MICROMOBILITY SAFETY APPLICATIONS USING AI

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Jaume Prats Cristià

In partial fulfilment

of the requirements for the degree in

TELECOMMUNICATION TECHNOLOGIES AND SERVICES ENGINEERING

Advisors:

Josep Ramon Morros Rubió, Elisa Sayrol Clols

Barcelona, June 2021

Abstract

In recent years, population overcrowding in large cities has generated serious problems for urban transport, encouraging the use of new means of transportation such as micro-mobility vehicles. This thesis explores the possibilities of using computer vision techniques for the detection of pavement damages in bike lanes, thus reinforcing the safety of this type of vehicle. A damage detector is developed using a convolutional neural network. The project also provides a database of damages present in the bike lanes of the city of Barcelona.

Resum

En els darrers anys, la massificació de la població a les grans ciutats ha generat greus problemes per al transport urbà, encoratjant l'ús de nous mitjans de transport com és el cas dels vehicles de micromobilitat. Aquesta tesi explora les possibilitats de les tècniques de visió per ordinador per la detecció de danys i desperfectes als carrils bici, reforçant d'aquesta manera la seguretat per aquest tipus de vehicles. Es desenvolupa un detector de danys utilitzant una xarxa neuronal convolucional. El projecte també aporta una base de dades de danys presents en els carrils bici de la ciutat de Barcelona.

Resumen

En los últimos años, la masificación de la población en las grandes ciudades ha generado graves problemas para el transporte urbano, fomentando el uso de nuevos medios de transporte como es el caso de los vehículos de micromobilidad. Esta tesis explora las posibilidades del uso de técnicas de visión por ordenador para la detección de daños en los carriles bici, reforzando de esta forma la seguridad de este tipo de vehículos. Se desarrolla un detector de daños usando una red neuronal convolucional. El proyecto también aporta una base de datos de daños presentes en los carriles bici de la ciudad de Barcelona.

Acknowledgments

I would like to express my deep gratitude to Professor Josep Ramon Morros Rubió and Professor Elisa Sayrol Clols, my project supervisors, for their patient guidance, enthusiastic encouragement, and useful critiques of this thesis work. I would also like to especially thank Miquel Torrecilla Mercado, my project partner with whom I have managed to carry out this project, helping each other in the challenges that have arisen.

Finally, I wish to thank my family and friends for their support and encouragement throughout this stage.

Revision history and approval record

Revision	Date	Purpose
0	27/05/2021	Document creation
1	19/06/2021	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Jaume Prats Cristià	jaume.prats.cristia@estudiantat.upc.edu
Josep Ramon Morros Rubió	ramon.morros@upc.edu
Elisa Sayrol Clols	elisa.sayrol@upc.edu

Written by:		Reviewed and approved by:	
Date	19/06/2021	Date	21/06/2021
Name	Jaume Prats Cristià	Name	Josep Ramon Morros Rubió Elisa Sayrol Clols
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

1. Introduction

This thesis is an extension and continuation of a broad project proposed by professors Elisa Sayrol Clols and Josep Ramon Morros Rubió, intending to study and reinforce micro-mobility safety in cities.

Even though this project has been carried out in parallel by two students, Miquel Torrecilla Mercado and myself Jaume Prats Cristià, this thesis only describes my contribution to the project.

1.1. Statement of purpose

Cities globally are grappling with the negative externalities of car travel and are therefore striving to move towards a more sustainable urban transportation system. The introduction and popularity of new personal transport modes, such as e-scooters and electric bicycles, could potentially accelerate this transition.

Barcelona, as any large city, is adapting to this transformation by incorporating more and more bike lanes to the extensive network of more than 200km and with the filling of pioneering regulations for the regulation of micro-mobility such as this January 2021.

This thesis focuses on the use of Computer Vision techniques for the detection of damages in bike lanes that may pose a threat to micro-mobility users. Information that could be used to warn the user or the city council to repair them.

1.2. Requirements and specifications

An important part of this project consists of the creation of a dataset of damages in the bike lanes of Barcelona. The goal is to capture and annotate about 300-400 images.

Subsequently, this dataset will be used to train object detection models to be able to detect and classify the different types of road damages in the city.

As the main objective of this project is to work as a proof of concept and evaluate the possibilities of Deep Learning techniques applied to detect bike lane damages, no further quantitative technical specifications were established to assess the requirements.

1.3. Methods and procedures

The database used in this thesis consists of two datasets, the first one from the Global Road Damage Detection Challenge 2020 [1], and the second one entirely crated in this project. As for the algorithm, a state-of-the-art object detection model has been used, the Faster R-CNN [2].

1.4. Work Plan

The project breakdown structure consists of 7 work packages.

Autonomous Learning		WP1
Description:	Learn about foundational topics in the field of Machine Learning and its evolution to Deep Learning for object detection.	
Dates:	Start: 02/02/2021 End: 02/03/2021	
Milestones:	-	

Table 1.1 - Work package 1

State of Art research		WP2
Description:	Research on state of art road damage detection techniques.	
Dates:	Start: 09/02/2021 End: 02/03/2021	
Milestones:	-	

Table 1.2 - Work package 2

Documentation		WP3
Description:	Write documentation on the work done throughout the project.	
Dates:	Start: 23/02/2021 End: 21/06/2021	
Milestones:	Project Proposal (08/03/2021) Critical Review (14/04/2021) Final Report (21/06/2021)	

Table 1.3 - Work package 3

Dataset		WP4
Description:	Capture and label a collection of images of road defects and search on the internet for useful datasets.	
Dates:	Start: 16/02/2021 End: 08/06/2021	
Milestones:	Dataset (08/06/2021)	

Table 1.4 - Work package 4

Model		WP5
Description:	Search the internet for suitable models to use for our project. Compare the different models found and decide which one is best for our application. Finally, understand deeply the model chosen and learn how to train it and adapt it to our solution.	
Dates:	Start: 02/03/2021 End: 27/04/2021	
Milestones:	-	

Table 1.5 - Work package 5

Training		WP6
Description:	Train the model with different combinations of the resulting datasets from WP3.	
Dates:	Start: 02/02/2021 End: 02/03/2021	
Milestones:	-	

Table 1.6 - Work package 6

Test & Evaluate		WP7
Description:	Evaluate and test the algorithm to maximize its performance. Make possible adjustments and experiments to obtain final results.	
Dates:	Start: 27/04/2021 End: 15/06/2021	
Milestones:	-	

Table 1.7 - Work package 7

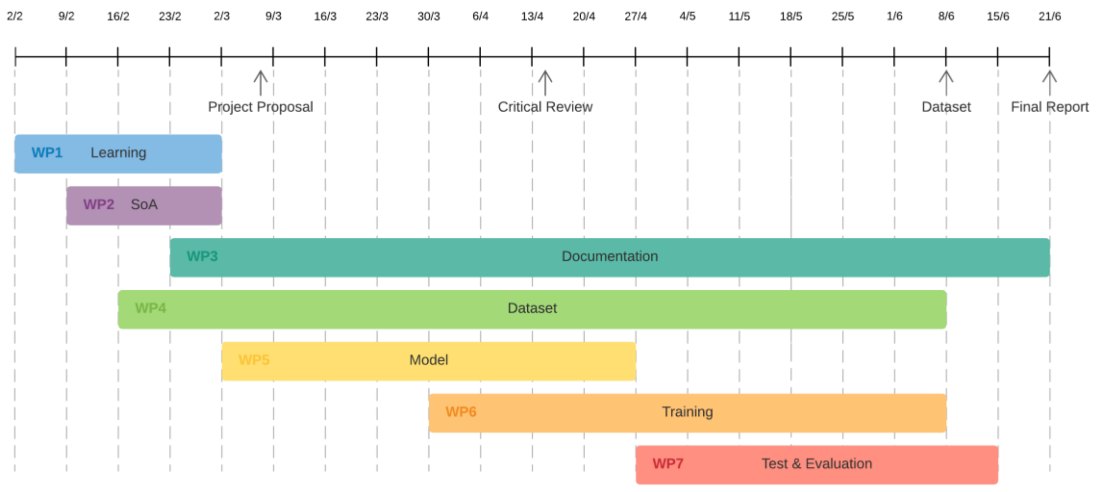


Figure 1.1 - Gantt Diagram

1.5. Deviations

Initially, the creation of the dataset was proposed as a first step prior to the training and testing of the algorithm, but as a public dataset was found that met our initial needs the capture and labelling became a background work package.

Overall no further substantial deviations from the proposed timeline have appeared.

2. State of the art

As stated in the introduction, the project has been developed in parallel by two students, Miquel Torrecilla Mercado and myself. For this reason, this section has been written jointly and is shared in both theses.

2.1. Road Damage Detection

Road maintenance requires a periodic evaluation of their condition. This maintenance is carried out by different state agencies using road inspection vehicles with different sensors installed. They usually make use of three-dimensional (3D) cameras and cameras with lasers that capture the pavement surface with the highest possible resolution [3] (see Figure 2.1).

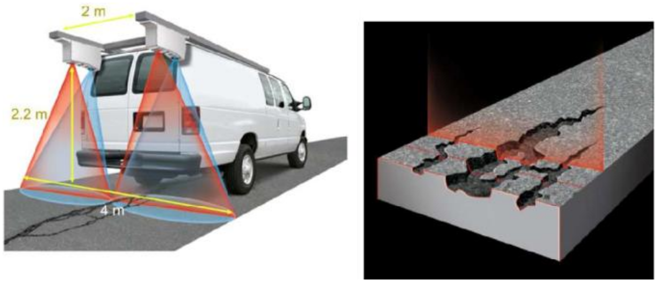


Figure 2.1 - Illustration of the ARAN 9000 [3]

However, these dedicated vehicles are usually very expensive and often unaffordable for local agencies with more limited budgets. In addition, most likely if they are vehicle-based they will not adapt to micro-mobility lanes. This leads to the requirement of low-cost methods capable of comprehensively surveying road surfaces, such as methods that can be implemented using Smartphones and object detection with Deep Learning techniques.

