\ cat \\ score \end{bmatrix}_i , i = 1,...,N \tag{1}$$

where cat identifies the category of the object, $score$ is a scalar in the interval $[0,1]$ representing the confidence on the detection, (x_0,y_0) represents the top-left coordinates of the bounding box and (x_1,y_1) represents its bottom-right coordinates.

2.1.1 General architecture and choice of algorithm

The architecture of modern object detector can be resumed in the elements of the figure 4.

The input is generally an image (in our application, a RGB image). The backbone is a feature extractor, and usually is the most computational expensive part of the network. Its goal is to transform the input in a series of tensors across multiple scales. Intuitively, this brings the input into a higher dimension tensor where information is less correlated among itself. This has been historically beneficial to the performance of neural networks.

It has been shown that the deepest layers of the backbone have higher semantic value, whereas the locations of the objects are clearer in the lower-level features in the beginning of the backbone. The neck joins these features, which have been shown to increase performance. A variety of methods are proposed in the literature, as the original Feature Pyramid Networks (FPN) (21), the results of Natural Algorithm Search FPN (22) and Deep-Layer Aggregation (4) (which is also a backbone).

There are two different approaches in the literature for the next final step: one-stage or two-stage object detection. In the latter, a Region Proposal Network (RPN) is employed to find regions of interest in the feature maps by proposing a fixed number of bounding boxes regions and their probability of containing an object. This stage is followed by a sort of classifier, whose is role to name the category of each region and refine the position of the bounding box. Examples of architectures using this approach are the Faster-RCNN (23) and Mask-RCNN (24).

Alternatively, the region-proposal and classification can be done in a single step. Algorithms like Yolo v4 (3) and EfficienDet (25) do so by defining anchor boxes which are regressed to find the target object's bounding boxes. On the other hand, the Single Shot Detector (26) and Centernet (27) do so by finding the centre point of objects and regressing their bounding box size.

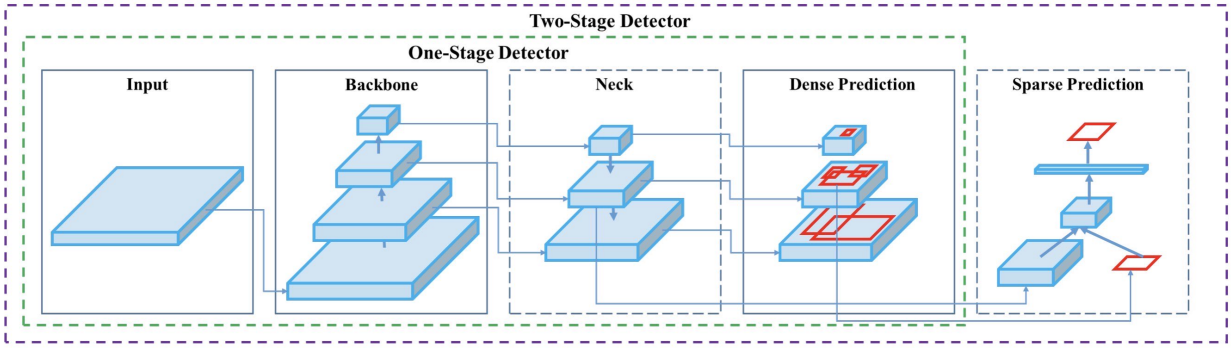


Figure 4: General object detector structure. Source: (3).

The Deep Learning based algorithm used in this project is Centernet (27), an anchor-free object detector. This choice is based on a series of reasons:

- it is already used in a series of CES's projects and products, making it a preferred choice from the company's point of view;
- it is anchor-free, meaning it needs low post-processing, which makes it easily integrable in embedded devices.
- it is fast, as real-time inference is a requirement, the use of a one-stage object detector is crucial;
- it is the state of the art anchor-free one-stage object detector;

In addition, the fact that this model is anchor-free makes it robust to resolution variations between cameras, which is a positive point in the development of a general-purpose project.

2.2 Centernet

The general pipeline of Centernet(27) is presented in figure 5, with a input image of dimensions $H \times W \times 3$. As every modern one-stage object detector, the image is passed through a backbone and a neck. The former's outputs are 3 heads whose dimensions are 4 times smaller than those of the original image, that is, their output stride is 4.

The Heatmap (hm) head has dimensions of $\frac{H}{4} \times \frac{W}{4} \times C$, where C is the number of categories. It contains the (x,y) coordinates of the central points of each detected objects of each target category.

Visually, the width-height(wh) head has dimensions $\frac{H}{4} \times \frac{W}{4} \times 2$, representing the width and height of each object's bounding box. However, this head is in fact a $K \times 2$ matrix, where K is the maximum number of objects to be detected ($K = 100$ in this project) and each row in the format $[weight,height]$. Fixing the value of K allows to consume less memory.

Lastly, the local offset head has a "theoretical" dimension of $\frac{H}{4} \times \frac{W}{4}$, but it is also reduced to a $K \times 1$ vector to avoid memory consumption. It predicts an offset between the detected objects' bounding boxes in the output stride of 4 (in the heads) and the original stride ($H \times W$). In short, this head accounts for the discretization error.

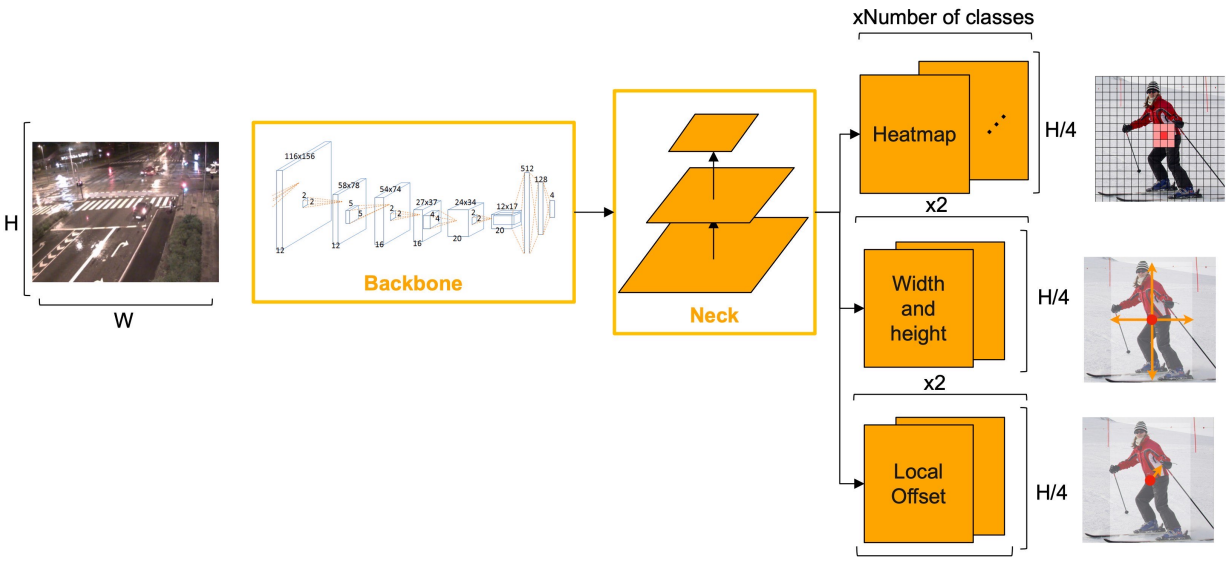


Figure 5: General Centernet's architecture.

2.2.1 Loss functions

During the training phase, the parameters of the network are adapted by the optimiser (i.e, stochastic gradient descent, ADAM) to minimise of the loss function. Therefore, it is crucial to correctly define these functions.

Centernet's total loss L is a weighted sum three of loss functions, one for each of its heads:

$$L = L_k + +\lambda_{size}L_{size} + \lambda_{off}L_{off} \tag{2}$$

The weights were fine-tuned by the authors as $\lambda_{size} = 0.1$ and $\lambda_{off} = 1$, and we use it as such.

The heatmap head's loss L_k is the main one. It is defined in equation 3:

$$L_k = \frac{1}{N}\sum_{xyc}\begin{cases} -(1-\hat{Y}_{xyc})^{\alpha}log(\hat{Y}_{xyc}) & \text{if } Y_{xyc}=1 \\ -(1-Y_{xyc})^{\beta}(\hat{Y}_{xyc})^{\alpha}log(1-\hat{Y}_{xyc}), & \text{otherwise} \end{cases} \tag{3}$$

where $Y_{xyc} \in [0,1]^{\frac{H}{4}\times\frac{W}{4}\times C}$ is the ground truth heatmap tensor constructed using the annotations of the current image, $\hat{Y}_{xyc}$ is the Centernet's output heatmap tensor, α and β are hypermeters for weighting positive and negative samples.

This is a variant of the focal loss (28), a popular loss function in object detection known to help with the class invariance problem. The ground truth centre points are splattered using a gausian kernel:

$$Y_{xyc} = exp\left(\frac{(x-\tilde{p}_x)^2 + (y-\tilde{p}_y)^2}{2\sigma_p^2}\right) \tag{4}$$

where $\tilde{p}_x = [\frac{p_x}{4}]$ and $\tilde{p}_y = [\frac{p_y}{4}]$ are the centre points of the object located at (p_x,p_y) in the input image,

and σ_p is per-category standard deviation. With this approach, even the points near the centre point will have a heatmap value close to 1. Because the focal loss defined in equation 3 does not greatly penalise close classifications, the total L_k loss will not be so high. Essentially, the use of a focal loss alongside the gaussian kernel considers neighbour points of the object's central point as possible candidates. Moreover, the hyper-parameters α and β have been fine-tuned by the authors to 2 and 4 respectively, and we use them as such.

The width-height loss is a simple L1 loss. The ground truth GT and the detections DET are in the format:

$$GT = \begin{bmatrix} w_0^{GT} & h_0^{GT} \\ w_1^{GT} & h_1^{GT} \\ ... & ... \\ w_{K-1}^{GT} & h_{K-1}^{GT} \end{bmatrix} \text{ and } DET = \begin{bmatrix} w_0^{DET} & h_0^{DET} \\ w_1^{DET} & h_1^{DET} \\ ... & ... \\ w_{K-1}^{DET} & h_{K-1}^{DET} \end{bmatrix} \tag{5}$$

Therefore, the L1 loss is calculated as:

$$L_{size} = L1(GT,DET) = \sum_{i=0}^{K-1} |w_i^{GT} - w_i^{DET}| + |h_i^{GT} - h_i^{DET}| \tag{6}$$

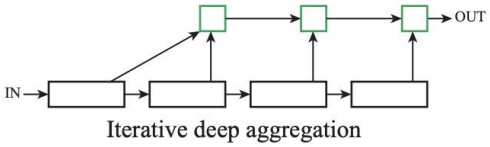
Similarly, the offset loss is also defined as a L1 loss between the ground truth reconstruction offsets and the predicted reconstruction offsets.

2.2.2 Deep Layer Aggregation

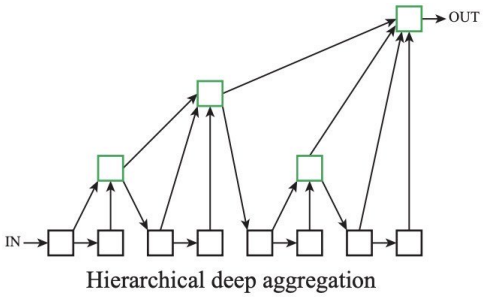
The authors of Centernet reviewed a series of backbones. We choose to use the Deep Layer Aggregation with 34 aggregation levels, because it performs only slightly worse to the top-rated Hourglass (29) while taking 3.5 times less processing time in the authors' experiments.

The central idea is the aggregation of deeper and shallower features, keeping the spatial and semantic information as illustrated in figure 6. The use of Iterative Deep Aggregation (IDP) allows the refinement of shallower features, and the Hierarchical Deep Aggregation helps to keep a spam of the hierarchy of the features in the backbone. The HDA's role is complementary to the IDA's one: if forwards the aggregation of all previous blocks instead of just the previous one. The final Deep Layer Aggregation's (DLA) architecture uses both.

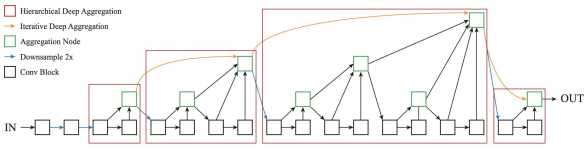
By joining features from different levels, the DLA also fulfils the role of the network's neck, therefore recuding the model's total complexity.



(a) Iterative Deep Aggreagtion(IDA)



(b) Hierarchical Deep Aggreagtion (HDA)



(c) Deep Layer Aggreagtion (DLA)

Figure 6: Representation of HDA, IDA and DLA. Source: (4).

2.3 Metrics

Evaluating the performance of an object detector can be a complicated task, and the metrics for doing so must be adapted to the task at hand. This section aims to explain the definitions, processes and parameters chosen to evaluate the models in these works.

The code used is standalone, so it can work with any object detection algorithm to produce repeatable and comparable results in future works on the subject.

Some basic definitions are shared thought different metrics. It must be stated that, as a pre-processing step, the detections *must* be sorted by decreasing confidence score (6) before proceeding the evaluations take place, as the detector itself privileges the high-confidence detections. The detections with the top $K = 100$ scores of each image were considered in the evaluations through this project.

2.3.1 Intersection over Union

The basic definition is the *Intersection Over Union (IoU)*, a scalar in the interval $[0,1]$ representing the ratio between the area of the intersection between a detected object's bounding box with a ground truth annotation and their union's total surface area, as illustrated in the figure 7.

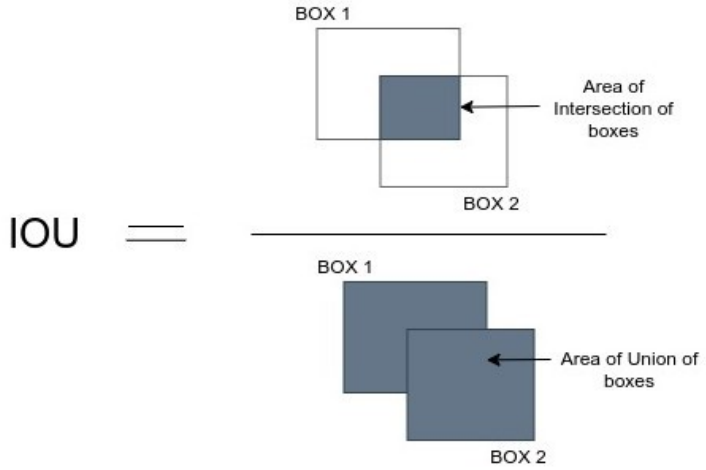


Figure 7: Intersection of Union (IoU).
Source: <https://medium.com/analytics-vidhya>

.