2.2. Object Detection

Computer vision is an interdisciplinary field that has been gaining huge amounts of traction in recent years and an integral part of it is object detection. Object detection is the task of detecting instances of objects of a certain class within an image, a challenge that in turn is made of two separate problems, knowing where the object is located within the image, and knowing what class of object it is.

According to how they approach this task, the state-of-the-art methods can be categorized into two main types: one-stage detectors and two-stage detectors.

Two-stage detectors use a Region Proposal Network (RPN) to generate a series of regions of interest (ROIs) and these are fed to a second module that classifies the image within each proposed region. **One-stage detectors** on the other hand try to solve both tasks in only one pass.

In general, two-stage detectors prioritize detection accuracy and one-stage detectors focus more on inference speed for real-time use.

2.2.1. Faster R-CNN [2]

Faster R-CNN is the third iteration of the R-CNN family, one of the most representative networks in two-stage detectors. R stands for regions and CNN stands for convolutional neural networks. The main objective of Faster R-CNN is to reduce computational expenses and to be able to work in real-time.

In Faster R-CNN the image is provided as an input to a convolutional neural network which provides a convolutional feature map. Then, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped and fed to a second module that classifies the image on each proposed region (see figure 2.2).

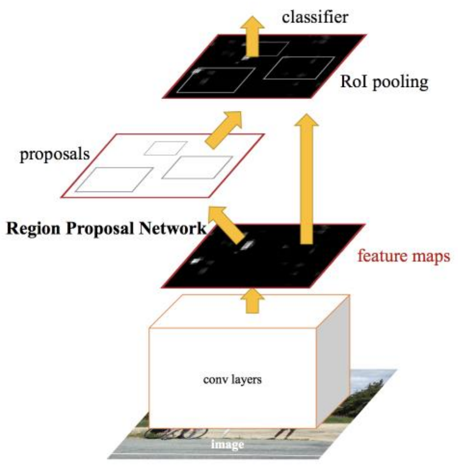


Figure 2.2 - Faster R-CNN architecture

2.2.2. Cascade R-CNN [4]

Cascade R-CNN is an object detection architecture proposed by the University of California San Diego in 2017 to address problems with degrading performance with increased IoU [1] thresholds.

Usually, an object detector trained with a low IoU threshold produces noisy detections. However, if we try to increase the threshold to combat the noise, detection performance tends to degrade. This degradation is due to two main factors, overfitting during training by eliminating positive samples, and inference-time mismatch between IoUs for which detector is optimal and the inputs.

Cascade R-CNN tries to solve this problem with a multistage extension of the R-CNN, where detector stages deeper in the architecture are sequentially more selective against false positives. The Cascade R-CNN stages are trained sequentially, using the output of one stage to train the next.

When operating in this manner, the Cascade R-CNN consists of a sequence of detectors adapted to increasingly higher IoUs to beat the overfitting problem and be effectively trained.

[1] Intersection over Union (IoU) is known to be a good metric for measuring the overlap between the predicted and the actual bounding box of an object, it is just de division of the overlap by the union. In object detection, an IoU threshold is used to define positives and negatives.

2.2.3. YOLO [5]

All of the previous object detection algorithms use regions to locate the object within the image. YOLO or You Only Look Once is an object detection algorithm much different that uses a single convolutional network to predict the bounding boxes and the class probabilities for these boxes. It works by taking an image and dividing it into grid cells, for every cell, several bounding boxes are predicted. Then, the algorithm adjusts and outputs a class probability for every bounding box (see figure 2.3).

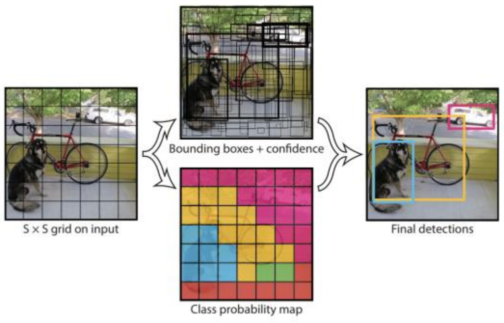


Figure 2.3 - YOLO diagram

YOLO was first proposed in 2016 by Joseph Redmon and it was a milestone in object detection research due to its capability of detecting objects in real-time with better accuracy.

Subsequently, Redmon evolved its algorithm with two new versions YOLOv2 and YOLOv3. However, in 2020, as YOLO became famous, Joseph Redmond announced that he stopped his research in computer vision due to several concerns regarding the potential negative impact of his work.

Despite Redmon's withdrawal, it was not the end of YOLO, later in 2020 two new versions of YOLO have been released in parallel, YOLOv4 [6] by Alexey Bochkovskiy in Darknet, the original YOLO framework, and YOLOv5 [7] by Glenn Jocher a PyTorch implementation.

2.3. Global Road Damage Detection Challenge

The Global Road Damage Detection Challenge (GRDDC) [1] was created with the intention to improve state-of-the-art road damage detection using computer vision techniques.

It was initiated by The University of Tokyo in 2018 in the IEEE Big Data Cup [8]. The first challenge had the intention to evaluate the contemporary detection methods. This lead to the creation of the actual challenge in the IEEE International Conference on Big Data'2018 in Seattle, USA. It was a success with the participation of 54 teams with several new methods and it became an annual event.

After the first challenge, some Japanese cities started to use the proposed methods, providing the first feedback of the solutions which was not so good at the time.

The two premises that make up the challenge are:

- A dataset of road damages and,
- an online competition.

The dataset consists of about 26.000 labelled images from Japan, India and the Czech Republic. The two main challenges are the detection and the subsequent classification of the damages.

To evaluate the models proposed by the participants two criteria are set to define a hit or a miss detection:

- The intersecting area divided by the union between the predicted and ground truth bounding box must be over 50% (IoU > 0.5).
- The predicted category must match the ground truth category.

With the described criteria the models are ranked by the resulting F-score in two test sets.

Table 2.1 shows the best teams, models and scores from the 2020 edition:

Rank	Team	Test1 F-score	Test2 F-score	Proposed solution
1	IMSC (USA - Jordan)	0.6748	0.6662	Ensemble Learning with Ultralytics-YOLO and Test Time Augmentation
2	SIS Lab (USA)	0.6275	0.6358	Ensemble model with YOLOv4 as base model.
3	DD-Vision (China)	0.629	0.6219	A Consistency Filtering Mechanism and model ensemble with Cascade R-CNN as the base model
4	Titan_mu (USA)	0.5814	0.5751	YOLO model trained on CSPDarknet53 backbone
5	Dongjuns (Denmark)	0.5683	0.5710	YOLOv5x
6	SUTPC (China)	0.5636	0.5707	Ensemble (YOLOv4 + Faster-RCNN)
7	RICS (USA)	0.565	0.547	EfficientDet
8	AIRS-CSR (China)	0.554	0.541	YOLOv4
9	CS17 (Japan)	0.5413	0.5430	Resnet-18 and Resnet-50 backbones based Faster-RCNN two-stage detection architecture
10	BDASL (USA)	0.5368	0.5426	Multi-stage Faster R-CNN with Resnet-50 and Resnet-101 backbones

Table 2.1 - GRDDC2020 Ranking

3. Methodology

3.1. Database

The database of this project consists of two datasets. The first one is a public dataset from the Global Road Damage Detection Challenge 2020 [1], and the second one has been entirely created in this thesis.

3.1.1. RDD2020 Dataset

The Road Damage Detection 2020 Dataset [9] was the dataset used for the Global Road Damage Detection Challenge (GRDDC), a Big Data Cup organized as a part of the IEEE International Conference on Big Data'2020.

The RDD2020 dataset contains 26.336 road images collected from India, Japan, and the Czech Republic with more than 31.000 instances of road damages. It is divided into a training set of 21.041 images, and two test sets, test1 and test2 of 2.631 and 2.664 images, respectively (see figure 3.1).

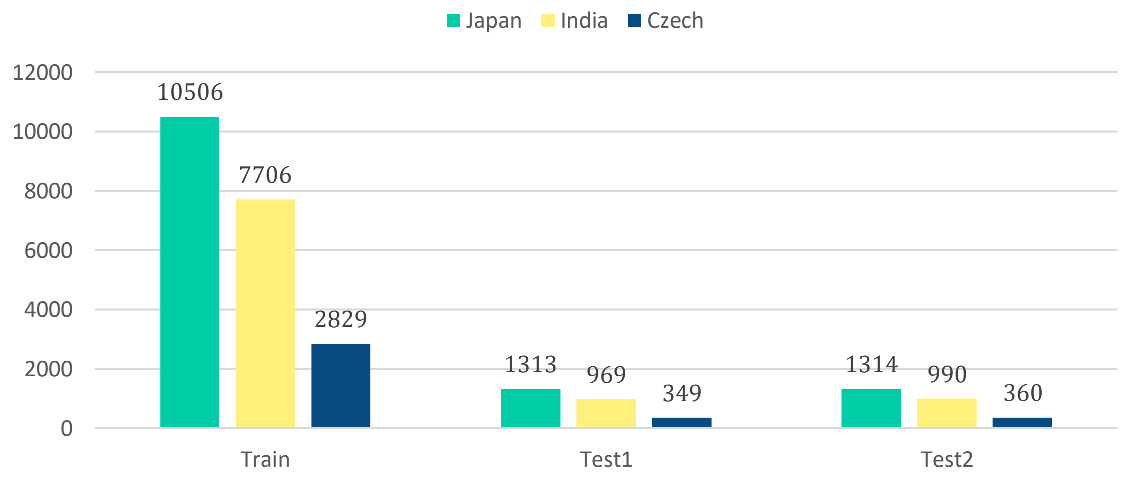


Figure 3.1 - RDD2020 Statistics

The images were captured using vehicle-mounted smartphones, to make it useful for municipalities and road agencies to develop methods for low-cost monitoring of road pavement surface conditions. The instances of road damages are delimited with bounding boxes. These annotations are in XML files in PASCAL VOC [10] format.