2.3.2 True positives (TP), False Positives(TP) and False Negatives(FN)

Assuming the detections are sorted by decreasing confidence score, the True Positives, False Positives and False Negatives are defined according to the following algorithm:

1. *set* all the *ground truths* in the image as unmatched;
2. *loop* through all the detections made on the image;
3. *calculate* the IoU between the present detection and all the unmatched *ground truths* in the image;
4. *if* the highest IoU is more than a given threshold (the value chosen for this project is 0.5), set the corresponding detection as a **true positive** and the corresponding *ground-truth* as *matched*. Otherwise, set the detection as *false negative*.
5. Set all the unmatched *ground-truths* as *false negatives*, and proceed to the following image. Repeat until all the images in the *test set* are evaluated.

These definitions are the basis for the precision and recall, as well as for the F1 score. A summary is shown in figure 8. Not that the concept of True Negative (TN) does not apply to the object detection problem because the detector is not looking to classify areas without objects.

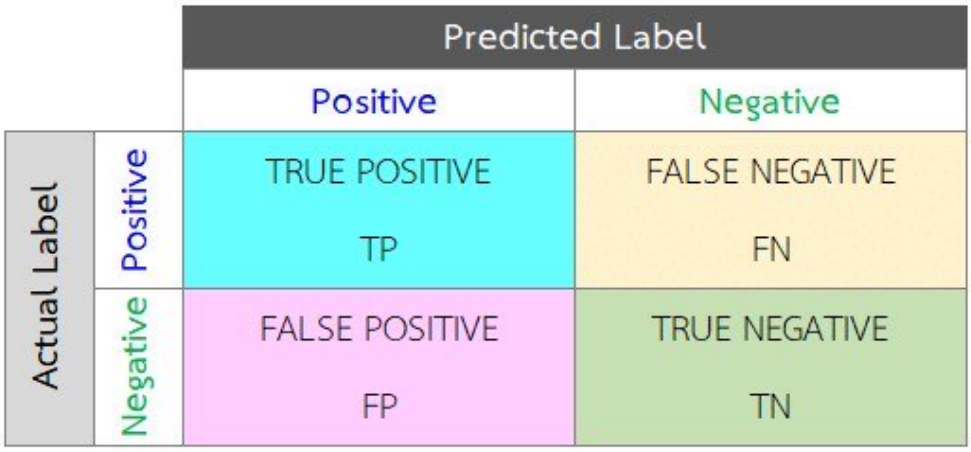


Figure 8: Summary of True Positive, True Negative, False Positive and False Negative definitions (5).

2.3.3 Precision and Recall

The precision is a metric that tries to answer the following question: what proportion of positive detections is correct (30)? On the other hand, the recall seeks to define the proportion of possible positives which are correctly identified. Formally:

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

Note that these two metrics are in fact curves, defined for each level of confidence score. As we consider lower confidence scores, the precision *tends to decrease* and the recall increases, as shown in figure 9. Therefore, these two metrics have a trade-off among themselves, defined by the confidence score threshold: when one wants to increase the recall, it *usually* must decrease the precision, and vice-versa.

The precision and recall are not always are trade-off. However, if one can increase both, there is no reason to choose a confidence score point in between. For this reason, the precision versus recall curve for practical purposes is the interpolated one, as shown in the same figure.

We note that the precision and recall curves are calculated for each category of the detector.

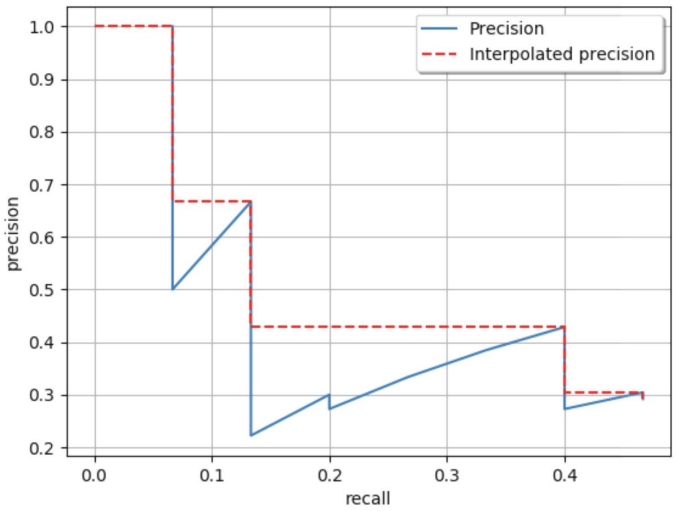


Figure 9: Example of precision $\times$ recall curve. Source : (6)

2.3.4 Average Precision(AP) and Mean Average Precision (mAP)

The area under the curve of the interpolated precision versus recall curve is defined as the Average Precision (AP), calculated for each category of the object detector. This metric represents the trade-off between precision and recall in a single scalar value in the interval $[0,1]$.

Different benchmarks have different approaches on how to calculate this metric: for example, the PASCAL-VOC (31) challenge uses a 11-point interpolation of the precision versus recall curve to calculate the AP.

We choose to calculate average precision of the category c, noted AP_c, similarly to the MS-COCO challenge (32), that is, the AP_c is the all-points interpolated curve's area:

$$AP_c = \sum_{i=1}^{N} precision(recall_i) \times recall_i \tag{9}$$

where N is the number of detections. Furthermore, the mAP is defined as the average of the APs for all the categories in the model:

$$mAP = \frac{1}{C} \sum_{c=1}^{C} AP_c \tag{10}$$

2.3.5 F1-score

The F1-score translates imbalance between precision and recall, and is defined as the harmonic mean between each precision and recall at each point. For the category c, it is defined as:

$$F1_c(recall) = 2\frac{precision(recall) \times recall}{precision(recall) + recall} \tag{11}$$

During these works, the F1 is represented as a function of the confidence score threshold of the detector, which means that it can be used to find the optimal per-category confidence threshold, as exemplified in the figure 10.

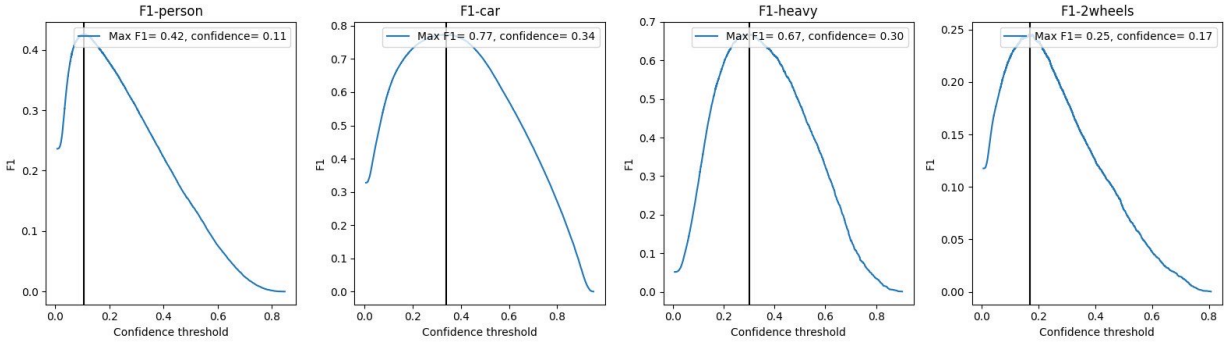


Figure 10: Example of F1-score *versus* confidence threshold curve.

2.3.6 Weighted F1-Score

To translate the overall F1-score across categories in a single scalar value, we define the *weighted F1-score* as the weighted sum of the maximum per category F1-scores, the weights being the number of elements of each category (i.e., number of points in the precision versus recall curve of each category)

Let L_c denote the number of elements in the precision vector of the class $c \in [1,...,C]$. The weighted F1-score $wF1$ is defined as:

$$wF1 = \frac{1}{\sum_{c=1}^{C} L_c} \sum_{c=1}^{C} max(F1_c) \times L_c \tag{12}$$

2.4 Surveillance applications

As discussed in the previous sections, our choice of algorithm is Centernet. Therefore, we study some applications of this algorithm in the literature. The domains are similar, but not identical, to those in our project.

2.4.1 Pedestrian surveillance

The authors of Pedestrian as Points (PP-net)(33) used U shaped feature pyramids to join feature maps of the backbone more effectively, avoiding large semantic gaps (Feature Fusion Unit). It also ads skip connections to the pyramid levels (Deep Guidance Module) to better aggregate the features before feeding it to a centre point-based object detector. However, the authors do not study the inference time or number of parameters of the proposed model, which discourages its use in applications where real-time inference

is crucial.

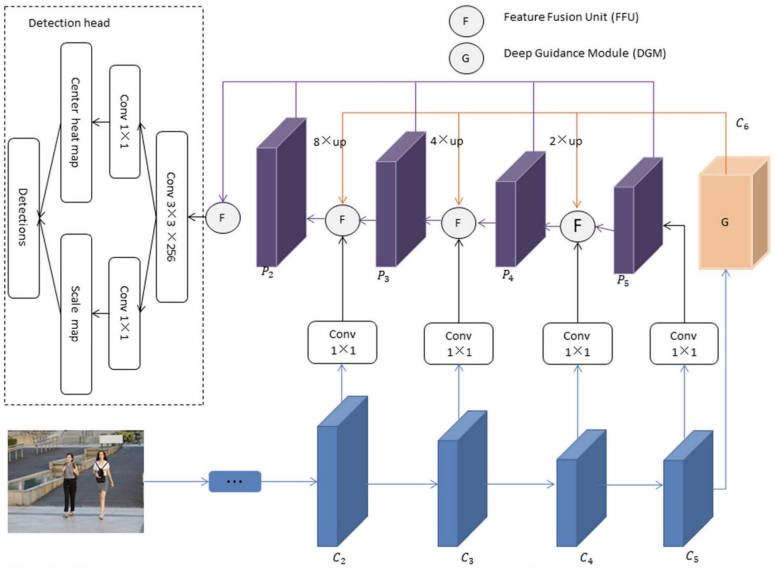


Figure 11: PP-net architecture.

2.4.2 Drone surveillance

Images taken from the point of view of a drone are not the target of this project, but these can be like surveillance camera images when the drone's camera is angled downwards and the drone itself is not very high up. Therefore, research on this domain is applicable to ours.

The authors of (8) propose a straightforward approach: train a Centernet model on the Visdrone (7) dataset and compare it to state of the art models on the dataset's challenge. During the test phase, they use a multi-scale approach, passing the images on the scales 0.5, 0.75, 1.0, 1.25 and 1.5, and averaging outputs. The results of the challenge are shown in figure 12, where we can see the proposed model in the 5th position. Considering only one of the models above this one is a one-stage detector, we can position Centernet as one of the best algorithms in the task. However, the multi-scale pre-processing increases the inferences complexity by 5, and therefore these results must not be taken for granted considering a performance point of view.

Figure 12: Examples of inference on Visdrone(7) aerial images using the proposed centernet model(8).

Position	Method	AP[%]	AP50[%]	AP75[%]	AR1[%]	AR10[%]	AR100[%]	AR500[%]
1	DBAI-Det	29.22	58	25.34	14.3	35.58	50.75	53.67
2	AFSRNet	24.77	52.52	19.38	12.33	33.14	45.14	45.69
3	HRDet+	23.03	51.79	16.83	4.75	20.49	38.99	40.37
4	VCL-CRCNN	21.61	43.88	18.32	10.42	25.94	33.45	33.45
5	**CN-DhVaSa(ours)**	21.58	48.09	16.76	12.04	29.6	39.63	40.42

Figure 13: Results of presented on the Visdrone dataset. Source: (8)

2.4.3 Traffic counting

Estimating the number of vehicles passing through a specific traffic intersection is crucial for traffic planning, especially in big cities. The authors of (9) combine state of the art one-stage object detectors to estimate real-time traffic flow. The results are presented in figure 14, showing competitive results for combinations involving Centernet, including one of the best (Centernet and DeepSort). In addition, the authors stated that it took roughly 22 hours for training Centernet on a GTX 1080Ti GPU with 11.000 thousand images, which is a reasonable time for resources similar to ours.

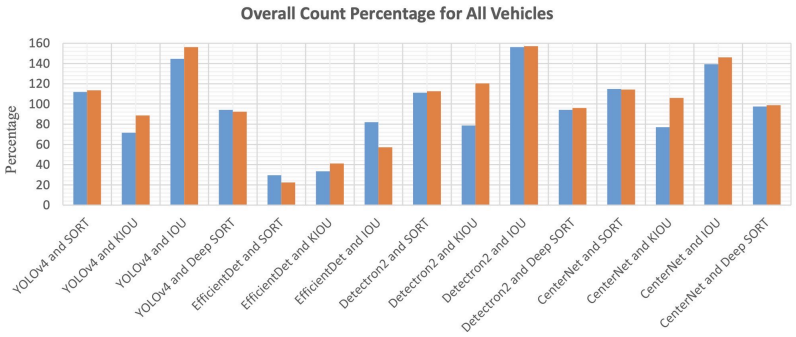


Figure 14: Results of the experiments on traffic flow count. Source : (9).

3 DATA MANAGEMENT

Data is the first and arguably the most important step for building a Deep Learning algorithm. However, datasets for surveillance-like applications whose license allow commercial use are rare, probably because of privacy concerns. Non-Commercial license still allows the use of the data for testing and benchmarking, but not as part of training or validating sets.

In this scenario, the data management is of crucial importance for this project. We will start by reviewing the options found in the literature, and if they can be used now or in future works. The adaptions and pre-treating made to the chosen data will also be presented, as well as statistics and the impacts it will have on our project. Moreover, we will present data augmentation options and discuss synthetic data generation.

This section, in which we will review the datasets effectively used during these works, is an extension of the bibliography review. Moreover, the data used to develop our Proof of Concept is described in its own subsection (4.10), reserving this one for object detection databases only.

3.1 AAU Rainsnow

The AAU Rainsnow dataset (34) (during these works, named AAU for short) is focused on traffic surveillance under bad weather conditions as rain, snow, haze, or fog, as well as low light conditions at night-time. It contains 22-minute videos from seven different traffic RGB and thermal cameras, both with a 640×480 pixels resolution. In these videos, there are 2200 frames with pedestrians, bicycles, motorcycles, cars, vans and trucks annotated on the pixel-level, meaning this dataset is suitable for both semantic segmentation and object detection problems (a bounding box is easily extracted from the extremes of an object's semantic map). To our knowledge, this is the *only* traffic surveillance dataset with a stationary downward angle camera whose commercial use is allowed.

This dataset demands some adjustments before it can be used for our purposes. It supposes that information from both RGB and thermal camera will be used, and it is annotated accordingly. Therefore, some unidentified objects in the RGB camera due to lighting or weather conditions are annotated because they could be detected in thermal camera.