The dataset contains annotations for the following damage categories:

- D00 - Longitudinal Crack
- D10 - Transverse Crack
- D20 - Alligator Crack
- D40 - Pothole

Japan images also contain three additional categories:

- D43 - Cross Walk Blur
- D44 - White Line Blur
- D50 - Manhole

In the figures below we can see an example of every category.

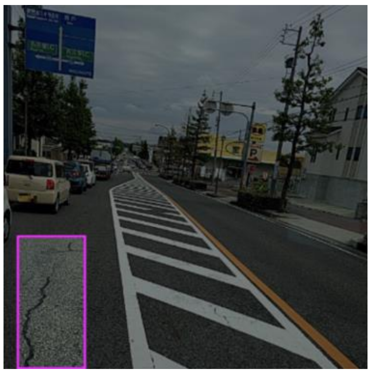


Figure 3.7 - RDD2020 - D00

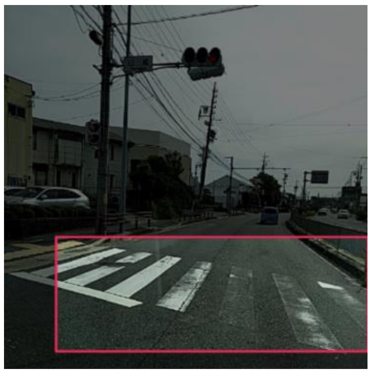


Figure 3.5 - RDD2020 - D43

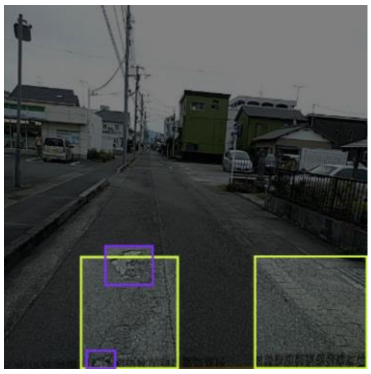


Figure 3.6 - RDD2020 - D20 - D40

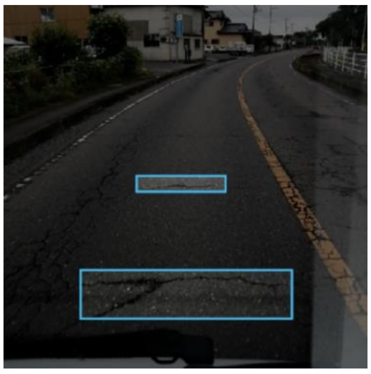


Figure 3.4 - RDD2020 - D10

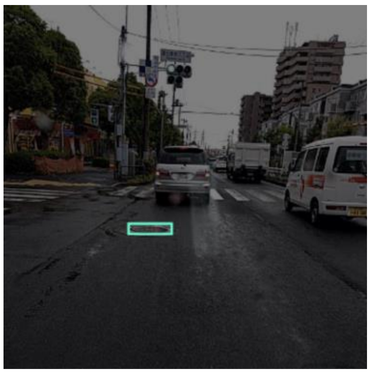


Figure 3.3 - RDD2020 - D50

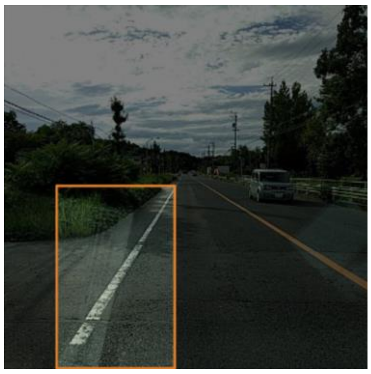


Figure 3.2 - RDD2020 - D44

The distribution of the different categories in the train set can be seen in figure 3.8.

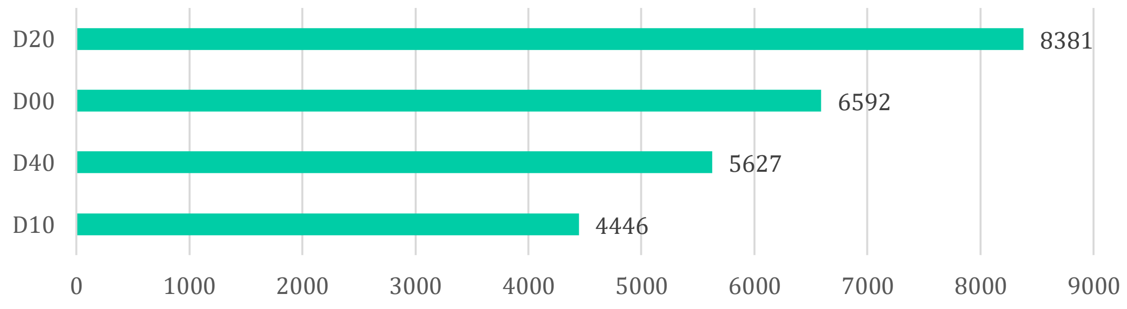


Figure 3.8 - RDD2020 train instances distribution

3.1.2. BLDD Dataset

The Bike Lane Damage Detection Dataset is a dataset created during the four months of this project. It contains 370 images of bike lanes from the Barcelona Metropolitan Area with 677 instances of damages. The images have been captured using a smartphone camera simulating the perspective of a camera placed on a personal mobility vehicle or bicycle and annotated using an online annotation tool for object detection called MakeSense.AI [10].

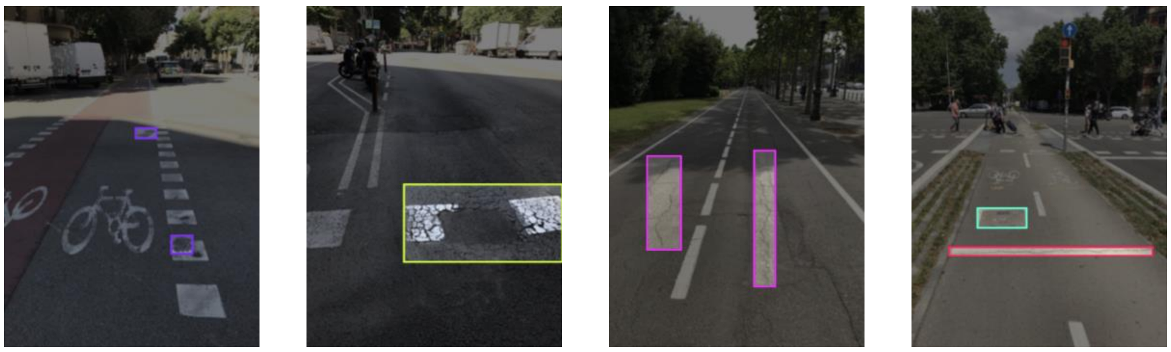


Figure 3.9 - BLDD - from left to right, D40, D20, D00, D10 and D50

The dataset contains annotations in PASCAL VOC [10] format of the five following classes:

- D00 - Longitudinal Crack
- D10 - Transverse Crack
- D20 - Alligator Crack
- D40 - Pothole
- D50 - Manhole

The main problem encountered when creating the dataset was the surprisingly small number of defects found in Barcelona's bike lane network. The city's cycle lanes are relatively new and even with more than 50km cycled it was difficult to find potholes (D40) or alligator cracking (D20), two defects that are usually present on roads but are very rare on bike lanes due to their lower degradation.

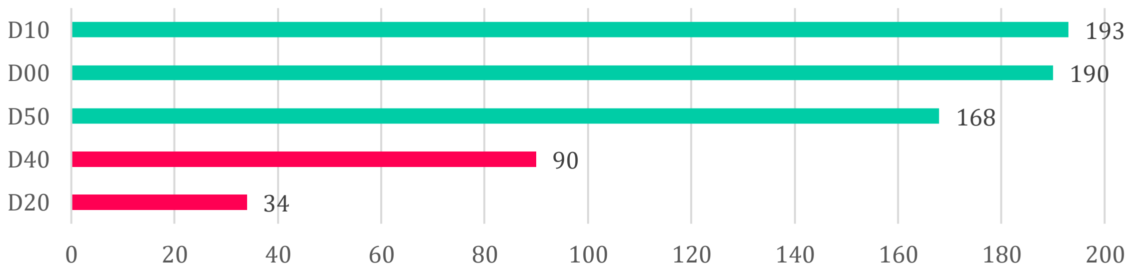


Figure 3.10 - BLDD instances distribution

As can be seen in Figure 3.10 the dataset is not balanced with two under-represented defect categories due to the challenge mentioned above. This will lead to different experiments to try to solve this problem. The BLDD dataset has been transferred to the university to be used in future research. The annotations can also be found in the public repository of the project: https://github.com/JaumePrats/BikeLaneDamageDetection.

3.2. Damage Detection

3.2.1. Model

Object detection is a growing area of computer vision and there are currently several state-of-the-art models to choose from for a project like this one.

The first selection was mainly based on the models used by the best teams in the Global Road Damage Detection 2020. As can be seen in section 2.3, most of the participants that ended up in the first positions used Faster R-CNN, Cascade R-CNN, YOLO, or a combination of those.