Because traffic surveillance is the main task, some areas that could contain vehicles do not have annotations, like parking lots. However, a correctly trained model should detect vehicle no matter its surroundings, according to our goals. In addition, careful visual inspection also showed that some regions contained nearly non-identifiable objects by the human eye, especially faraway roads. Moreover, some categories were wrongly annotated (i.e truck as van or bicycle as pedestrian).

(a) Example of fog and rain.

(b) Example of nighttime and glare.

(c) Example of parking spaces.

Figure 15: Samples from the AAU Rainsnow Dataset illustrating some of its challenging scenarios.

To tackle this issues, a per-image inspection of the annotations was made, and some easily identifiable problems were corrected. The annotations of objects that could not be identified only by visual inspection of the RGB camera's image were removed. Some images were removed from the dataset.

In order to avoid overfitting the model or super estimating its performance on the cars in the parking lots, their positions were substituted by a black region, that is, whose RGB value is set to $[0,0,0]$. The same was done to faraway roads where objects were hardly distinguishable.

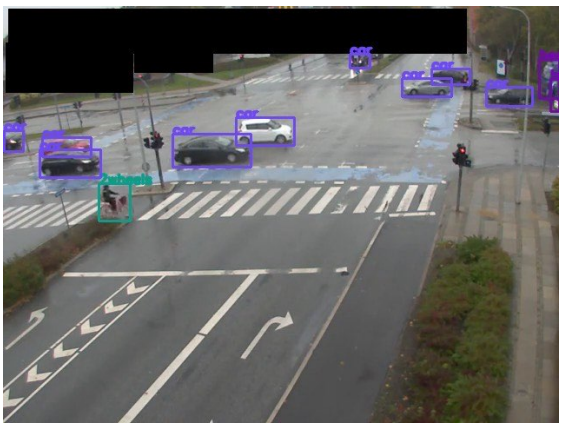
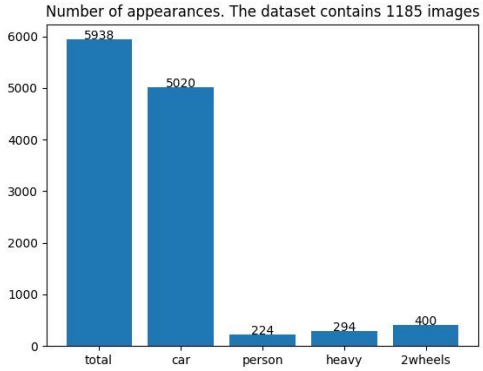


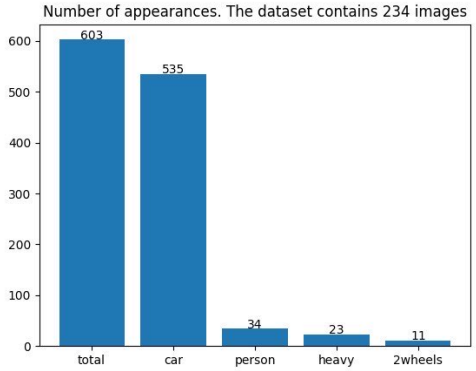
Figure 16: Example of removed regions of the filtered AAU dataset.

The number of images and instances of each category in each split are presented in figure 17. The

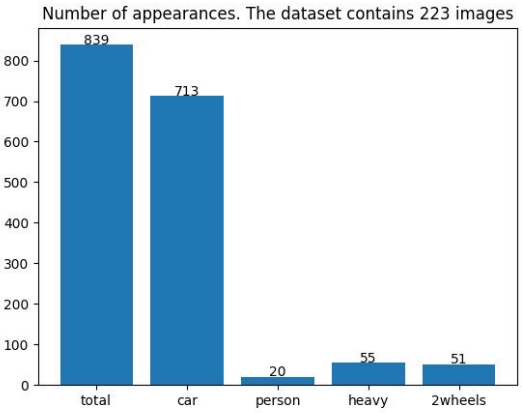
categories represented are those described in the introduction (section 1.1). As expected empirically, the passenger cars are heavily over sampled relative to other categories in all the splits. In addition, the distribution of categories is different between splits, especially in the test set. It is important to state that to avoid similar frames between splits, each sequence was only employed exclusively for train, validation, or test, which explains the difference in sampling of categories.



(a) Train split.



(b) Validation split.



(c) Test split.

Figure 17: Per-category statistics of the defined AAU's train, validation and test split.

The figure 18 presents a per-category bounding box area distribution of the dataset. For comparison with other datasets, all the images were resized to $height = 640$ and $width = 1152$ pixels. The standard deviation is high, mainly because the camera is angled, therefore farther objects of the same category can be smaller. However, the person category is alarmingly smaller than the others, which is a known challenge to object detectors (35). This problem will be further be discussed in the results section 4.6

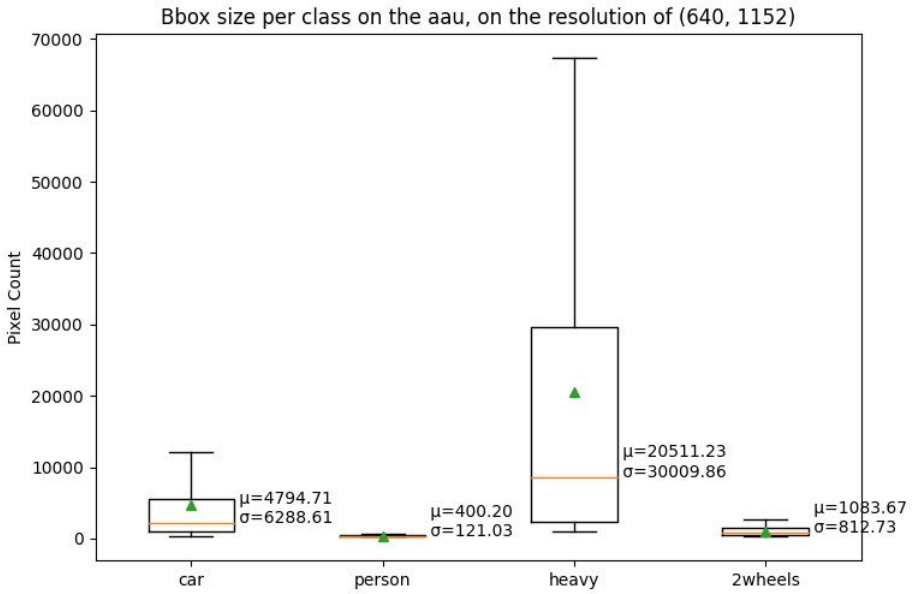


Figure 18: Average per-category bounding box size, AAU dataset.

3.2 BDD100k & BDD10k

The Berkley Deep Drive database (20) is known for being one of the largest driving datasets adapted to the task of object detection. It contains 100,000 frames of 1280×720 pixels resolution, 90,000 of which have publicly available annotations of cars, traffic signs, traffic lights, buses, trucks, pedestrians, riders, motorcycles, bicycles, and trains. Its license allows its commercial use, and it was partially used to train the base Centernet model at CES. A different group of 10k frames, namely the BDD10k, are pixel-wise annotated, allowing its use for both semantic segmentation and object detection.

Figure 19: Some samples from the BDD100k dataset.

Compared to the AAU dataset, the BDD100k is huge, which is a challenge to our approach of joining both datasets. One of the objectives stated in section 1.2 being to keep performance in both surveillance camera and driver's view points of view, we choose to use only a part of this set for training validation and testing purposes, as it will be explained in the section 3.5.

One small incoherence between the AAU and BDD datasets is that the latter annotated the rider of a two-wheel vehicle, and the former considers the two-wheel vehicle as the rider and the vehicle. The adjustment made to the BDD annotations is shown in figure 20: we match all the bounding boxes of riders with the vehicles they most overlap with, and we join the two bounding boxes by defining the smallest region that intersects both of the original bounding boxes. The frames with parked two-wheel vehicles (that is, without a rider) are removed, which is not a problem since we do not aim to use the whole dataset, and because frames like these are not numerous.

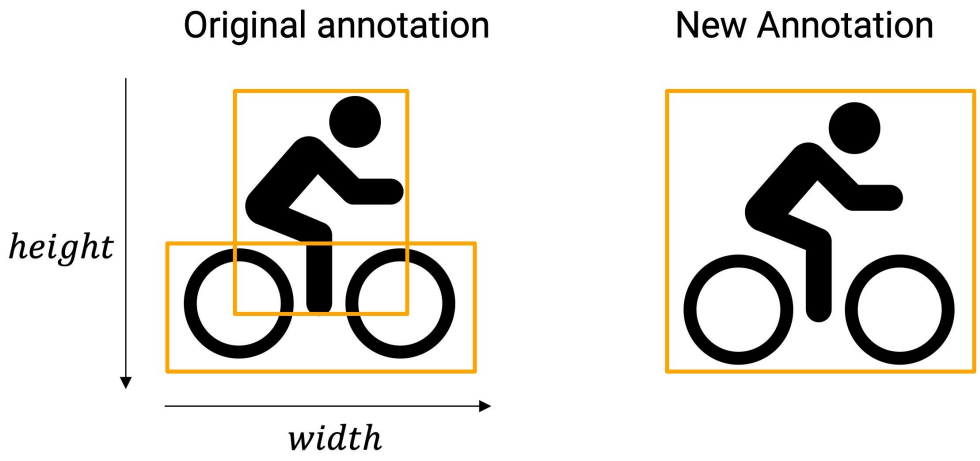
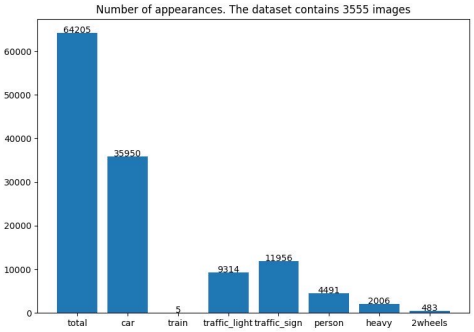
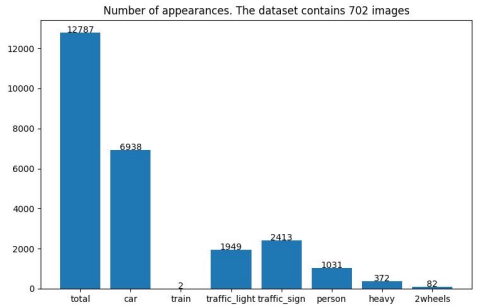


Figure 20: How the categories rider and vehicle are joined.

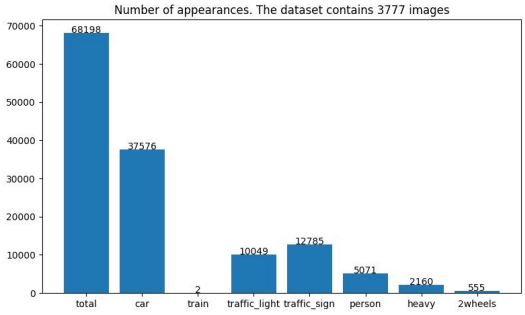
The statistics of the train, validation and test sets extracted from the BDD100k dataset are presented in figure 21 (note that some categories are *not pertinent* to our problem). We observe that the car category is overrepresented, as in the AAU's splits, but the person category is better represented. This implies that we need more surveillance-like data to make the test set more reliable. As the frames are random, the validation, train and test splits have a nearly identical distribution of objects.



(a) Train.



(b) Validation.



(c) Test.

Figure 21: Per-category statistics of the defined BDD train, validation and test splits.

Figure 22 show the per-category bounding box area distribution on the resolution of 1152×640 pixels, the same shown in figure 18 for the AAU dataset. As expected, objects viewed from a driver's point of view seem larger than the same objects viewed from a surveillance downward camera. The standard deviation is larger: as the angle of view gets more parallel to the ground, changes in perspective means smaller objects as the distance increases. The problematic cases of two-wheel vehicles and person categories are easier to identify in this dataset as their average surface areas are 6.11 and 5.29 times larger then AAU's, respectively.

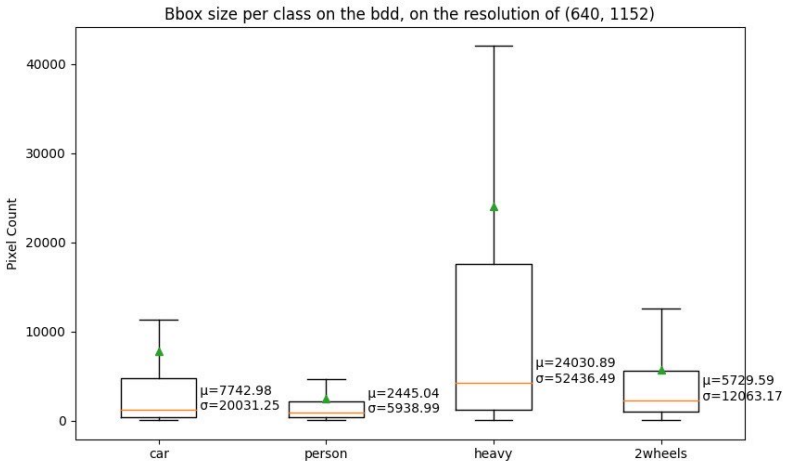


Figure 22: Average bounding box size for each category, BDD dataset.

3.3 Synthetic data

One of the first large scale synthetic driving datasets was the VKITTI (36), which pushed research into more development, including a new version (37) of the same dataabase. For example, the domain adaption approach described in (38) consists in using diverse parameters for the simulation. The authors have shown that the use of this technique correlates to performance, but the addition of real images is important to increase the model's performance. The issues from using only synthetic data are related to the difference between real-world and synthetic image distributions (39).

Synthetic data is definitely an important tool for saving time and resources in gathering and annotating real data, but its usage must be further investigated. On one of the few public available works on the surveillance context, the authors of (40) have been able to obtain better results by using only synthetic pedestrians than hybrid real-synthetic models. Another example is a traffic surveillance camera rain removal done by the authors of (41), illustrating how the artificially generated datasets can be used in problems where obtaining real data is impractical or too challenging.

In this context, it is one of our goals to evaluate the suitability of synthetic data for improving performance of a model pre-trained on real data. The tool for generating data was an internal work in progress project at Continental Engineering Services based on the Carla (42) driving simulator. Because this tool was not available up until the final stages of these works, the studies made with synthetic data were the last objective.

The synthetic data generation tools can be used to generate surveillance-like images by attaching the point of view to traffic light. This point is chosen because it allows the visualisation of a dynamic scene with plenty of objects, therefore proving numerous positive samples for training or validating the model. We choose to generate data under cloudy, rainy, and sunny weather, but not during the night because there were annotation problems on the used version of the tool. The traffic sign is randomly changed after 100 frames to change the background and object dispositions, effectively increasing the diversity of the dataset. The occlusion of the bounding boxes was also annotated, and only objects with less than 70% of occlusion

were kept. Examples of annotated images are presented in figure 23.