YOLO as explained in section 2.2.3 is a single-shot or one-stage detector which is great especially for achieving low inference times but can struggle when the training data is limited compared to the multi-stage ones. On the other hand, Faster R-CNN and Cascade R-CNN have lower inference speed but as they are multi-stage detectors will usually perform better with limited training data, which is the case of our dataset.

After this initial evaluation, the models were initially trained with a small part of the RDD2020 Dataset and the model that performed better was the Faster R-CNN followed by the Cascade R-CNN. It is for this reason that the network chosen to be used in this thesis is the Faster RCNN.

The models used in this thesis have been previously trained by my project partner Miquel Torrecilla Mercado. In his thesis, he has trained and compared different object detectors (Faster R-CNN and Cascade R-CNN) with the RDD2020 dataset. The resulting weights from the Faster R-CNN are used as a basis to train the models in this thesis.

3.2.2. Training

The experiments and training of the models in this thesis are carried out using MMDetection [12], an open-source object detection toolbox based on PyTorch [13]. MMDetection features a model zoo that contains many state-of-the-art detectors and a modular design that meets the needs of the project.

In MMDetection a model is defined by a configuration file and its parameters saved in a checkpoint file.

A configuration file contains 4 basic components:

- **Model:** Describes de detector model.
- **Schedule:** Defines the train, validation, and test pipelines.
- **Dataset:** Sets the dataset for training, validation, and test.
- **Runtime:** Defines other parameters such as the learning rate, the number of epochs, or the validation interval.

In this project, several configuration files have been created to conduct the experiments described in the results. All are based on the *faster_rcnn_r50_caffe_fpn_mstrain_1x_coco*, which uses a Faster R-CNN detector with an R50-FPN backbone.

All the configuration files used in this thesis can be found in the public repository: https://github.com/JaumePrats/BikeLaneDamageDetection.

As previously mentioned, the detectors used in this thesis have been previously trained with the RDD2020 Dataset, which is why the checkpoint file used in this thesis is the result of the training performed by my project partner Miquel Torrecilla Mercado.

3.2.3. Pre-processing

Although most of the images from the BLDD Dataset have been captured using the same smartphone there are some images captured with other devices with different aspect ratios.

Furthermore, in some experiments conducted the dataset has been combined with the dataset from Global Road Damage Detection Challenge, raising the need for pre-processing the images before training.

To adapt all images to the same aspect ratio, all images have been resized to 600x600 adding black edges to fill the ratio (figure 3.11). In this way, the image size of the captured images is also reduced for smaller file sizes and faster training.

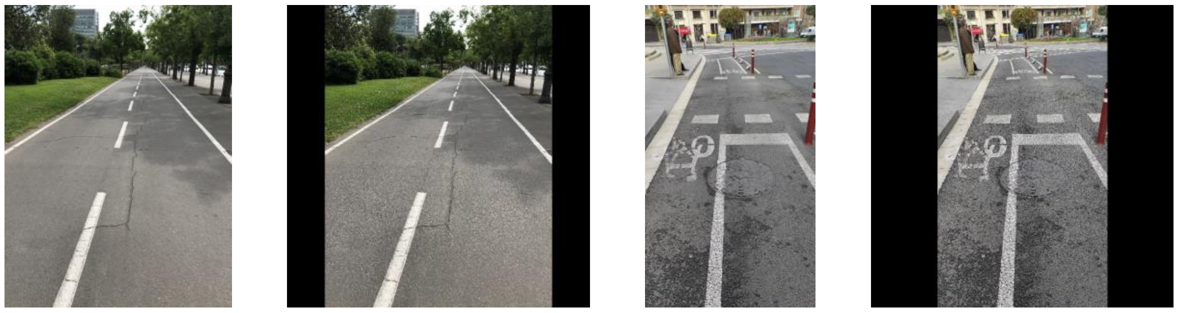


Figure 3.11 - Resize example

In addition, the dataset annotations have been also converted from PASCAL VOC [10] to COCO [14] format to simplify the implementation in MMDetection.

All the pre-processing mentioned has been conducted through Roboflow [15], an online platform for computer vision that allows the management of datasets. Working in this way, every time images were added to the dataset a new version was created and could be exported to the training environment.

3.2.4. Data Augmentation

Data Augmentation is a set of techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data.

Data augmentation is commonly used in deep learning for two main reasons. The first one is to counter the lack of training data, which is exactly the case of this thesis, the second one is that it avoids memorization preventing overfitting in training, creating a model that performs better in unseen samples.

According to where the Data augmentation is conducted there are two methods:

The first one is known as **offline data augmentation** which consists of performing all the necessary transformations beforehand, essentially increasing the size of your dataset by a factor equal to the number of transformations performed.

The second one is known as **online data augmentatio**n or **augmentation on the fly**. In this method, the transformations are performed in mini-batches before feeding them to the model. Although this is usually preferred for larger datasets to avoid the increase in the dataset size, this is the method used in this thesis.

An important aspect of data augmentation is the randomness of the augmentations, every transformation is applied with a probability, in this way every batch fed to the detector has substantially different transforms than the last one, preventing even more memorization by the model.

To perform data augmentation MMDetection features some basic transformations that can be applied directly to the training pipeline but to explore more complex ones the Albumentations [15] library has been used. Albumentations is a Python library for image augmentation that can be used through the data pipelines in MMDetection. The augmentation pipeline used in the MMDetection configuration file can be seen in the **Annex**.

Albumentations offers an extensive list of pixel-level and spatial-level transformations to apply to images, to select the chosen ones, research was done to determine their effects on training and whether they were applicable or made sense in our case.

The data augmentation applied is the following:

Horizontal Flip

Flipping the images is useful to prevent that the model recognizes the objects in a particular orientation and also change the location of the objects in the image. In our case, just a horizontal flip is applied with a probability of 0.5 because a vertical flip does not make sense since the algorithm is not expected to see upside-down images for this application.

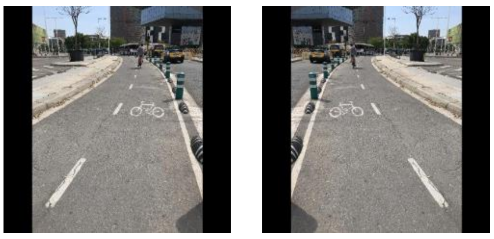


Figure 3.12 - Horizontal Flip

Blur and Noise

Many types of imperfections can make their way into an image: blur, noise, poor contrast, JPEG compression, and more. Among these, blur and noise have the most detrimental effect on neural networks [16]. In our application blur makes a lot of sense because the developed system is intended to be used with a moving camera placed on a micro-mobility vehicle. For this reason, the Albumentation transforms applied are GaussNoise to apply gaussian noise, and MovingBlur, which mimics the result of a moving image.

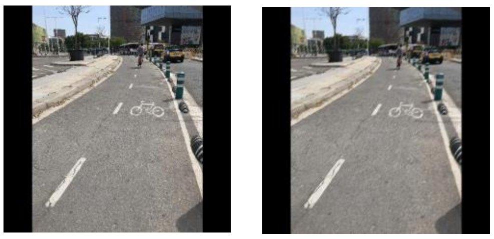


Figure 3.13 - Motion Blur

Rotation

As stated above the intended application of the developed algorithm is to be placed in a moving vehicle, in this case, the images captured may be tilted. For this reason, rotation between -15 to 15 degrees is applied to able the algorithm recognizes the damages in these circumstances. It is important to note that too much rotation may be detrimental because the model could confuse longitudinal (D00) and transversal (D10) cracks.

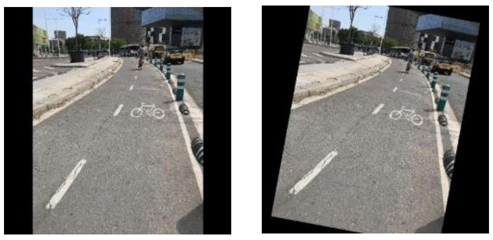


Figure 3.14 - Rotation

Brightness

To simulate different lighting conditions that the model may face in production use, a brightness augmentation has also been applied.

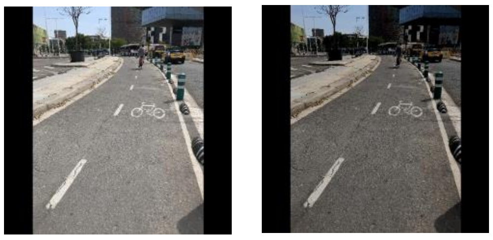


Figure 3.15 - Brightness

CutOut

In our application, some objects may be partially covered by other vehicles o elements in the bike line. To force the model to learn about the different parts of the objects detected a CutOut has been also implemented. Using the Cutout transform from Albumentations 8 25x25 squares are distributed randomly within the image.

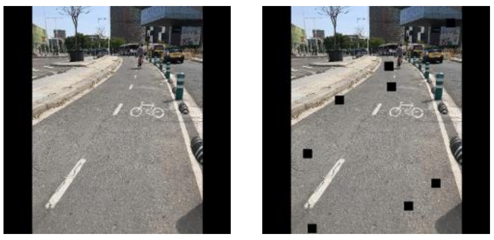


Figure 3.16 - Cutout

Color transformations

Our dataset is mostly captured using the same device, to adapt the model to be able to work with images from other devices some color transformations have been also added. RGBShift to slightly shift values for each channel of the image and HueSaturationValue to change the hue, saturation, and value.

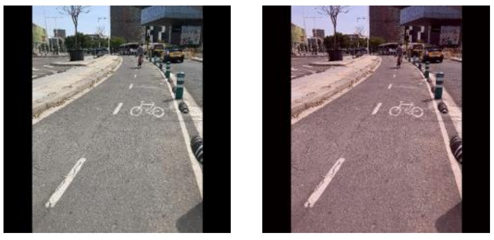


Figure 3.17 - RGBShift / HueSaturationValue

4. Results

4.1. Experimentation

A total of 5 experiments have been conducted to evaluate the model and the techniques applied during the training. To perform these experiments the BLDD Dataset has been divided between train (70%), validation (20%) and test (10%) sets. The resulting images for the train, validation and test set are 259, 74 and 37 respectively.

E1 - BLDD Dataset (bldd)

The first consists of training the detector with the dataset created in this thesis, BLDD Dataset, without any data augmentation techniques.

It is important to note that the BLDD Dataset has two under-represented classes, resulting in an unbalanced training set (see figure 4.1).

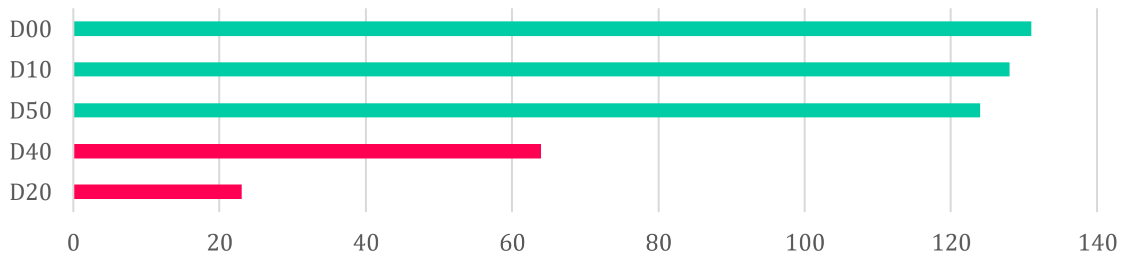


Figure 4.1 - bldd train instances distribution

The results from this experiment are used to compare the effects of the techniques applied to the others.

E2 - BLDD Dataset Balanced (bldd_bal)

In the second experiment, the unbalanced problem has been addressed by adding images from the RDD2020 dataset to the training set (see figure 4.2). The images added are exclusively from the Czech Republic and Japan, as the Indian ones depict rural areas and are visually distinct from the ones from our dataset. A total of 153 images containing the two under-represented classes were added to achieve the balance of the training set. The resulting training set is composed of 412 images with 909 instances.

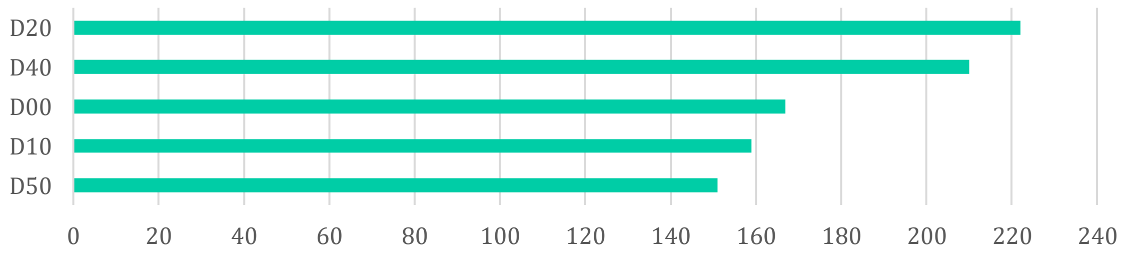


Figure 4.2 – bldd_bal train instances distribution

E3 - BLDD - RDD2020 (bldd_bal100)

This experiment intends to evaluate the effect that incrementing the dataset size has on the performance of the detector. Taking the balanced training set from the last experiment as a basis, an additional 100 images from the RDD2020 dataset have been added. Resulting in a training set of 512 images and 1119 instances. The instances distribution in the train set can be seen in figure 4.3.

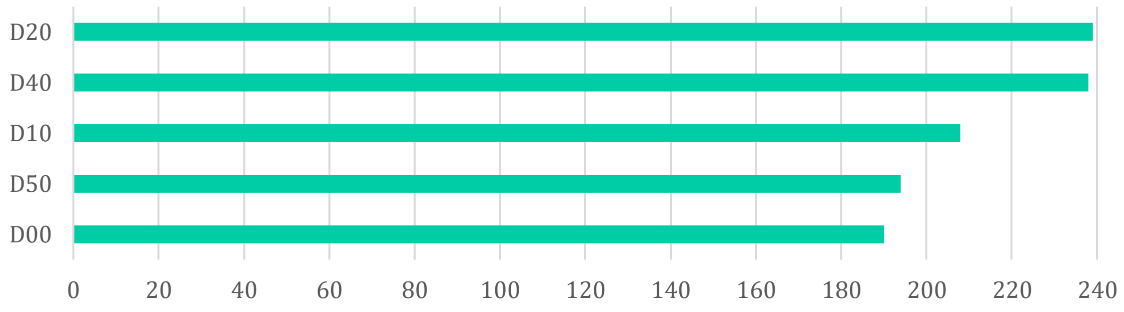


Figure 4.3 - bldd_bal100 train instances distribution

E4 - DA (bldd_da_pipeline)

Unlike the experiments described above, in the following experiments, no modifications have been made to the initial dataset. These focus on the use of augmentation techniques during training.

In this fourth experiment, the following transforms have been added to the augmentation pipeline:

- Horizontal flip.
- Motion blur.
- Gaussian noise.
- Rotation (between -15 and 15 degrees).
- Brightness modification.
- Cutout (8 squares of 25x25 pixels).
- Color transform (RGBShift and HueSaturationValue)

The training pipeline used in the configuration file is shown in the **Annex**.

E4 - DA - Flip – Blur (bldd_da_flip bldd_da_blur)

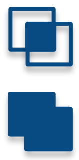
Finally, to further evaluate flip and blur augmentations, two more models have been trained. The first one, using only a horizontal flip transform in the augmentation pipeline. The second one using only a blur augmentation, the MotionBlur transform from Albumentation to be more precise.

4.2. Metrics

To analyze and compare the resulting detectors from every experiment the following metrics have been used.

To determine the similarity between the predicted bounding boxes and the ones from the ground truth, we have measured the overlap between these two using the Intersection over Union (IoU).

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$



In this way, we can define what constitutes a right prediction. If the predicted category matches the real one and the IoU is over 50%, we consider the prediction to be correct.

To measure how accurate the predictions of the detectors are, we have used two well-known metrics, precision and recall. Basically, precision measures the percentage of correct positive predictions among all predictions made, and recall measures the percentage of correct positive predictions among all positive cases in reality.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$TP: True\ Positive \quad TN: True\ Negative \quad FP: False\ Positive \quad FN: False\ Negative$$

There is a tradeoff between these two measures, combining them we obtain the F-score:

$$Fscore = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Lastly, an also common metric is used, Average Precision (AP). AP is defined as the area under the precision-recall curve. By averaging the AP for different classes, we get the Mean Average Precision (mAP) that has been used in validation.

4.3. Result analysis

To analyze the results, we will use the metrics described to compare the performance of the detectors trained in each experiment.

4.3.1. Dataset balancing

First of all, we evaluate the effects of balancing the training set by adding images from the RDD2020.

- **bldd:** training set from our dataset (Unbalanced).
- **bldd_bal:** training set + 153 RDD2020 images (Balanced).
- **bldd_bal100:** trained set + 253 RDD2020 images (Balanced).

We start by comparing the training loss. In figure 4.4 we can notice that while in the first model the training loss goes flat around epoch 7, as we add more images to the training set the loss takes longer to stabilize.

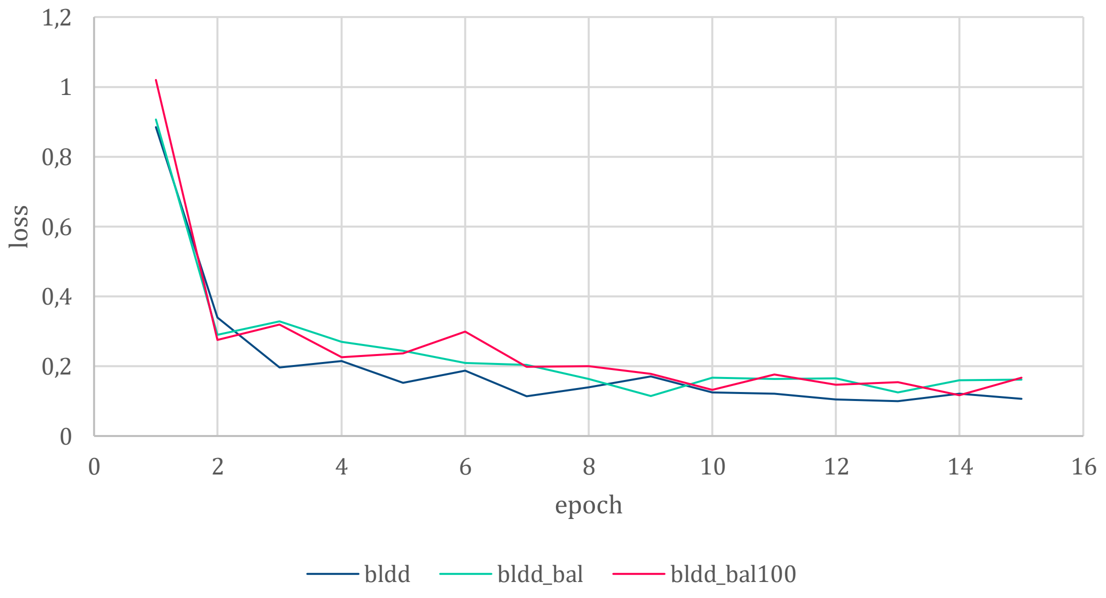


Figure 4.4 - Dataset balancing - train loss

We continue by examining the validation. In figure 4.5 we can see how the mAP decreases as we add RDD2020 images to the training set. This contradicts what we might think beforehand, the more images the better the results. In this example, we see that this is not the case. Although the added images provide more defect samples to the training set, they are visually distinct from the ones in our dataset, resulting in a degradation of the mAP. However, the results are not clear, because in the second experiment, where we add 100 additional images from Japan, a slight increase in mAP can be observed.