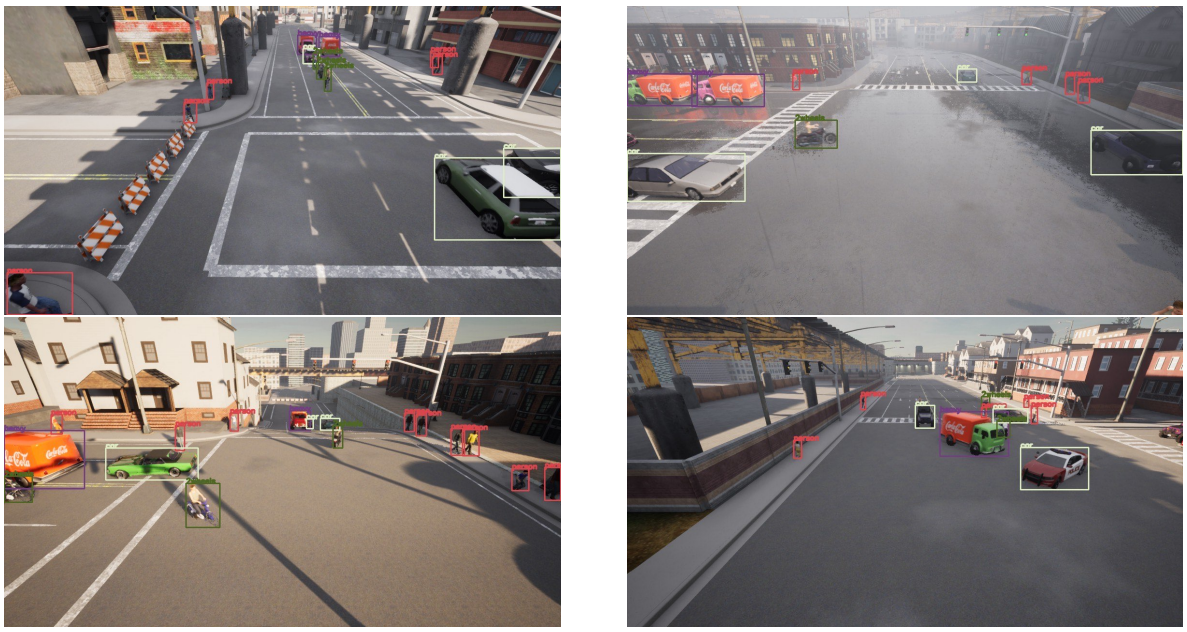


Figure 23: Some samples of the Carla dataset.

The per-category distribution of a sample set is presented in figure 24, but all the generated sets follow similar distributions because the number of loaded elements of each category were manually set. We choose to load more instances of the person category because it was underrepresented on the AAU and BDD dataset, and we only did so for this category to compare the impacts with other categories that remained relatively under sampled.

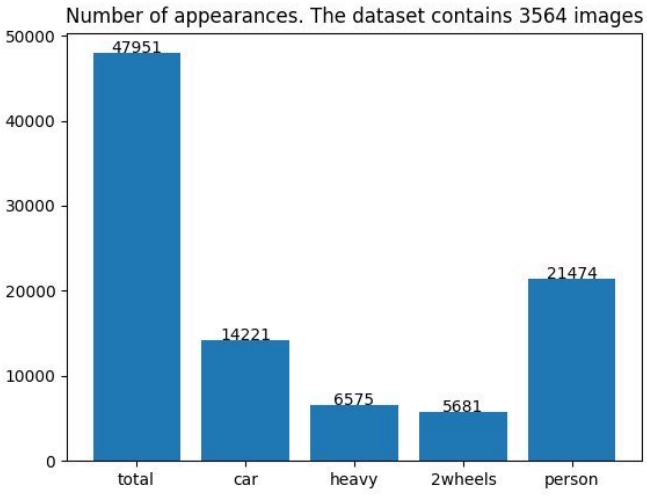


Figure 24: Per-category statistics of a Carla's dataset sample of 3564 images.

3.4 Visdrone

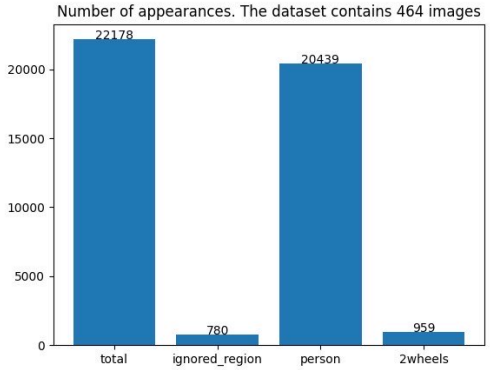
The Visdrone2019(7) is one of the most famous databases on drone-based computer vision. It contains 265k frames and over 10k statics high-resolution outdoor images, containing around 2.6 million annotations of pedestrians, cars, vans, bus, trucks, motorcycle, and bicycles. Its scenes are diverse, covering traffic intersections, parking lots, sports courts, and parks. Because we need more surveillance-like data for testing purposes as explained in section 3.1, we choose to use this dataset. The Visdrone's licence does not allow its commercial use without the authors' consent, which is the reason why we only used it for testing purposes.

One of our goals being to have a general-purpose model that can be used on future works, evaluating the performance in a similar surveillance task from a drone's camera is in phase with our objectives, provided that the camera's angle is inclined downwards, and its altitude is neither too low nor too high (in short, similar to a stationary surveillance camera).

Following this criteria, we choose by visual inspection the sequences 0,1,2 and 6 of the dataset, which are presented in figure 25. These sequences are diverse and challenging due to the great number of object. However, the distribution of the categories are unequal, which it is not really a problem since all the frames will be added to the dataset. The sequence 0, for example, contains no vehicles since it was filmed on a basketball court.

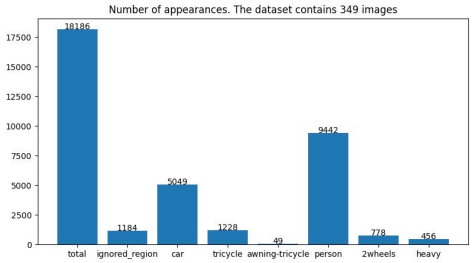
The crowded or hardly visible objects are in "ignored regions", which are annotated. To avoid evaluating our model on these regions and following the benchmarks procedures on (7), we choose not to consider detected objects or annotations that have an intersection of over 50% with an ignored region.

(a) Video 0 sample.



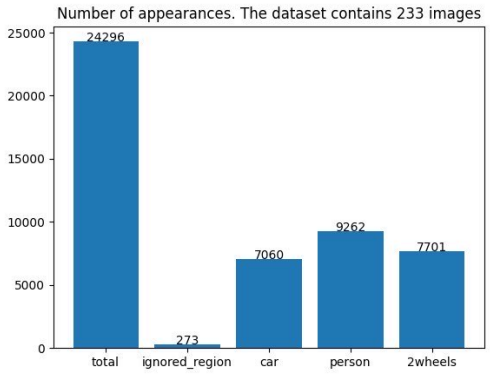
(b) Video 0 statistics.

(c) Video 1 sample.

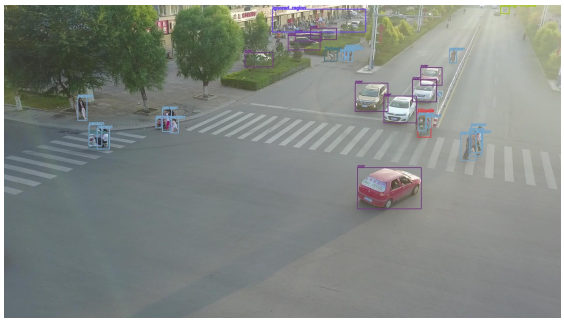


(d) Video 1 statistics.

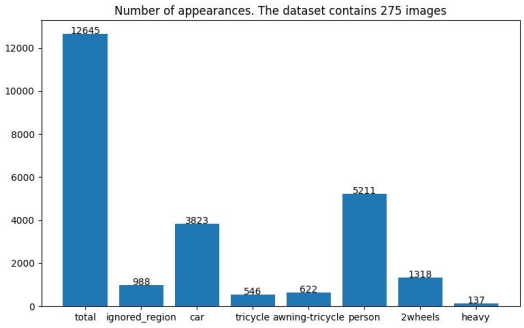
(e) Video 2 sample.



(f) Video 2 statistics.



(g) Video 6 sample.



(h) Video 6 statistics.

Figure 25: Visdrone's videos used for testing purposes, alongside their respective per-category occurrences distribution

3.5 Training and validation set construction

The default datasets for training and validation (splits) purposes are constructed as illustrated in figure 26. Both splits are constructed following the same procedure, the difference being the number of images. We take 1185 images from the re-labelled and filtered AAU dataset for the training and 234 for the validation splits. We add 3 times more data from the BDD dataset to that, to allow the model to keep performance in the driver's point of view and to increase our dataset's size. A summary of the data is presented in table 1

Synthetic data will be added in our experiments by a factor r in function of the total size $N = 5676$ of the dataset: for example, if $r = 10\%$, we will be adding $0.1 * N = 568$ synthetic images, respecting the size proportions of the splits.

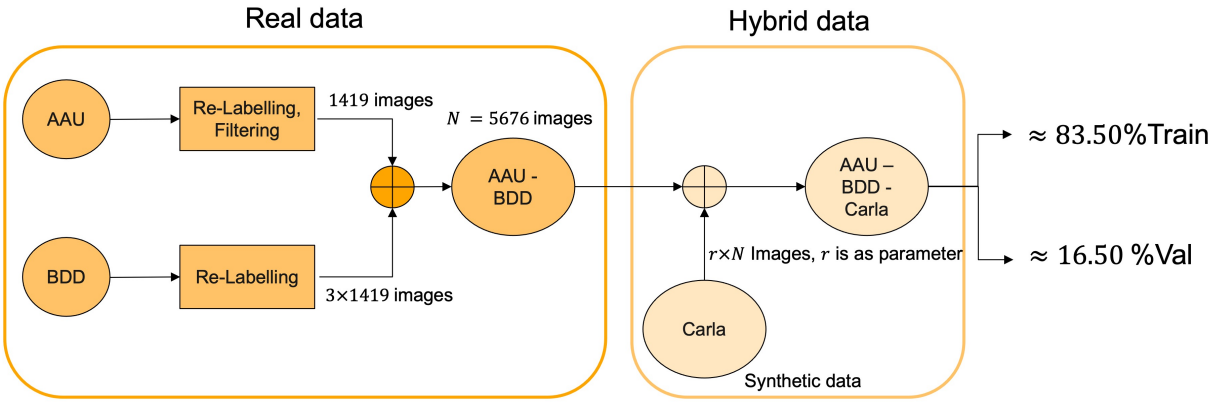


Figure 26: Process of creating the train and validation sets

Table 1: Number of images per split on the train and validation sets.

Split	# AAU images	# BDD Images	# Total Images
Train	1185	3555	4740
Validation	234	702	936
Total	1419	4257	5676

3.6 Test set summary

Our goal is to have a general-purpose test set that grasps the performance from a surveillance-camera view and from a driver's point. The statics of each dataset used to construct our test set are presented in their respective sections. A summary of the number of images in the test set is presented in table 2. We reinforce that fact that, even though we have a lot of sample images from driver's view, the large number of objects in the Visdrone tends to balance this fact.

Table 2: Number of images from each dataset's test split on the final test set

Split	# AAU images	# BDD Images	# Visdrone video 0	# Visdrone video 1	# Visdrone video 2	# Visdrone video 6	# Total Images
Test	223	3777	464	349	223	275	5311

3.7 Data augmentation

The amount of data at our disposal is not huge, making the use of smart data augmentation techniques a central topic. In this matter, object detection is suitable for a diverse set of data augmentation techniques, because most of them do not change the essential characteristics of the target objects. This is not the case for problems like Image Quality Assessment.

The data augmentation pipeline presented in figure 27 was implemented to work with all the Deep Learning frameworks (Keras, Pytorch,...), and therefore is usable in in other projects in development at CES. The colour, geometry and Copy & Paste augmentations will be present in the following sections. Even though it is implemented and shown in figure 27, the Mosaic data augmentation did not show good preliminary results, reason why it will not be analysed in detail.

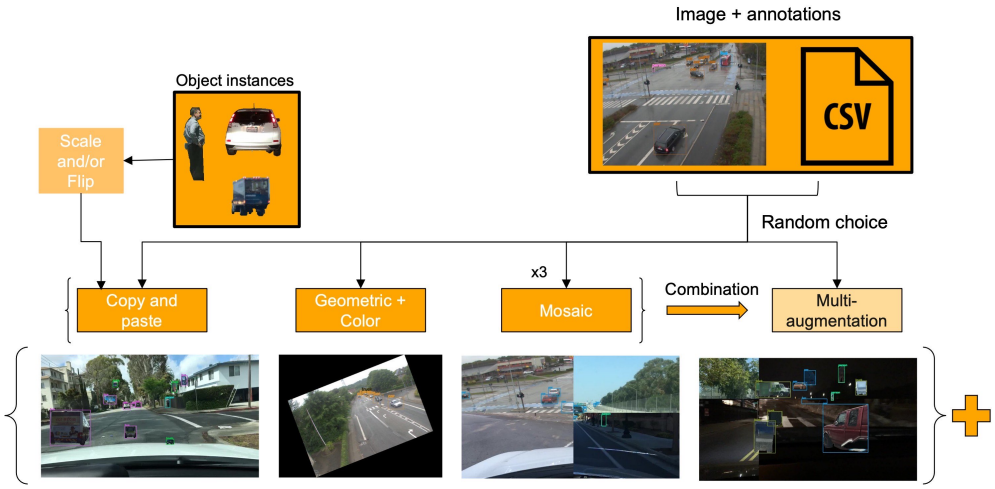


Figure 27: Implemented data augmentation pipeline.

3.7.1 Colour and geometric augmentations

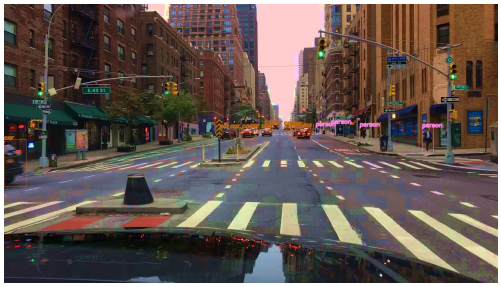
The colour augmentations aims to reproduce the eventual aspects and processing of the cameras, as well as problems they might have during their lifetime. Therefore, presenting images with these variations increases robustness of the neural networks and reduces the chance of overfitting by effectively augmenting the quantity of data it sees during training. Examples of blur, brightness, saturation and gaussian noise transforms are presented in figure 28, and it is visually clear that this variations can happen during a camera's normal work.

Geometric data augmentation is implemented under the same reasons that colour data augmentation is, and examples of horizon flip, random crop, saturation and rotation are presented in figure 29.