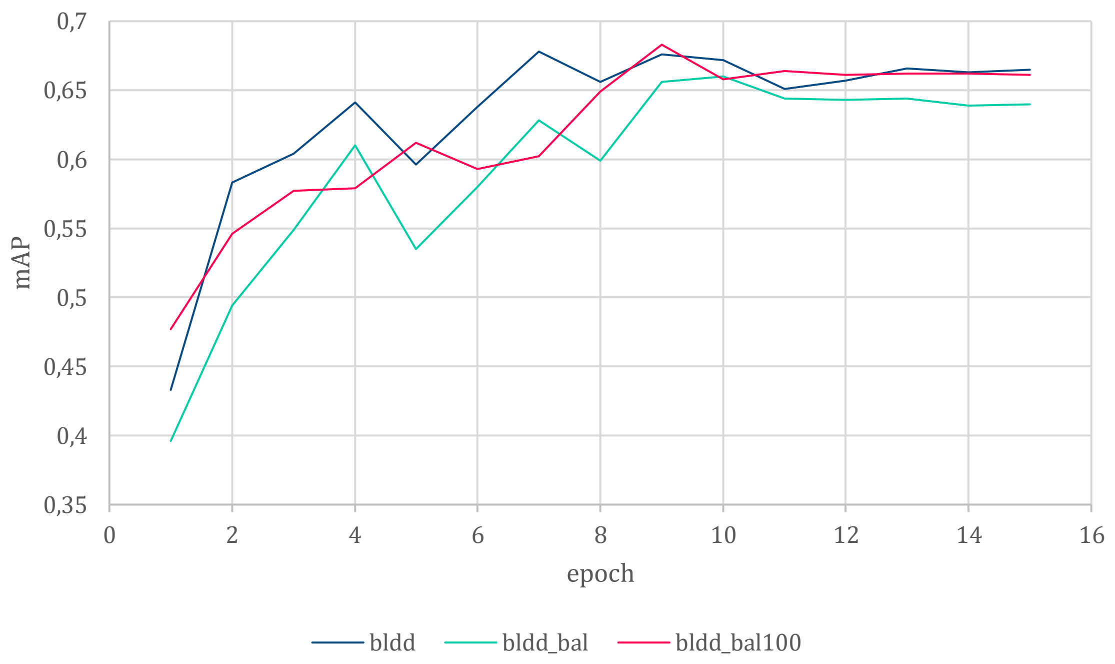


Figure 4.5 - Dataset balancing - validation mAP

Looking at the measures of precision and recall in the test set (table 4.1) we can further analyze what is going on.

		bldd	bldd_bal	bldd_bal100
D00	**Precision**	0,478	∧ 0,474	∧ 0,500
	Recall	0,367	∨ 0,321	∨ 0,345
	F-score	0,415	∨ 0,383	∨ 0,408
D10	**Precision**	0,684	∨ 0,571	∨ 0,619
	Recall	0,481	∨ 0,462	0,481
	F-score	0,565	∨ 0,511	∨ 0,542
D20	**Precision**	0,750	0,750	∨ 0,500
	Recall	0,143	0,143	∨ 0,100
	F-score	0,240	∨ 0,240	∨ 0,167
D40	**Precision**	0,400	∧ 0,600	0,400
	Recall	1,000	1,000	1,000
	F-score	0,571	∧ 0,750	0,571
D50	**Precision**	0,625	∧ 0,818	∧ 0,818
	Recall	0,667	∨ 0,643	∨ 0,643
	F-score	0,645	∧ 0,720	∧ 0,720

Table 4.1 - Dataset balancing - test

At first glance, we can see that adding images does not improve, or even worsen, the metrics of the first three categories. These correspond to longitudinal crack (D00), transversal crack (D10), and alligator cracking (D20). A possible explanation could be that cracks found in bike lanes tend to be narrower and less deep than the ones present in the roads, due to the less deterioration and difference in the used pavement. This could be making the detector less able to generalize to detect the bike lane ones from the test set.

As for the D40 (pothole) category, we can observe that in the three cases recall equals 1, which means that the detector identifies every pothole sample in the test set correctly, however, it also mislabels other parts of the images as potholes, obtaining a much lower precision value.

Finally, in the D50 category (manhole) we can see an increase in the F-score, given by a substantial increase in precision (0,8), indicating that the resulting detectors have identified correctly most of the instances. This is due to the following, in contrast to other categories, manhole samples from the RDD2020 dataset are almost identical to the ones from our dataset, which means that the samples added to the training set are consistent and produce this rise in precision.

4.3.2. Data Augmentation

In this section, we compare the results obtained in the conducted experimentation on data augmentation.

Four different models were trained to test different data augmentation strategies:

- **bldd:** Base model trained without any data augmentation technique as a reference.
- **bldd_da_pipeline:** Model trained with a data augmentation pipeline with a variety of transformations: horizontal flip, blur, noise, rotation, brightness modification, cutout, and color transformation.
- **bldd_da_flip:** Model trained using only horizontal flip in the augmentation pipeline.
- **bldd_da_blur:** Model trained using the blur transform as the only data augmentation strategy.

In contrast with the last set of experiments, in this case, the training loss plotted in figure 4.6 does not present any significant changes between the models.

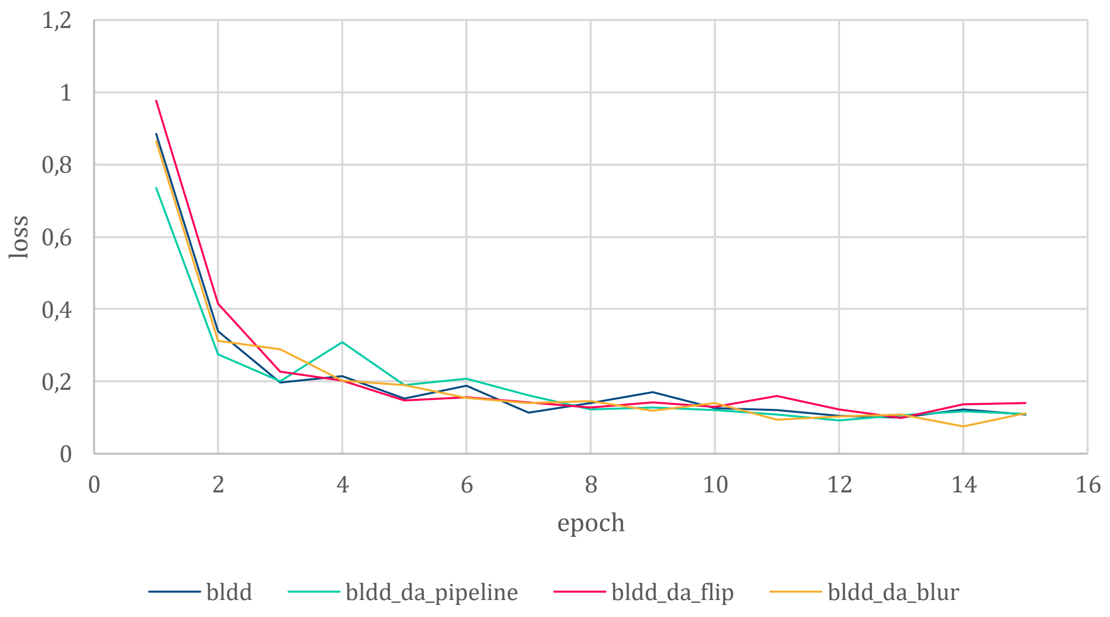


Figure 4.6 - Data Augmentation - train loss

In figure 4.7 we compare the obtained mAP during validation. We can notice that the model trained with all the transforms achieves the best mAP in validation, 0,69 the higher for all the trained models.

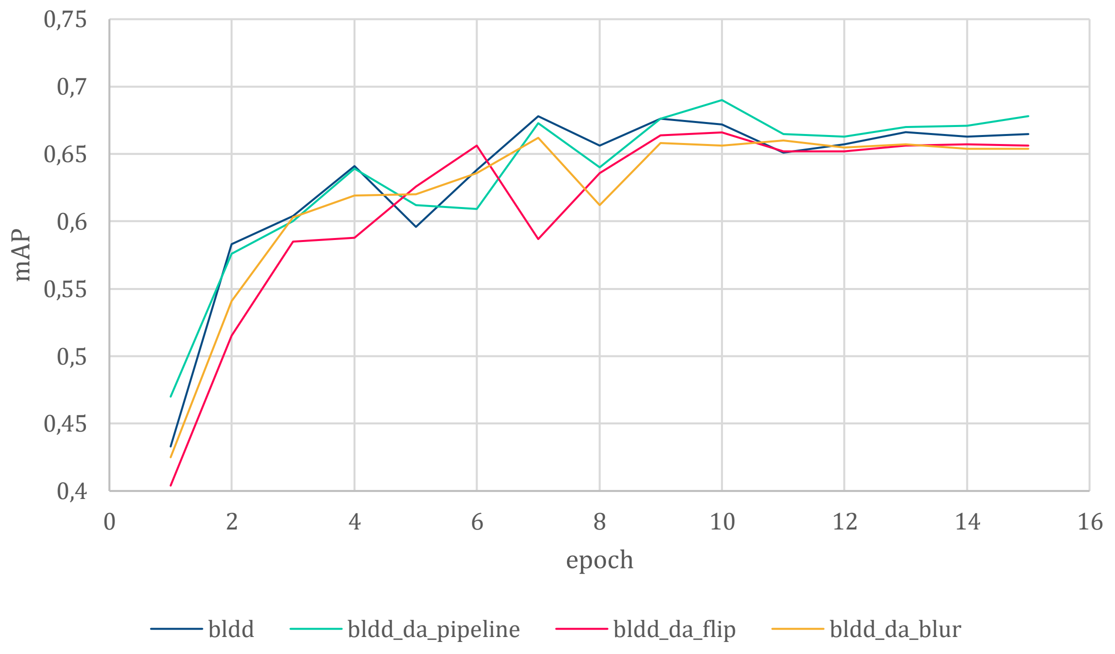


Figure 4.7 - Data Augmentation - validation mAP

To further analyze the differences between the models we compare the results obtained in the test set in table 4.2.

		bldd	bldd_da_pipeline	bldd_da_flip	bldd_da_blur
D00	**Precision**	0,478	∧ 0,500	∨ 0,450	∨ 0,400
	Recall	0,367	∨ 0,321	∨ 0,333	∨ 0,296
	F-score	0,415	∨ 0,391	∨ 0,383	∨ 0,340
D10	**Precision**	0,684	∨ 0,474	∨ 0,652	∨ 0,652
	Recall	0,481	∨ 0,391	∧ 0,517	∧ 0,517
	F-score	0,565	∨ 0,429	∧ 0,577	∧ 0,577
D20	**Precision**	0,750	∨ 0,600	0,750	∧ 1,000
	Recall	0,143	0,143	0,143	0,143
	F-score	0,240	∨ 0,231	∧ 0,241	∧ 0,250
D40	**Precision**	0,400	0,400	∧ 0,500	0,400
	Recall	1,000	1,000	1,000	1,000
	F-score	0,571	0,571	∧ 0,667	0,571
D50	**Precision**	0,625	∧ 0,692	∧ 0,750	∧ 0,818
	Recall	0,667	∨ 0,643	∨ 0,643	∨ 0,429
	F-score	0,645	∧ 0,667	∧ 0,692	∧ 0,720

Table 4.2 – Data Augmentation - test

On one hand, we can observe that for the first two data augmentation models the results do not show any substantial improvement for the crack classes (D00, D10, D20), even for the D00 we see a drop in precision and recall. On the other hand, a considerable rise in the precision of the D50 class (manhole) using flip and blur transforms.

In this case, in contrast to the results obtained in validation, the model trained with the more complex augmentation pipeline performed substantially worse than the others. Only showing a slight improvement for the D50 category.

A priori, we might think that augmentation techniques would lead to an improvement in detector performance. But clearly, the results do not match expectations. Apart from the D50 class, the performance has not been affected much. Even when adding more transformations in the augmentation pipeline the results have been worse.

It should be noted, however, that most of the images from the dataset, and therefore the ones from the test set, are captured using the same smartphone and perfectly static and level. This could be the reason why the data augmentation techniques explored have not produced a substantial improvement in the performance of the detectors. Nonetheless, these techniques could be beneficial in a real case scenario where the images are captured from a mounted camera in a micro-mobility vehicle.

In general, we can say that the experiments conducted have not led to a substantial improvement in the performance of the detectors. In the first case, we have seen how adding images from the RDD2020 dataset has had a negative effect on the ability to generalize in the test images, especially for the crack categories. In the second case, the results obtained using flip or blur augmentation showed a slight improvement for some classes, however, the model trained with different augmentation transforms combined has obtained the worst results in the test set.

Yet, there is one class that has behaved well in every experiment, D50 (manhole), for every method applied the precision in the test has been around or over 0,7, even 0,8 in some cases.

Finally, we must say that in this thesis, the dataset used is very small, we are talking about 259 train images, 74 for validation, and 37 in the test set. It is for this reason that the results obtained have some randomness preventing a conclusive analysis of the techniques applied and the subsequent draw of clear conclusions.

4.3.3. Test examples

Figure 4.5 shows some examples of the test set and the predicted results from the different detectors trained. In some cases, we can see how the detector placed more than one bounding box of the same category over the same damage. Even though this produces a false positive that lowers the precision, it is not particularly bad in our application since the algorithm still detects the presence of a defect.

In figure 4.6 we can see some examples of difficult images from the test set. These examples are intended to emphasize that cracks are objects that do not have defined edges. This makes it difficult to delimit where they begin and where they end, a fact that in some cases has made annotation difficult and therefore their detection by the models.