(a) Blur (opposite would be sharpening).

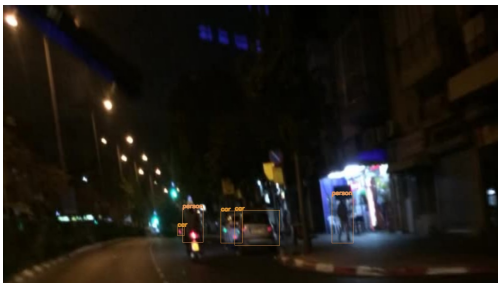
(b) Brightness.

(c) Saturation.

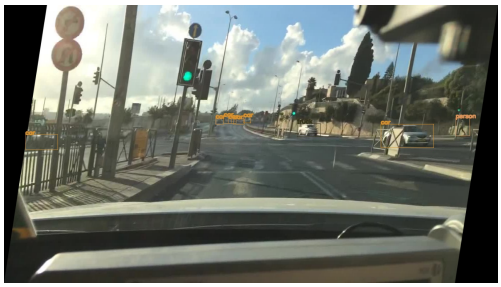


(d) Gaussian noise.

Figure 28: Examples of colour augmentation techniques applied to our images.



(a) Crop.



(b) Mirror, or horizontal flip.

(c) Rotation.

(d) Saturation

Figure 29: Examples of geometric augmentation techniques applied to our images.

3.7.2 Copy & Paste

The deep understanding of how a neural network learns and on what kind of features it focuses is not clear, and it is still objective of active research. In the object detection and instance segmentation task,

some studies assumed context could be important, and tried to carefully place new objects in an image by understating its surroundings (43).

In this context, The Google Brain team proposed the Copy & Paste data augmentation (10) for the instance segmentation task. It consists in simply pasting object instances (pixel-wise annotated) from other images in the training set without context awareness, and greatly changing this instance's total pixel area (large-jitter), as presented in figure 30. By doing so, they consistently improved performance of different algorithms.

This motivated the adaption of the Copy & Paste data augmentation for the object detection problem. However, as our datasets do not contain pixel-wise annotations, we extracted instances from the BDD10k dataset and created an instance dataset of objects without their background in RGBA format. During training time, clearly visible objects (pixel surface are over 1200) were randomly pasted in the images with large jitter and no context awareness. Moreover, we limited the occlusion a pasted object can produced on an already present object in the image by 70%.

The algorithm is fast enough to be used in real-time during the training phase. By using this method, we note that the information in the dataset is increased, since we are pasting instances from another dataset.

Figure 30: Principle of the Copy & Paste Data Augmentation (10).

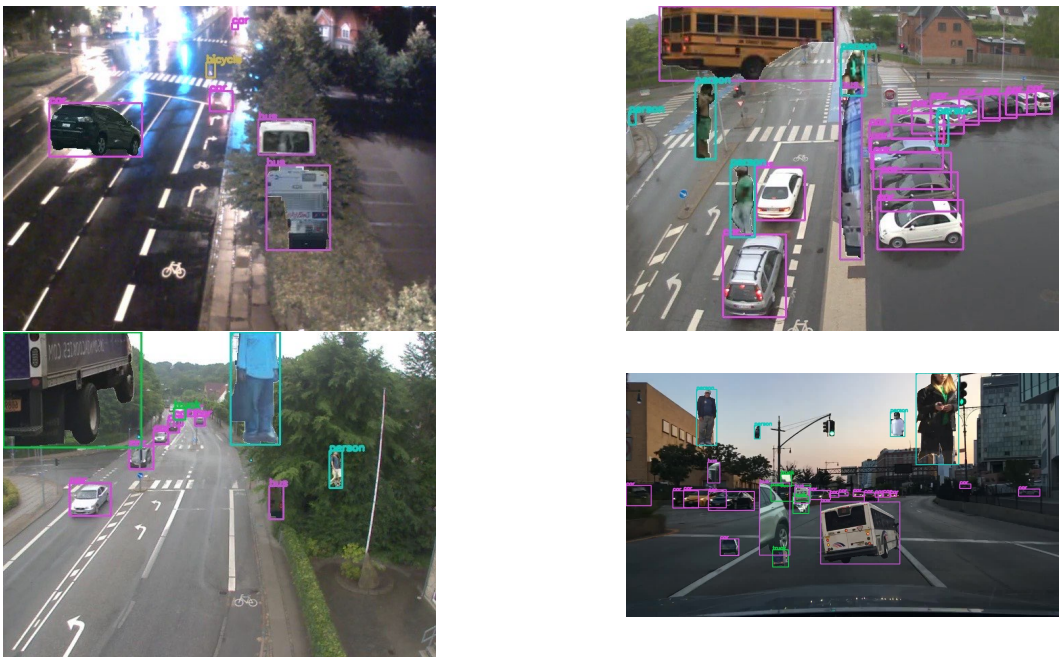


Figure 31: Examples of the Copy and Paste data augmentation applied to our images.

4 WORK DESCRIPTION

In this section, we explore different techniques presented thought these works and propose solutions to the challenges presented. We will evaluate our models with the metrics described in 2.3 and qualitatively whenever pertinent. Considerations on the inference time of our algorithm and proof of concept applications will be presented.

4.1 Summary of training phase parameters

Transfer learning consists of taking features learned on one problem, and leveraging them on a new, similar problem. A last, fine-tuning consists in unfreezing the entire model obtained above (or part of it), and re-training it on the new data, usually with a smaller learning rate. These methods can potentially achieve meaningful improvements, by incrementally adapting the pretrained features to the new data (44). As discussed in previous sections, our approach consists of transfer-learning and fine-tunning a pre-trained Centernet model. We define *frozen backbone* as the step where the backbone's weights are frozen.

The experiments were made used a laptop with a GTX1070 GPU and a i7 2,6 GHz CPU. Furthermore, the training phase parameters are summarised in the list below.

- *Number of epoch*s: empirically, we have seen that our model's losses did not improve much after 20 epochs, value choose as default.
- *Batch-size*: we searched to optimise the use of the GPU's memory, arriving at a batch size of 8 with a frozen backbone, and 4 otherwise.
- *Learning rate*: 1×10^{-5} for frozen backbone models, 1×10^{-4} otherwise.
- *Learning rate scheduler*: reduce on Plateau, patience of 0 epochs with a multiplication factor of 0.8.
- *Training and validation sets*: as specified in section 3.5. By default, no synthetic data is used.
- *Input image resolution*: 1152×640 pixels. We need to use input resolutions multiples of 32 because of the feature combinations in different levels of the Deep Layer Aggregation backbone architecture. We note that preliminary tests with a 512×512 resolution gave poor results.
- *Optimiser* : ADAM, as it is the best optimiser in general. We did not study the use of alternatives (i.e., Stochastic Gradient Descent).
- *Basic prepossessing*: normalisation using the MS-COCO's RGB mean and standard deviation, as it is common practice when used a pre-trained backbones on this dataset.
- *Loss weights*: 0.1 for regression head, 1 for offset head and 1 for the heatmap head of Centernet. We kept the standard values found in the experiments of the original authors (27).
- *Data augmentation*: by default, only colour and geometric data augmentations are used. Each image has an equal chance of being augmented of not. There is a 30% per cent chance that each colour or geometry augmentation will be applied to the image.

4.2 Summary of testing phase parameters

- *IoU threshold*: at least 50% to classify a detection as true positive.
- *Test set*: by default, the combination of AAU, Visdrone and BDD as described in section 3.6.
- *Model naming*: the base model is the ID #0. Subsequent models are IDs # of 1.x, 2.x, etc. The models #1.x were trained with the initial weights of the model ID #0, and so on. Exceptions are explicated in their respective sections.

4.3 Impact of category diversity reduction

The base model was trained partially on the BDD100k dataset and some CES's proprietary data to detect buses, traffic lights, traffic signs, people, bikes, trucks, motorcycles, cars, trains, and riders. For the reasons explained in section 1.2, our experiment consists in changing the heads of this model to detect cars, heavy vehicles, two-wheel vehicles and people. Only the weights of the backbone are kept.

Intuitively, we expect that by requiring less categories the model would perform better as it needs to learn fewer specific features. However, this is not necessarily true, as less categories could also mean that the model would lose its ability to distinguish among them. Moreover, most real world applied machine learning algorithms only distinguish a small number of categories.

The results on the default test are presented on table 3. This being the first test done in the original model, we note that the performance is unequal for the categories, being especially low for Person and 2-wheel and high for the car category. This is expected from the remarks on distribution and bounding box size of the BDD100k dataset used in the training phase of this base model.

The ID# 1 model was transfer-learned and fine-tuned from the original model. Its performance is consistently better than the base model's for nearly all categories, and notably in the general metric, the mAP. However, this improvement could have been tied to the use of the AAU in the training phase, meaning the test set is now more similar to train set, even if the video sequences are not the same.

To exclude this possibility, we present on table 4 the performance on the BDD test set, that is, only on images from a driver's point of view. We still observe an increase in performance, meaning that it is in fact due to the transfer-learning and simpler target categories.

Table 3: Number of categories reduction. The fine-tuned model used the original model's weights as starting point, except for the head weights.

			Person		Car		Heavy		2-Wheels		General	
ID	Base model	Train/Val Dataset	AP	F1	AP	F1	AP	F1	AP	F1	mAP	wF1
0	-	BDD100k	21,39%	32,09%	70,60%	72,39%	62,07%	64,00%	5,87%	14,22%	39,98%	58,08%
1	0	AAU-BDD	29,88%	40,78%	77,92%	76,72%	62,67%	63,93%	12,98%	23,55%	45,86%	58,88%
			+ 8,5%	+ 8,7%	+ 7,3%	+ 4,3%	+ 0,6%	-0,1%	+ 7,1%	+ 9,3%	+ 5,9%	+ 0,8%

Table 4: Number of categories reduction. The Fine-tuned model was used the Original Model's weights as starting point, except for the Heads. Test set is BDD100k

			Person		Car		Heavy		2-Wheels		General	
ID	Base model	Train/Val Dataset	AP	F1	AP	F1	AP	F1	AP	F1	mAP	wF1
0	-	BDD100k	70,90%	70,78%	81,93%	80,28%	69,23%	67,92%	37,96%	48,65%	65,00%	74,95%
1	0	AAU-BDD	70,31%	70,83%	82,81%	80,90%	70,39%	68,21%	54,44%	60,53%	69,49%	74,09%
			-0,6%	+ 0,1%	+0,9%	+0,6%	+ 1,2%	+ 0,3%	+ 16,5%	+11,9%	+4,5%	-0,9%

4.4 Impact of flip inference

To boost performance on the Centernet model, the authors (27) propose the use of flip-inference, which consists in feeding the network with both the original and horizontally flipped image in the testing phase. The mirrored heads are horizontally flipped and their outputs are averaged with the original heads' to obtain the detected objects, as shown in figure 32. Since the network is a non-linear function, feeding the mirrored image does not mean the output heads are simply the mirrored versions of the regular output heads.

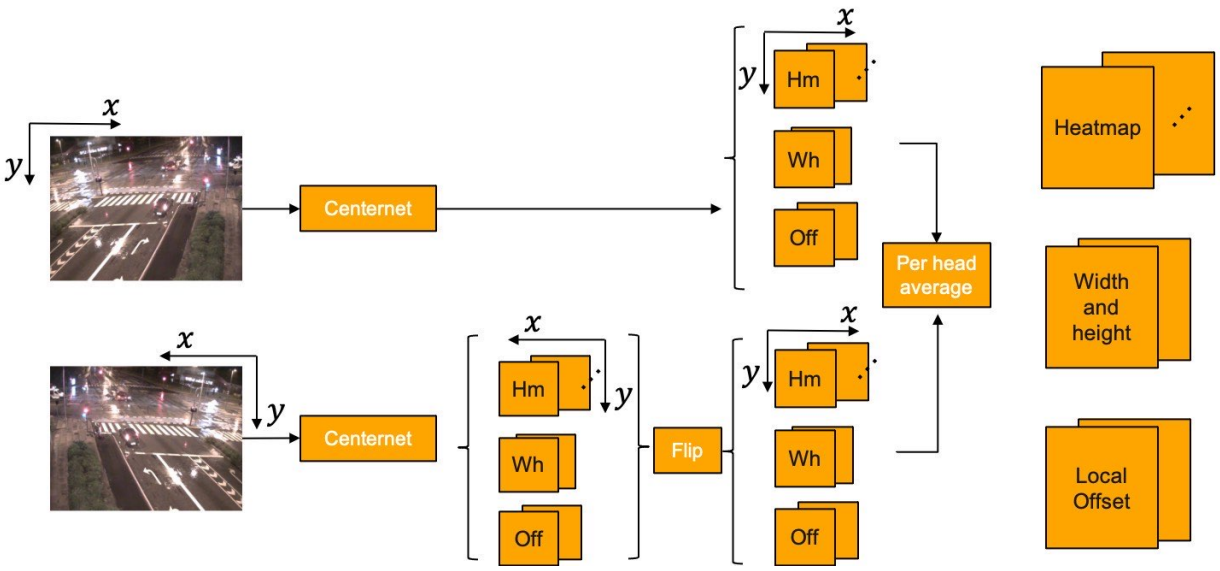


Figure 32: Flip inference pipeline.

Using the model previously obtained, we study the impact of using flip-inference. The results for the ID #1 model are presented in table 5, showing consistent gains in all metrics for all categories. However, gains not that big may not be worth in scenarios where computational resources are scarce, since we are effectively making 2 times the same inference. This operation can be done in parallel when a GPU is used (see section 4.8). For these works' purposes, following inferences will be made using flip-test.

Table 5: Flip-inference test.

		Person		Car		Heavy		2-Wheels		General	
ID	Flip-Inference	AP	F1	AP	F1	AP	F1	AP	F1	mAP	wF1
1		29.88%	40.78%	77.92%	76.72%	62.67%	63.93%	12.98%	23.55%	45.86%	58.88%
1	X	31.48%	42.49%	78.66%	77.31%	65.94%	66.69%	13.82%	24.52%	47.47%	59.96%
		+1.6%	+1.7%	+0.7%	+0.6%	+3.3%	+2.8%	+0.8%	+1.0%	+1.6%	+1.1%

4.5 Impact of 4-fold patch inference

The performance on small objects is not satisfactory, which is a common problem in object detection. To deal with the issue, we tried to split the image in 4 patches, resize those to the original image's size and make inferences on each of the patches. In the end, the inferences are joined, as shown in the complete pipeline presented on figure 33.

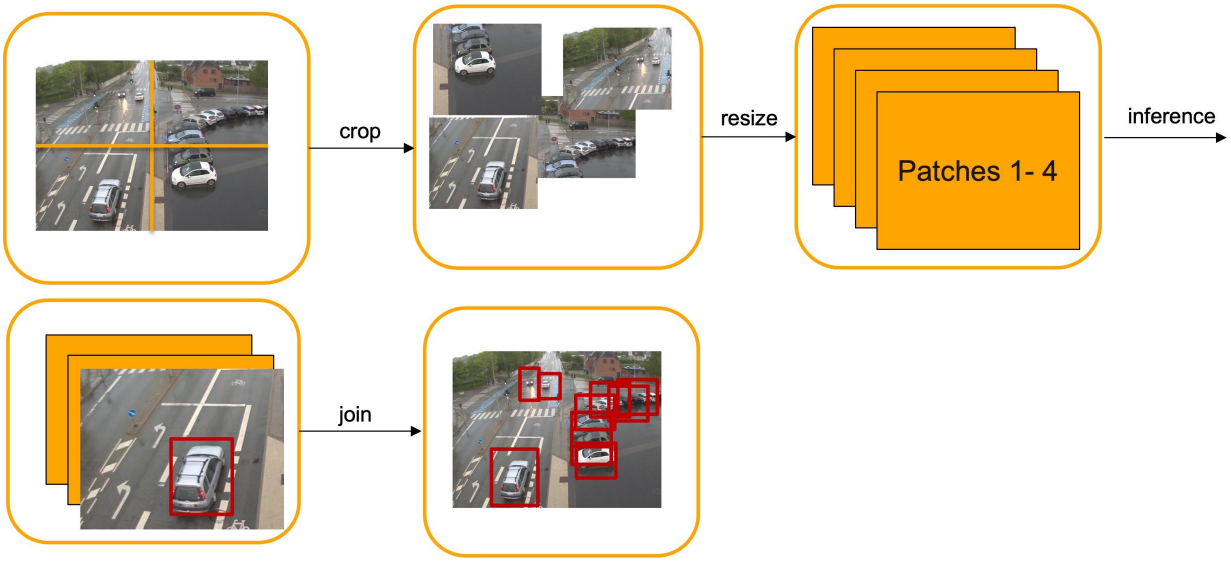


Figure 33: 4-patch inference pipeline.

In table 6 we show the results of the experiments. They are catastrophic, showing huge losses in performance, especially in big objects like those on the heavy and car categories. One possible explanation could be that the objects are split between patches, meaning they would be only partially present in each patch.

To better investigate the reason of this results, we qualitatively analysed these inferences in figure 34 in comparison with the regular inference method. Even though we do not prove that some bad results come from the above hypothesis, we observe some small objects in the image's corners are not detected, and also that some false positives are created. Therefore, the simple fact of "zooming" in these regions by using the crops do not help the model detect smaller objects, but indeed weakness this capacity. One possible explanation resides in the use of Feature Pyramid Networks on the Deep Aggregation Layer backbone already accounting for multi-scale features, and therefore the use of patches would simply mean less information to the network

Based on this analysis, we chose to not use this method anymore.

Table 6: Results of the 4-patch inference on model ID # 1.

		Person		Car		Heavy		2-Wheels		General	
ID	4-Patch inference	AP	F1	AP	F1	AP	F1	AP	F1	mAP	wF1
1		31.48%	42.49%	78.66%	77.31%	65.94%	66.69%	13.82%	24.52%	47.47%	59.96%
1	X	20.84%	30.79%	52.56%	57.41%	11.30%	25.89%	15.40%	29.90%	25.03%	43.67%
		-10.6%	-11.7%	-26.1%	-19.9%	-54.6%	-40.8%	+ 1.6%	+ 5.4%	-22.4%	-16.3%

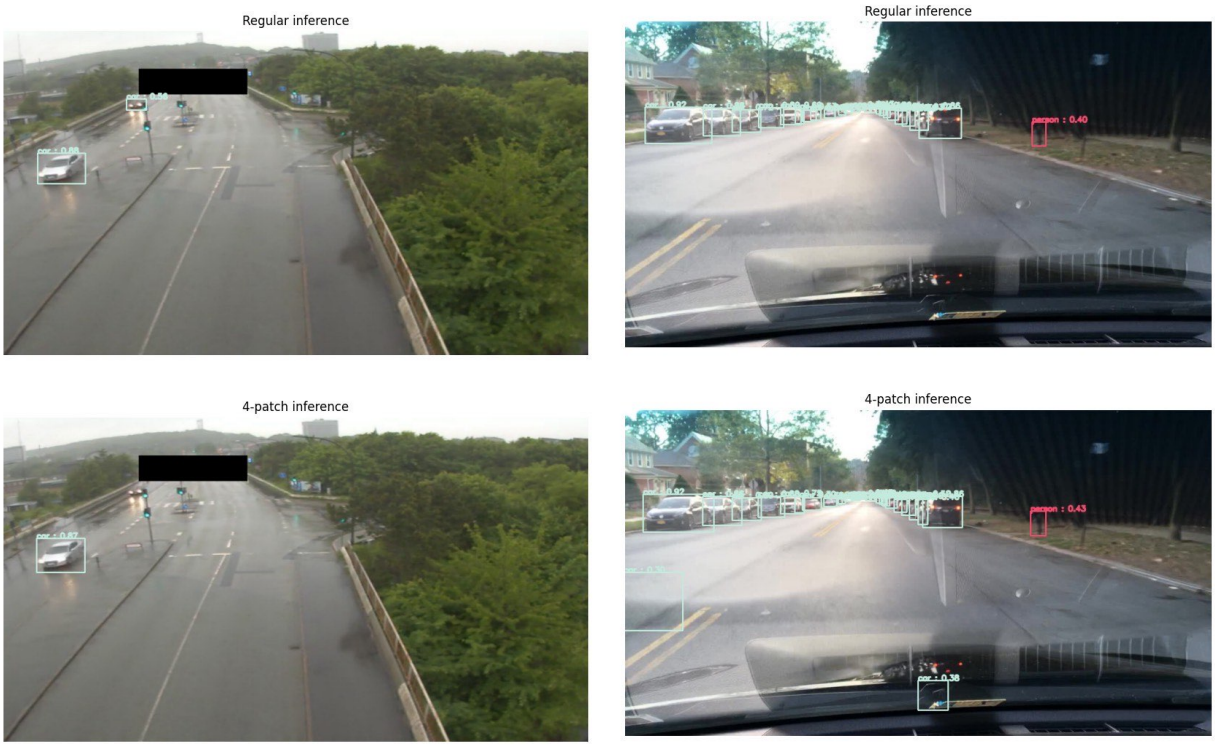


Figure 34: 4-patch inference visual inspection.

4.6 Investigating the performance on the Person category

Having established the a few heuristics on the inference configuration and fine-tuning, we now search to increase performance on the Person category. We hope to develop techniques that can be applied to small and underrepresented categories, like the two-wheel.

4.6.1 Positive samples, transfer-learning and person heatmap

We seek to investigate the effect of focusing on positive samples by using only images with at least one annotated person on it, both in the training and validation sets. This effectively reduces our dataset's size but can potentially show the importance of higher quality data.

In parallel, we also investigate the effect of only using the Person Heatmap's head to calculate the validation and training losses. This means that we are optimising the network's weights to person detection, potentially meaning a performance drop in other categories. The procedure is illustrated in figure 35. We do not freeze the weights corresponding to other categories, we simply do not take their outputs into account to calculate the loss function:

$$L_k = L_c|_{c=\text{Person}} = \frac{1}{N}\sum_{xy}\begin{cases} -(1-\hat{Y}_{xy})^{\alpha} log(\hat{Y}_{xy}) & \text{if } Y_{xy}=1 \\ -(1-Y_{xy})^{\beta}(\hat{Y}_{xy})^{\alpha} log(1-\hat{Y}_{xy}), & \text{otherwise} \end{cases} \tag{13}$$

We also experimented on simply using larger weights on the person category, without effectively ig-

noring the other categories, but preliminary test led us to not pursue with this approach.

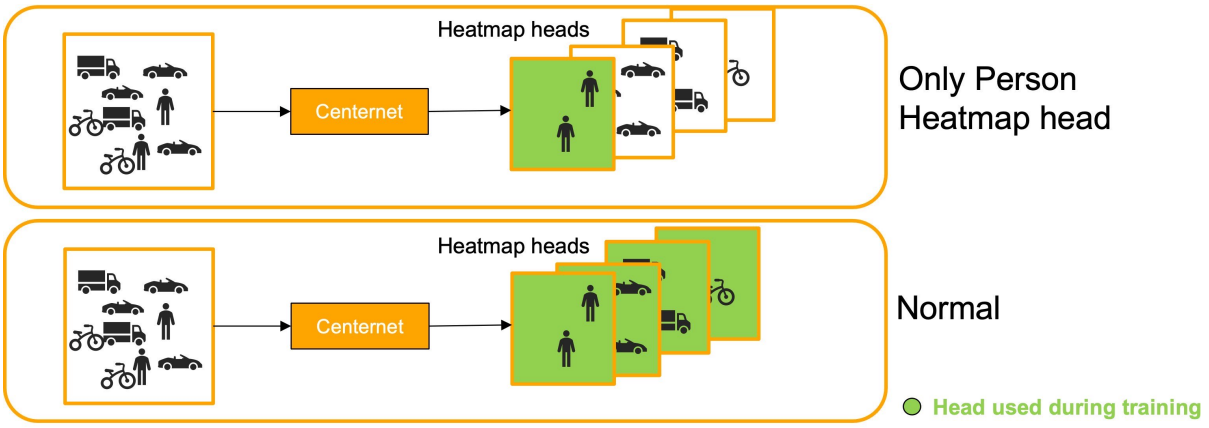


Figure 35: Only person head: using only the heatmap corresponding to the Person category for loss computation.

Training the network by considering only the Person Heatmap head could mean that the performance on other categories would drop because we are adapting the weights of all the heads to improve performance on only one category. Therefore, one could think that freezing the weights in backbone could avoid this issue. The impact of doing so is also studied.

We proceed as follows: starting from model ID #1, we freeze the weights of the model's backbone and study the effects using only the Person's Heatmap and only frames with at least one person annotated, separately and combined.

By analysing the results in table 7 for models ID# 1, 2.0, 2.1 and 2.2, we observe that combining these methods improves performance on the person category, even if the person Heatmap head does the opposite when used individually. In addition, the performance in other categories has not decreased, leading us to repeat the experiment without the frozen backbone. We archive a 5.85% increase in Person AP, and even some slight increase in order categories' performance. Therefore, focusing on one category at a time can increase performance, and this procedure could be repeated individually for each of the categories, specially the under sampled ones.

Table 7: Ablation study of using only frames for person, only the person's Heatmap head in centernet and freezing the weights in the model's backbone.

				Person		Car		Heavy		2-Wheels		General	
ID	Only Person Frames	Only Person Heatmap Head	Frozen Backbone	*AP*	*F1*	AP	F1	AP	F1	AP	F1	mAP	wF1
1				*31.48%*	*42.49%*	78.66%	77.31%	65.94%	66.69%	13.82%	24.52%	47.47%	59.96%
2.0	X		X	*33.01%*	*42.92%*	78.59%	77.26%	65.79%	66.53%	13.20%	25.66%	47.65%	58.60%
2.1		X	X	*29.21%*	*40.63%*	78.58%	77.00%	66.21%	67.07%	12.98%	23.28%	46.75%	60.84%
2.2	X	X	X	*33.55%*	*43.73%*	78.50%	77.03%	66.21%	67.21%	14.18%	25.42%	48.11%	58.27%
2.3	X	X		***36.83%***	***46.31%***	78.53%	77.38%	66.31%	66.99%	15.18%	26.96%	49.21%	58.87%

4.6.2 Prioritising small bounding boxes and Copy & Paste augmentation

The study of the statistics of our datasets have shown that the more frequent categories are coincidentally those with smaller bounding boxes: person and 2-wheel vehicles. This leads to the hypotheses that

the training might be leading the model to prioritise larger bounding boxes as these are more frequent.

Therefore, one idea is to forcefully prioritise smaller bounding boxes, as shown in figure 36.

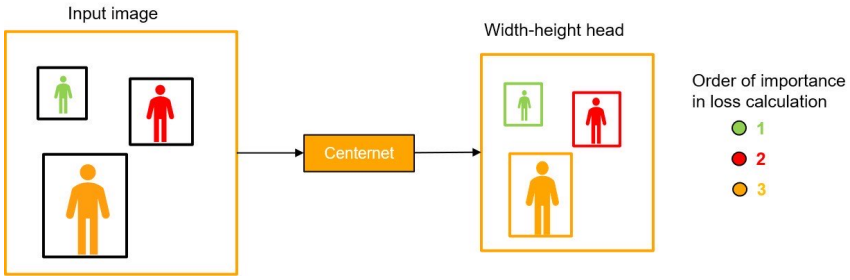


Figure 36: Illustration of the wL1 loss, giving more importance to the smaller bounding boxes.

We propose to adjust the width-height loss function to counter this effect by using a weighted L1 loss ($wL1$). Originally, this it is a simple L1 loss between the $K \times 2$ (we detect only the top $K = 100$ objects) matrices defining the width and height of the ground truth (GT) bounding boxes and detected bounding boxes (DET), as defined in equation 6. The introduced weights are the inverse of the ground truth bounding box surface area, represented by $\frac{1}{h_i^{GT} w_i^{GT} + \epsilon}$ in equation 14. The ϵ and 1 is added to avoid a zero division.

$$wL1 = \frac{1}{\sum_{i=0}^{K-1} h_i^{GT} w_i^{GT} + 1} \sum_{i=0}^{K-1} \frac{(|w_i^{GT} - w_i^{DET}| + |h_i^{GT} - h_i^{DET}|)}{h_i^{GT} w_i^{GT} + \epsilon} \tag{14}$$

At the same time, we also investigate the impact of using Copy & Paste data augmentation (section 3.7.2), pasting instances of objects in images presented to train the network. By doing so we expect to increase positive samples presented to the model. An ablation study of the parameters would be extremely time consuming, and therefore we empirically determined the following:

- occlusion threshold of 30% with the present objects in the image;
- between 1 and 6 randomly pasted objects for the person category, and between 1 and 3 for the other categories;
- between 0.2 and 1.8 jitter (scale change);
- 30% probability of pasting each category.

We use the best model from the previous section (ID #2.3) as a starting point as we keep using its configuration (Only Person Frames and Only Person Heads). The results presented in table 8 show that the use of $wL1$ loss decreases performance both when used individually and alongside copy & paste data augmentation, so it is not a good approach.

We also show that copy and paste augmentation promotes a 7.04 AP increase on the person category, which is the target of this experiment. Therefore, pasting instances without additional concern for the background is a good approach not only in the instance segmentation task, as shown in (10), but also the object detection task in the context of this project.

Table 8: Ablation study of weighted L1-loss (wl1) and Copy & Paste data augmentation. All experiments are using only frames with people, and only the person's Heatmap head in centernet is considered in the loss.