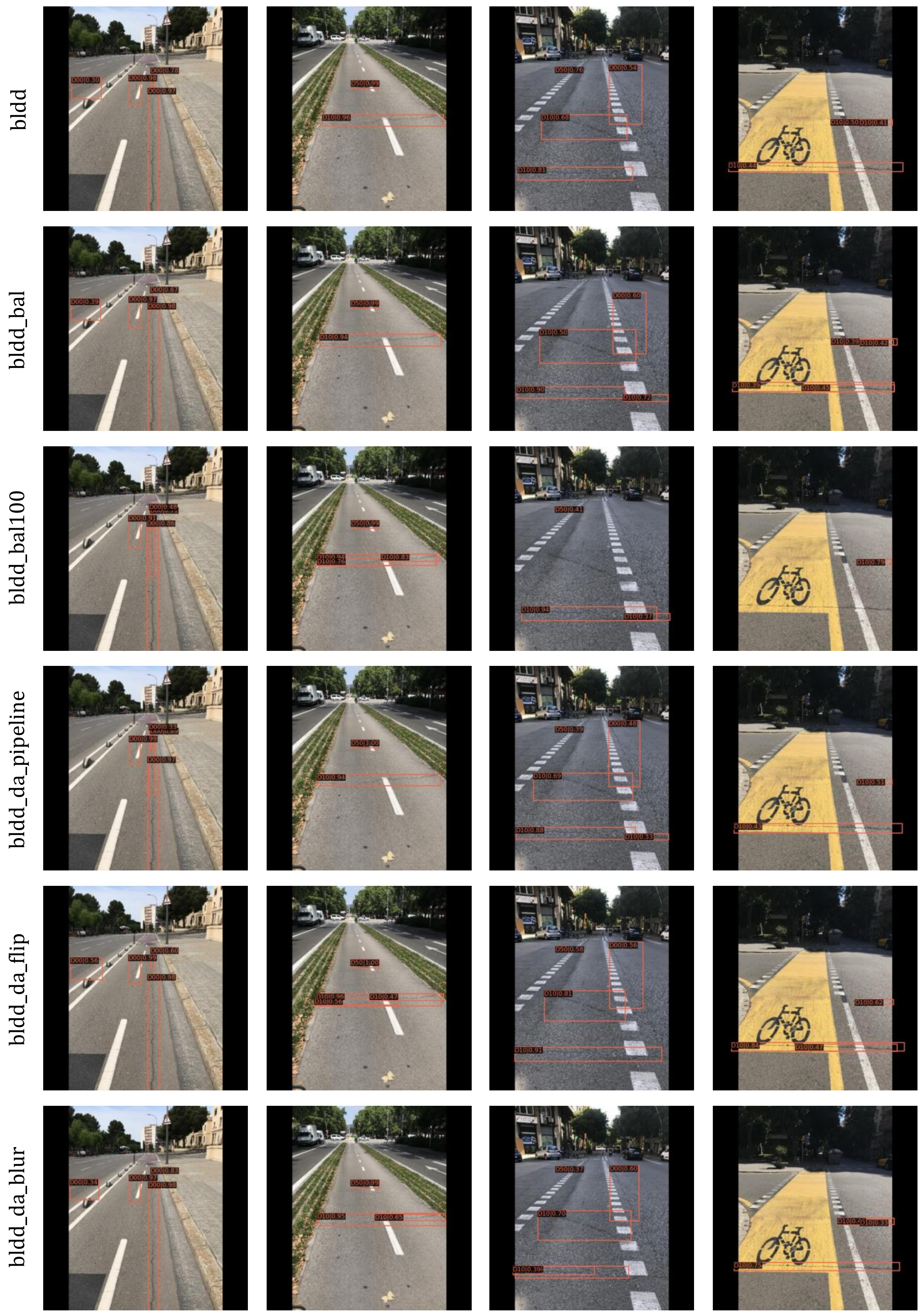


Figure 4.8 - Test images

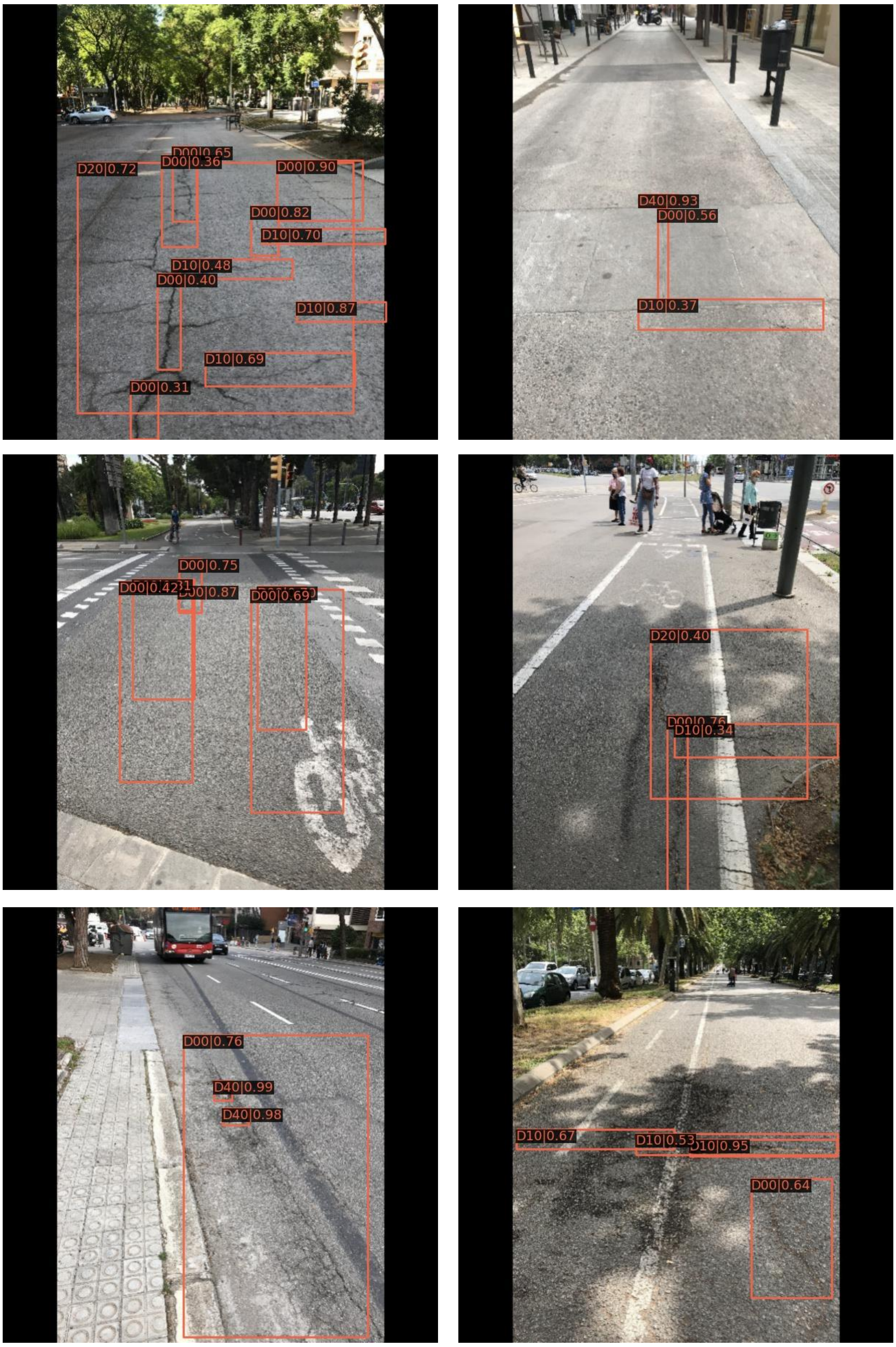


Figure 4.9 - Difficult test images

5. Budget

For the estimation of the budget of the project, a series of premises have been held:

- The project has lasted 20 weeks, including the learning phase. About 25 hours per week have been employed to its development.
- The worker's salary consists of 11 € per hour (an internship salary is considered). Consequently, the total salary rises to 5.500 €.
- The project has been supervised by two professors from ETSETB. A price of 70 € per hour is assumed and 1 hour per week for each supervisor. Resulting in a total of 2.800 €.
- For the development of the project, the free basic plan of Google Colaboratory has been used, as well as a remote server provided by UPC's Image Processing Group. However, to estimate the cost, we assume the use of the paying plan of Google Colaboratory, which is around 9 € per month. Resulting in a total of 36€.
- All licenses and software used are open source so no cost is added.

Table 5.1 shows the total cost of the project.

Concept	**Cost**
Salary	5.500,00 €
Superviesion	2.800,00 €
Development	36,00 €
Licenses and software	0,00 €
TOTAL	**8.336,00 €**

Table 5.1 - Budget

6. Conclusions and future development

The objective of this project was to develop a system to reinforce micro-mobility safety by detecting the damage on bike lanes. To achieve this goal, a dataset has been created and subsequently used to train an object detector, the Faster R-CNN. Afterward, a set of experiments has been conducted to explore different techniques used to improve the detection performance.

We have seen how, even using a rather small data set, we have been able to satisfactorily detect the defects present in different images, achieving more than decent results. This could serve as the first step to further develop a system to detect and identify numerous types of bike lane damages and use the information to warn users or maintenance services.

A relevant part of the project has consisted of the capture and posterior annotation of images to create the dataset. These resources can be used for future research to further develop the system and the field of damage detection in bike lanes.

Going into more detail about the experiments performed, we have seen how consistency between train and test is important for performance, not always more images mean better results. As for data augmentation, the strategies applied may be useful for a production application of the detector but didn't show a substantial performance improvement.

Still, the limited data has posed a challenge for the analysis of results. The small number of images in the validation and test set had led to inconclusive results about the effectiveness of the techniques applied.

For this reason, future work would be to use K-Fold cross validation during training, in this way, we could evaluate the techniques used in a more deterministic manner. Another improving area would be the use of class weights during training to tackle dataset imbalance.

As stated in the introduction, this thesis is to work as a proof of concept and evaluate the possibilities of Deep Learning techniques applied to bike lane damage detection. After the work performed, we can say without a doubt that the implementation of such a system could be used as part of a package to improve the safety of micro-mobility users in today's cities. For example, a collaborative application could be implemented that uses the images captured by different users to establish a classification of the state and deterioration of the different bike lanes in a city.

Bibliography

[1] "Global Road Damage Detection Challenge 2020," 2020. [Online]. Available: https://rdd2020.sekilab.global.

[2] S. Ren, K. He, R. Girshick and J. Sun, *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,* arXiv:1506.01497, 2015.

[3] F. Naveed, "Application Of Deep Learning In Identifying Road Cracks," October 2019. [Online]. Available: https://towardsdatascience.com/application-of-deep-learning-in-identifying-road-cracks-8153e50ce9e2.

[4] Z. Cai and N. Vasconcelos, *Cascade R-CNN: Delving into High Quality Object Detection,* arXiv:1712.00726, 2017.

[5] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection,* arXiv:1506.02640, 2016.

[6] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, *YOLOv4: Optimal Speed and Accuracy of Object Detection,* arXiv:2004.10934, 2020.

[7] G. Jocher, "ultralytics/yolov5," 2020. [Online]. Available: https://github.com/ultralytics/yolov5.

[8] "IEEE Big Data Cup," 2021. [Online]. Available: http://bigdataieee.org/BigData2021/CallBigDataCupChallenges.html.

[9] D. Arya, H. Maeda, S. K. Ghosh, D. Toshniwal, H. Omata, T. Kashiyama, T. Seto, A. Mraz and Y. Sekimoto, "RDD2020: An Image Dataset for Smartphone-based Road Damage Detection and Classification," 2021. [Online]. Available: https://data.mendeley.com/datasets/5ty2wb6gvg/1.

[10] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn and A. Zisserman, *The PASCAL Visual Object Classes (VOC) Challenge,* https://homepages.inf.ed.ac.uk/ckiw/postscript/ijcv_voc09.pdf.

[11] P. Skalski, "Make Sense," [Online]. Available: https://www.makesense.ai.

[12] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai and J. Wang, *MMDetection: Open MMLab Detection Toolbox and Benchmark,* arXiv:1906.07155, 2019.

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani and S. Chilamkurthy, *PyTorch: An Imperative Style, High-Performance Deep Learning Library,* arXiv:1912.01703, 2019.

[14] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, P. P. James Hays, D. Ramanan, C. L. Zitnick and P. Dollár, *Microsoft COCO: Common Objects in Context,* arXiv:1405.0312, 2014.

[15] B. Dwyer, J. Nelson, J. Solawetz, A. Morrow, K. Williams, T. Hansen, M. Traore, R. Huerta, N. Kelley and S. Sahoo, "Roboflow," [Online]. Available: https://roboflow.com.

[16] A. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov and A. A. Kalinin, *Albumentations: fast and flexible image augmentations,* arXiv:1809.06839, 2018.

[17] S. Dodge and L. Karam, *Understanding How Image Quality Affects Deep Neural Networks,* arXiv:1604.04004, 2016.

Annex:

Data augmentation pipeline used (entire configuration file in the project repository: https://github.com/JaumePrats/BikeLaneDamageDetection):

```
albu_train_transforms = [
    dict(
        type='HorizontalFlip',
        p=0.5),
    dict(
        type='OneOf',
        transforms=[
            dict(
                type='MotionBlur',
                blur_limit=(3,7),
                p=1),
            dict(
                type='GaussNoise',
                var_limit=(10.0, 60),
                p=1),
        ],
        p=0.2),
    dict(
        type='Rotate',
        limit=(-15, 15),
        interpolation=1,
        border_mode=0,
        value=(0, 0, 0),
        mask_value=None,
        p=0.3),
    dict(
        type='RandomBrightness',
        limit=[-0.2, 0.2],
        p=0.2),
    dict(
        type='Cutout',
        num_holes=8,
        max_h_size=25,
        max_w_size=25,
        p=0.2),
    dict(
        type='OneOf',
        transforms=[
            dict(
                type='RGBShift',
                r_shift_limit=10,
                g_shift_limit=10,
                b_shift_limit=10,
                p=1.0),
            dict(
                type='HueSaturationValue',
                hue_shift_limit=20,
                sat_shift_limit=30,
                val_shift_limit=20,
                p=1.0)
        ],
        p=0.1),
]

train_pipeline = [
    ...
    dict(
        type='Albu',
        transforms=albu_train_transforms,
        bbox_params=dict(
                type='BboxParams',
                format='coco',
                label_fields=['gt_labels'],
                min_visibility=0.0,
                filter_lost_elements=True),
            keymap={
                'img': 'image',
                'gt_bboxes': 'bboxes'
            },
            update_pad_shape=False,
            skip_img_without_anno=True),
        ...
]
```