				Person		Car		Heavy		2-Wheels		General	
ID	Base model	wL1 Loss	Copy & Paste	AP	*F1*	AP	F1	AP	F1	AP	F1	mAP	wF1
2.3	1.0			*36.83%*	*46.31%*	78.53%	77.38%	66.31%	66.99%	15.18%	26.96%	49.21%	58.87%
2.4	*2.3*	X		*34.37%*	*43.93%*	78.03%	76.95%	64.96%	66.77%	13.66%	25.34%	47.75%	58.71%
2.5	*2.3*		***X***	***43.87%***	***52.21%***	78.28%	76.99%	63.58%	64.66%	17.64%	31.06%	50.84%	59.42%
2.8	*2.3*	X	*X*	*42.02%*	*50.59%*	78.33%	77.20%	65.17%	66.55%	11.59%	25.89%	49.28%	58.86%

4.7 Impact of synthetic data

Now that we have managed to augment performance on the person category, that is, one of the most important under sampled categories, we will experiment using synthetic data.

We aim to increase performance on all categories, and therefore we use the full training and validation sets (including images without people) and we consider all the categories on the loss computation. In other words, we drop the only images with people and only person Heatmap head introduced in the previous section. However, copy and paste data augmentation is kept.

The dataset is composed as described in section 3.5), and the percentage of synthetic data added is relative to the original dataset size as explained in section 3.5.

The results shown in table 9 prove that we can indeed increase performance in a general sense with synthetic data. As observed in figure 37, the general tendency is a steady increase performance up until 50% per cent of added data, and the performance oscillates and slightly decreases after this point. This proves that the synthetic data is indeed correlated with the real data of the test set, making it suitable to the project.

Table 9: Evolution of model's performance withe the increase of synthetic data in the dataset.

			Person		Car		Heavy		2-Wheels		*General*	
ID	Base model	% Synthetic data added	AP	F1	AP	F1	AP	F1	AP	F1	*mAP*	*wF1*
2.5	2.3	0%	43.87%	52.21%	78.28%	76.99%	63.58%	64.66%	17.64%	31.06%	*50.84%*	*59.42%*
3.0	2.5	10%	42.25%	51.22%	79.05%	77.59%	66.03%	66.65%	17.18%	29.44%	*51.13%*	*62.18%*
4.0	3.0	25%	41.47%	51.35%	79.32%	78.12%	64.74%	64.70%	20.83%	32.63%	*51.59%*	*60.96%*
5.0	4.0	50%	42.21%	51.64%	79.51%	78.14%	64.85%	63.10%	21.65%	33.39%	***52.05%***	*60.66%*
6.0	5.0	75%	42.65%	51.77%	79.53%	78.16%	63.60%	62.87%	21.60%	33.58%	*51.85%*	*60.73%*
7.0	6.0	100%	43.13%	52.09%	79.68%	78.34%	63.51%	62.45%	21.35%	33.68%	*51.92%*	*60.84%*
8.0	7.0	125%	43.22%	52.18%	79.48%	78.23%	63.72%	62.97%	21.18%	33.36%	*51.90%*	*60.66%*

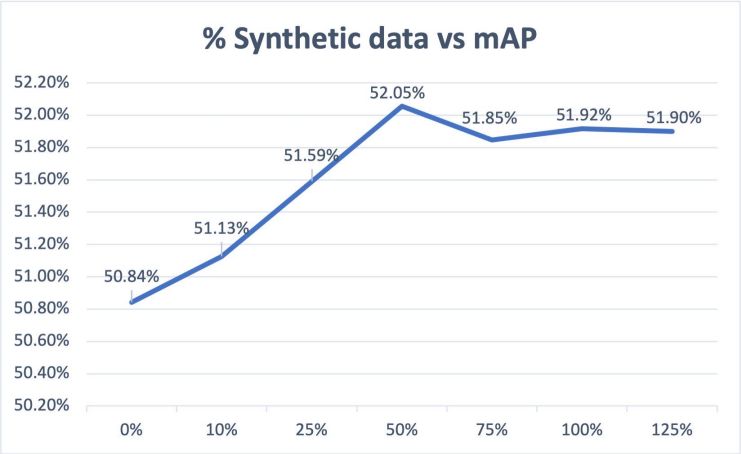


Figure 37: Evolution of mAP (x-axis) with the use of more synthetic data (y-axis).

To better investigate the reasons why the 50% of synthetic data is the performance threshold, we test our models on a combination of the AAU and Visdrone test sets, therefore only having surveillance-like images.

The results are shown in table 10 and on figure 38. We observe that the best model for surveillance is still using only 50% per cent of synthetic data, which is coherent with the previous results. However, we also note that the tendency of increasing performance after 125% of synthetic data is stronger than before, suggesting that adding more data of this type would increase performance even more.

This can be explained by the closer similarity between this test set and the synthetic data, as both contains only surveillance-like images. Therefore, the model is arriving at a point where performance from a surveillance camera point of view increases at the expense of a decrease in performance on the driver's point of view.

In short, synthetic data is useful to increase the model's performance on the surveillance task while keeping the performance on the driver's point of view. However, from a certain quantity of data (in our case, 50% of the real dataset), performance increases in one task at the expense of the other.

Table 10: Evolution of model's performance withe the increase of synthetic data in the dataset. Results for the AAU-Visdrone test set, with only surveillance images.

			Person		Car		Heavy		2-Wheels		*General*	
ID	Base model	% Synthetic data added	AP	F1	AP	F1	AP	F1	AP	F1	*mAP*	*wF1*
2.5	2.3	0%	42.15%	52.52%	67.93%	68.88%	45.17%	53.05%	15.79%	29.97%	*42.76%*	*51.74%*
3.0	2.5	10%	39.77%	50.78%	69.88%	70.83%	51.79%	57.38%	15.27%	28.19%	*44.18%*	*51.59%*
4.0	3.0	25%	39.70%	51.00%	68.54%	69.66%	50.71%	58.15%	14.22%	27.51%	*43.29%*	*53.13%*
5.0	4.0	50%	39.61%	51.42%	71.52%	71.96%	49.96%	50.87%	20.21%	32.35%	***45.33%***	*51.66%*
6.0	5.0	75%	40.12%	51.59%	71.76%	72.22%	45.52%	51.10%	20.15%	32.50%	*44.39%*	*51.98%*
7.0	6.0	100%	40.67%	51.95%	72.43%	72.77%	45.88%	51.47%	20.00%	32.77%	*44.74%*	*60.84%*
8.0	7.0	125%	40.82%	51.96%	72.04%	72.51%	47.33%	53.53%	19.86%	32.54%	*45.01%*	*60.66%*

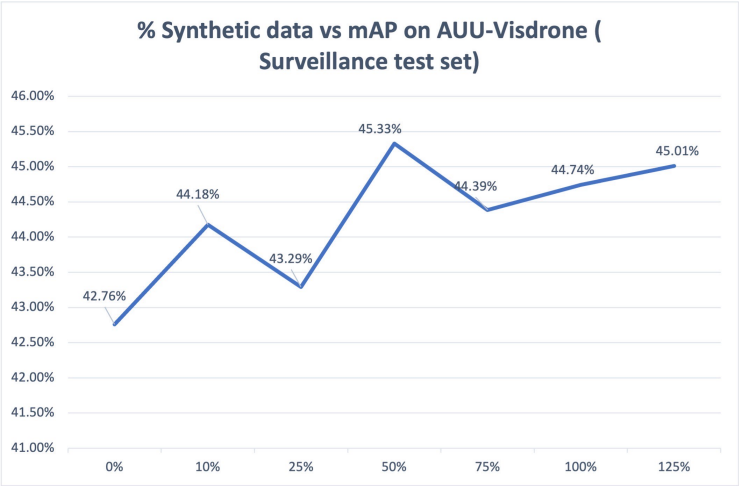


Figure 38: Evolution of mAP with the use of more synthetic data on the AAU-Visdrone teset set, that is, only surveillance images.

4.8 Inference time

The frames-per-second (FPS), calculated over 10 frames, on a CPU and GPU, as well as with and without flip-inference, are presented on figure 11. On the CPU, the FPS is quite low, with an average 1 inference every 5 seconds with regular inference, and the twice as slow with flip-inference. Therefore, it is usable on surveillance applications where a 5 to 10 seconds delay is acceptable. These results were obtained on a Python implementation and should be better on a C++ implementation of the CES framework.

With the use of a GPU (GTX1070 mobile), the results are real-time. As expected, parallel calculation avoids doubling the inference time when using flip-inference.

Table 11: Frames-per-second (FPS) for different inference configurations.

Device	Flip-test	FPS
CPU	Yes	0.21
	No	0.45
GPU	Yes	19.57
	No	22.5

4.9 Qualitative error analysis

The model tuned is a base model for other applications, and therefore it needs tuning on task-specific datasets before being deployed. For this model, we present the most common errors that should be treated in future projects, and suggested approaches for doing so.

The figure 39 presents a sequence of 3 frames where an object was detected in the first and last frame, but not in the middle one. This is a recurrent problem in image object detection, as the network is not aware of the time relation between frames. Therefore, tracking objects with post-processing is proposed in future works, at the expense of a longer inference time.

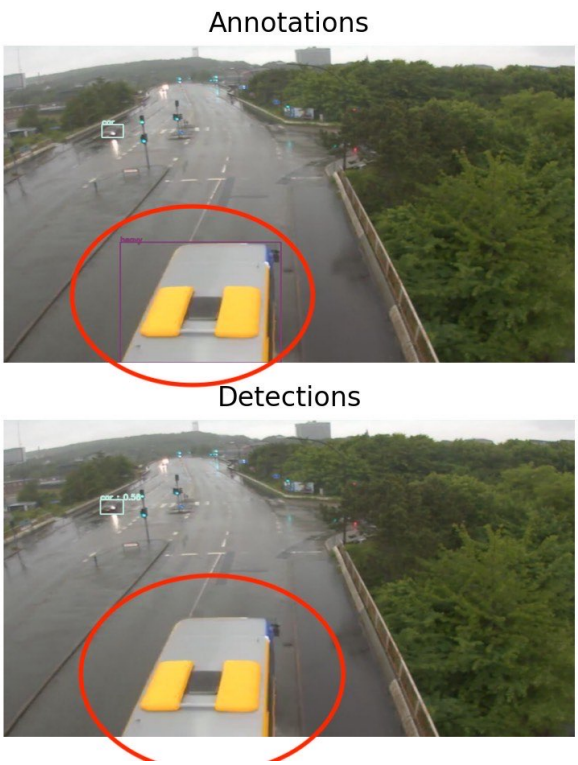
In figure 40 we observe some static inference errors. Most errors are due to the size of the object viewed from the camera's perspective, which is a common problem in object detectors, and one that we alleviated but not solved in this project. Moreover, errors due to occlusion and out-of-frame objects are common. The proposed approach to follow is the use of synthetic data with per-case focus. In case of the small object problem, we note that this is still a limitation of the model.

(a) 1.

(b) 2.

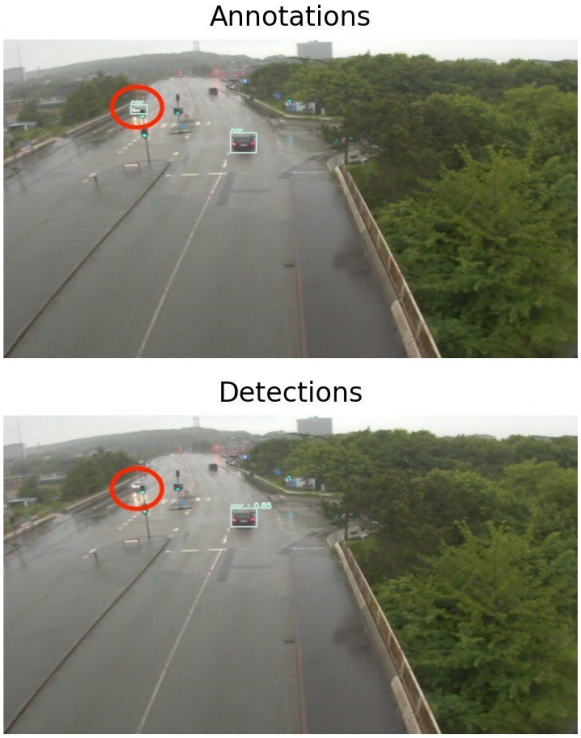
(c) 3.

Figure 39: Object flickering.



(a) False negative due to out-of-frame object

(b) False negative due to distance and object size.



(c) False negative due to occlusion.

Figure 40: Some weaknesses of the current model.

4.10 Application: parking space matching

A simple application of matching vehicle detection and parking spaces is presented in this section. The goal is to prove the concept for future works on the project.

Given a video sequence from a static camera of a parking lot whose parking spaces are annotated, the goal is to determine if each parking space is occupied.

We used the PKlot dataset (11), which contains 12,417 annotated images from 3 different parking lots on sunny, cloudy and rainy weather conditions as illustrated in figure 41. The annotations are the parking spaces and if they are occupied or not.

Figure 41: Example of PKlot images for the UFPR04(a,b and c), UFPR05(d,e and f) and PUCPR(g,h and i) parking lots. Source: (11).

By using our model, the following algorithm is proposed to classify each of the parking spaces:

1. detect the target categories on the image (in our case, with Centernet);
2. using a first(low) confidence threshold(in our case, 0.05), make inferences on the frame. Sort the detections from highest to lowest confidence score;
3. set all parking spaces as *unmatched*;
4. for every detection that correspond to a vehicle, loop thought the unmatched parking spaces. If the IoU between their bounding boxes is higher than a chosen threshold (50% in our case):
    4.1. change the detection's bounding box to the parking spot's one;
    4.2. set its corresponding confidence score to 1.0;
    4.3. add the 'parked' prefix to the category name;

4.4. set the parking spot as matched.

5. apply a second(high) confidence threshold (in our case, 0.3) to the remaining detections;

6. append the unmatched parking spaces to the detections.

Samples of the algorithm's output are presented in figure 42, where the blue boxes represent occupied parking spots, contrary to the green ones. Even tough, we can see some errors, it works as expected. In table 12, we compare our model to methods proposed in the Pklot paper. The accuracy of our solution is not at as high, but it performs well (near 89%) in the UFPR05 parking lot, which has a lower point of view than the others. This is expected since this is the kind of problem the model was tuned to.

There is room for great improvements for this proof-of-concept solution. A first step would be to use the temporal information by tracking the changes between frames, as described in the Qualitative Error Analysis section. Further developments could use a classifier, that in its turn could use the model's backbone, to pass the individual patches of parking spaces, therefore increasing the solution's robustness.

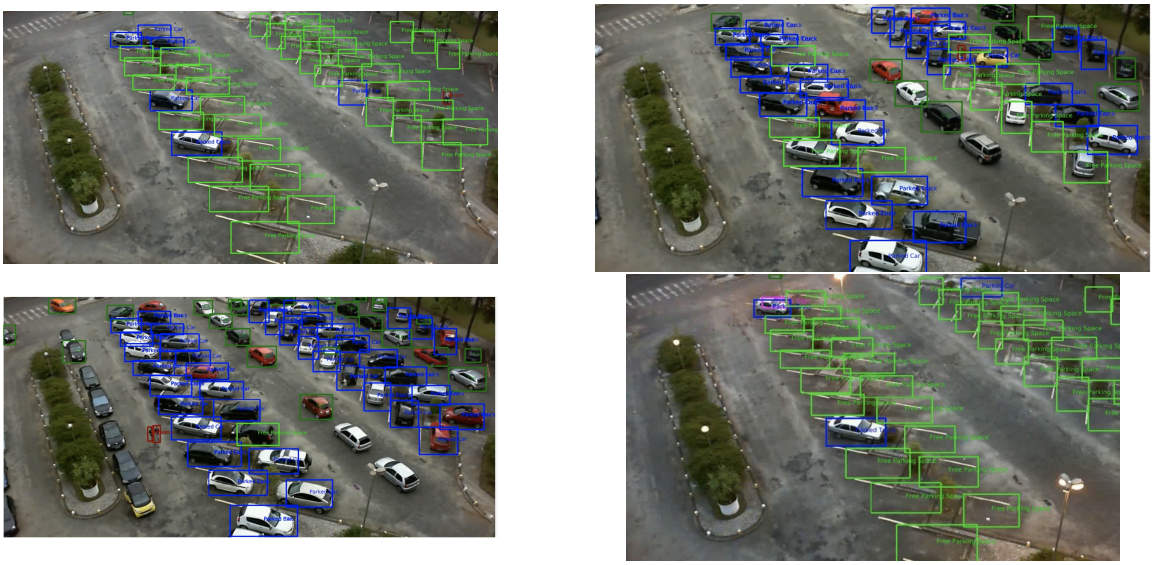


Figure 42: Examples of the algorithm's output.

Table 12: Comparison between our simple parking space detector and counter and the best descriptors described in (11).

	UFPR04	UFPR05	PUCPR
Ours	80.36%	89.00%	80.67%
Best in (11)	99.50%	95.95%	97.13%

5 CONCLUSION

This internship at Continental Engineering Services was a challenging task of starting the future environment monitoring project based on Artificial Intelligence. Therefore, during these works we proposed basis of comparison, code, material, and datasets, as well as results, heuristics, and approaches to follow on surveillance-like applications, and our defined objectives were attained.

Data is central to every machine learning problem. We studied different datasets, explaining their role and how they are suitable for the problem. The low availability of datasets allowing its commercial usage is a challenge for the development of surveillance products, and therefore correctly adapting databases from other tasks, as well as using data augmentation, has proven to be crucial. Even in face of those challenges, we managed to construct a strong and diverse dataset containing scenes from a driver's, surveillance camera's and drone's point of view.

As with every new project, techniques from other domains were explored, and some were proven to be beneficial to the task. For example, Copy & Paste data augmentation (10) showed promising results on the object detection for surveillance problem, and so did synthetic data. For improving performance on under sampled categories, the training on each individual categories at a time and providing more positive samples, that is, images containing the category in study, proved to be important. On the other hand, introducing weights on the sizes of objects and using smaller image patches for inference were not effective for improving performance on small objects, even if those were intuitive approaches.

This research was done with its deployment into a product in mind, and the objectives reflect that. The models developed are ready to be fine-tuned to other applications, very possibly by using the techniques described. With these approaches, we archived nearly 12% improve in mAP relative to the base model, and we did so while focusing on the hard task of Person detection. We hope that these tools can be used in future products of CES, and for doing so a proof-of-concept application was developed, tested, and possible improvements were proposed.

It would be unfair not to state our flaws, and among them is the lack of qualitative error analysis through the studies, which could have been useful to better focus the generation of synthetic data on the algorithm's weak points. We could have also applied improvements made to person detection to other categories, like the two-wheels. Furthermore, the use of single-category learning strategy opened the doors for other using other datasets that we could had explored, like pedestrian surveillance related ones. These points should be considered in future works.

Future application of this model includes parking-lot surveillance and traffic intersection surveillance, as well as client-specific demands like motorway surveillance. In addition, the use of synthetic data should be further explored, and the use of the techniques in other object detection algorithms is encouraged.

As a future engineer, obtaining useful information from huge quantities of data was a challenge, and a skill I am grateful to have improved during this internship. Alongside it, I improved my skills in image processing, deep learning, algorithm, and software designs skills which will be invaluable for my future career.

These works were based on many others, and we hope they can be used to create better projects.

BIBLIOGRAPHY

1 WANG, X.; CHEN, J.; WANG, Z.; LIU, W.; SATOH, S.; LIANG, C.; LIN, C.-W. When pedestrian detection meets nighttime surveillance: A new benchmark. In: BESSIERE, C. (Ed.). *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization, 2020. p. 509–515. Main track. Disponível em: <https://doi.org/10.24963/ijcai.2020/71>.

2 LIU, L.; OUYANG, W.; WANG, X.; FIEGUTH, P.; CHEN, J.; LIU, X.; PIETIKÄINEN, M. Deep learning for generic object detection: A survey. *International journal of computer vision*, Springer, v. 128, n. 2, p. 261–318, 2020.

3 BOCHKOVSKIY, A.; WANG, C.; LIAO, H. M. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020. Disponível em: <https://arxiv.org/abs/2004.10934>.

4 YU, F.; WANG, D.; SHELHAMER, E.; DARRELL, T. Deep layer aggregation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2018.

5 HIRUNYAWANAKUL, A.; KERDPRASOP, N.; KERDPRASOP, K. A novel heuristic method for misclassification cost tuning in imbalanced data. *International Journal of Machine Learning and Computing*, v. 8, p. 565–570, 12 2018.

6 Padilla, R.; Netto, S. L.; da Silva, E. A. B. A survey on performance metrics for object-detection algorithms. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. [S.l.: s.n.], 2020. p. 237–242.

7 DU, D.; ZHU, P.; WEN, L.; BIAN, X.; LIN, H.; HU, Q.; PENG, T.; ZHENG, J.; WANG, X.; ZHANG, Y.; BO, L.; SHI, H.; ZHU, R.; KUMAR, A.; LI, A.; ZINOLLAYEV, A.; ASKERGALIYEV, A.; SCHUMANN, A.; MAO, B.; LEE, B.; LIU, C.; CHEN, C.; PAN, C.; HUO, C.; YU, D.; CONG, D.; ZENG, D.; PAILLA, D. R.; LI, D.; WANG, D.; CHO, D.; ZHANG, D.; BAI, F.; JOSE, G.; GAO, G.; LIU, G.; XIONG, H.; QI, H.; WANG, H.; QIU, H.; LI, H.; LU, H.; KIM, I.; KIM, J.; SHEN, J.; LEE, J.; GE, J.; XU, J.; ZHOU, J.; MEIER, J.; CHOI, J. W.; HU, J.; ZHANG, J.; HUANG, J.; HUANG, K.; WANG, K.; SOMMER, L.; JIN, L.; ZHANG, L. Visdrone-det2019: The vision meets drone object detection in image challenge results. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. [S.l.: s.n.], 2019.

8 PAILLA, D. R.; KOLLERATHU, V. A.; CHENNAMSETTY, S. S. Object detection on aerial imagery using centernet. *CoRR*, abs/1908.08244, 2019. Disponível em: <http://arxiv.org/abs/1908.08244>.

9 MANDAL, V.; ADU-GYAMFI, Y. Object detection and tracking algorithms for vehicle counting: a comparative analysis. *Journal of Big Data Analytics in Transportation*, Springer, v. 2, n. 3, p. 251–261, 2020.

10 GHIASI, G.; CUI, Y.; SRINIVAS, A.; QIAN, R.; LIN, T.-Y.; CUBUK, E. D.; LE, Q. V.; ZOPH, B. Simple copy-paste is a strong data augmentation method for instance segmentation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2021. p. 2918–2928.

11 de Almeida, P. R.; OLIVEIRA, L. S.; BRITTO, A. S.; SILVA, E. J.; KOERICH, A. L. Pklot – a robust dataset for parking lot classification. *Expert Systems with Applications*, v. 42, n. 11, p. 4937–4949, 2015. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417415001086>.

12 AMATO, G.; CARRARA, F.; FALCHI, F.; GENNARO, C.; MEGHINI, C.; VAIRO, C. Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications*, v. 72, p. 327–334, 2017. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S095741741630598X>.

13 WU, Y.; TAN, H.; QIN, L.; RAN, B.; JIANG, Z. A hybrid deep learning based traffic flow prediction method and its understanding. *Transportation Research Part C: Emerging Technologies*, v. 90, p. 166–180, 2018. ISSN 0968-090X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0968090X18302651>.

14 POLSON, N. G.; SOKOLOV, V. O. Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, v. 79, p. 1–17, 2017. ISSN 0968-090X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0968090X17300633>.

15 ZHANG, S.; BENENSON, R.; SCHIELE, B. Citypersons: A diverse dataset for pedestrian detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2017.

16 LOY, C. C.; LIN, D.; OUYANG, W.; XIONG, Y.; YANG, S.; HUANG, Q.; ZHOU, D.; XIA, W.; LI, Q.; LUO, P. et al. Wider face and pedestrian challenge 2018: Methods and results. *arXiv preprint arXiv:1902.06854*, 2019.

17 WEN, L.; DU, D.; CAI, Z.; LEI, Z.; CHANG, M.; QI, H.; LIM, J.; YANG, M.; LYU, S. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, 2020.

18 GEIGER, A.; LENZ, P.; URTASUN, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2012.

19 DOLLÁR, P.; WOJEK, C.; SCHIELE, B.; PERONA, P. Pedestrian detection: An evaluation of the state of the art. *PAMI*, v. 34, 2012.

20 YU, F.; XIAN, W.; CHEN, Y.; LIU, F.; LIAO, M.; MADHAVAN, V.; DARRELL, T. BDD100K: A diverse driving video database with scalable annotation tooling. *CoRR*, abs/1805.04687, 2018. Disponível em: <http://arxiv.org/abs/1805.04687>.

21 LIN, T.-Y.; DOLLÁR, P.; GIRSHICK, R.; HE, K.; HARIHARAN, B.; BELONGIE, S. Feature pyramid networks for object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 2117–2125.

22 GHIASI, G.; LIN, T.-Y.; LE, Q. V. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2019.

23 JIANG, H.; LEARNED-MILLER, E. Face detection with the faster r-cnn. In: IEEE. *2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017)*. [S.l.], 2017. p. 650–657.

24 HE, K.; GKIOXARI, G.; DOLLÁR, P.; GIRSHICK, R. Mask r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2961–2969.

25 TAN, M.; PANG, R.; LE, Q. V. Efficientdet: Scalable and efficient object detection. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. [S.l.: s.n.], 2020. p. 10781–10790.

26 LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S. E.; FU, C.; BERG, A. C. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. Disponível em: <http://arxiv.org/abs/1512.02325>.

27 DUAN, K.; BAI, S.; XIE, L.; QI, H.; HUANG, Q.; TIAN, Q. Centernet: Keypoint triplets for object detection. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 6569–6578.

28 LIN, T.-Y.; GOYAL, P.; GIRSHICK, R.; HE, K.; DOLLÁR, P. Focal loss for dense object detection. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2980–2988.

29 NEWELL, A.; YANG, K.; DENG, J. Stacked hourglass networks for human pose estimation. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 483–499.

30 CO, G. *Classification: Precision and recall | machine learning crash course*. Google, 2020. Disponível em: <https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall>.

31 EVERINGHAM, M.; ESLAMI, S. M. A.; GOOL, L. V.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, v. 111, n. 1, p. 98–136, jan. 2015.

32 LIN, T.; MAIRE, M.; BELONGIE, S. J.; BOURDEV, L. D.; GIRSHICK, R. B.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. Disponível em: <http://arxiv.org/abs/1405.0312>.

33 CAI, J.; LEE, F.; YANG, S.; LIN, C.; CHEN, H.; KOTANI, K.; CHEN, Q. Pedestrian as points: An improved anchor-free method for center-based pedestrian detection. *IEEE Access*, v. 8, p. 179666–179677, 2020.

34 BAHNSEN, C. H.; MOESLUND, T. B. Rain removal in traffic surveillance: Does it matter? *IEEE Transactions on Intelligent Transportation Systems*, IEEE, p. 1–18, 2018.

35 KISANTAL, M.; WOJNA, Z.; MURAWSKI, J.; NARUNIEC, J.; CHO, K. Augmentation for small object detection. *CoRR*, abs/1902.07296, 2019. Disponível em: <http://arxiv.org/abs/1902.07296>.

36 GAIDON, A.; WANG, Q.; CABON, Y.; VIG, E. Virtual worlds as proxy for multi-object tracking analysis. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2016. p. 4340–4349.

37 CABON, Y.; MURRAY, N.; HUMENBERGER, M. *Virtual KITTI 2*. 2020.

38 TREMBLAY, J.; PRAKASH, A.; ACUNA, D.; BROPHY, M.; JAMPANI, V.; ANIL, C.; TO, T.; CAMERACCI, E.; BOOCHOON, S.; BIRCHFIELD, S. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. [S.l.: s.n.], 2018.

39 SHRIVASTAVA, A.; PFISTER, T.; TUZEL, O.; SUSSKIND, J.; WANG, W.; WEBB, R. Learning from simulated and unsupervised images through adversarial training. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2017.

40 HATTORI, H.; LEE, N.; BODDETI, V. N.; BEAINY, F.; KITANI, K. M.; KANADE, T. Synthesizing a scene-specific pedestrian detector and pose estimator for static video surveillance. *International Journal of Computer Vision*, Springer, v. 126, n. 9, p. 1027–1044, 2018.

41 BAHNSEN, C.; VÁZQUEZ, D.; M. López, A.; MOESLUND, T. Learning to remove rain in traffic surveillance by using synthetic data. In: KERREN, A.; HURTER, C.; BRAZ, J. (Ed.). *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. [S.l.]: SCITEPRESS Digital Library, 2019. v. 4, p. 123–130. 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (Visigrapp 2019) ; Conference date: 25-02-2019 Through 27-02-2019.

42 DOSOVITSKIY, A.; ROS, G.; CODEVILLA, F.; LOPEZ, A.; KOLTUN, V. Carla: An open urban driving simulator. In: PMLR. *Conference on robot learning*. [S.l.], 2017. p. 1–16.

43 DWIBEDI, D.; MISRA, I.; HEBERT, M. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2017.

44 ZHUANG, F.; QI, Z.; DUAN, K.; XI, D.; ZHU, Y.; ZHU, H.; XIONG, H.; HE, Q. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, v. 109, n. 1, p. 43–76, 2021.