UNIVERSITAT ROVIRA i VIRGILI

Internship Report 2021

Road Damage Detection

Thibault Roux

under the supervision of Dr. Hatem Rashwan & Saddam Abdulwahab

June 2021 - August 2021
Universitat Rovira i Virgili - Tarragona - Spain

Contents

1 Introduction

With the roads aging, a lot of deformations can appear and make them dangerous for vehicles and drivers. Thus there is a need to detect these damages on the roads in order to repair them. The best is to adopt method that detects these damages automatically, that is to say without an expert analyzing the road, because it's expensive and very time-consuming. Several works have been made to inspect the road conditions but not a lot have been focusing on both detection and classification of the damages. [1] Classification is very useful for road managers because in order to repair the roads, they have to understand clearly what the damages are to take effective actions. Thus, my work aims to train neural networks in order to detect and classify damages on the roads. To do this, I used several datasets available on this GitHub repository [2], and more precisely the datasets from 2018 and 2020.

2 Approach

In this section, I will explain in detail the approach I used on both datasets in order to get my results. The two datasets I used are available on the GitHub page and were publicly released to make a competition called Global Road Damage Detection challenge where everybody could compete and aim to get the best models. There is a competition with different datasets every year. I was asked to work on the 2018 and the 2020 Dataset.

2.1 Work on the 2018 Dataset

2.1.1 About the classes

The Road damages are split in 8 classes, separated according to their shape, orientation and position on the road. These classes are D00, D01, D10, D11, D20, D40, D43 and D44. The following table gives an definition of each of these classes.

Damage Type			Detail	Class Name
Crack	Linear Crack	Longitudinal	Wheel mark part	D00
			Construction joint part	D01
		Lateral	Equal interval	D10
			Construction joint part	D11
	Alligator Crack		Partial pavement, overall pavement	D20
Other Corruption			Rutting, bump, pothole, separation	D40
			White line blur	D43
			Cross walk blur	D44

Source: Road Maintenance and Repair Guidebook 2013 JRA (2013) in Japan.
Note from the dataset creators: In reality, rutting, bump, pothole, and separation are different types of road damage, but it was difficult to distinguish these four types using images. Therefore, they were classified as one class, viz., D40.

Figure 1: Road damage types used in the 2018 dataset (*taken from* [1]
)

2.1.2 About the dataset

The 2018 dataset is composed of 9053 600x600 images of roads taken in different cities and wards in Japan (Ichihara, Chiba, Sumida, Nagakute, Adachi, Muroran and Numazu). The following chart shows the number of instances of each class in the dataset.

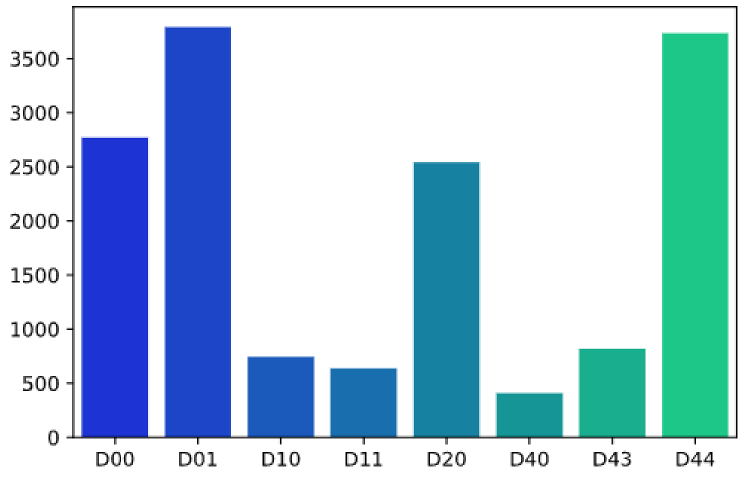


Figure 2: Number of instances of each class in the 2018 dataset

Finally, here are some labeled images belonging to the dataset.

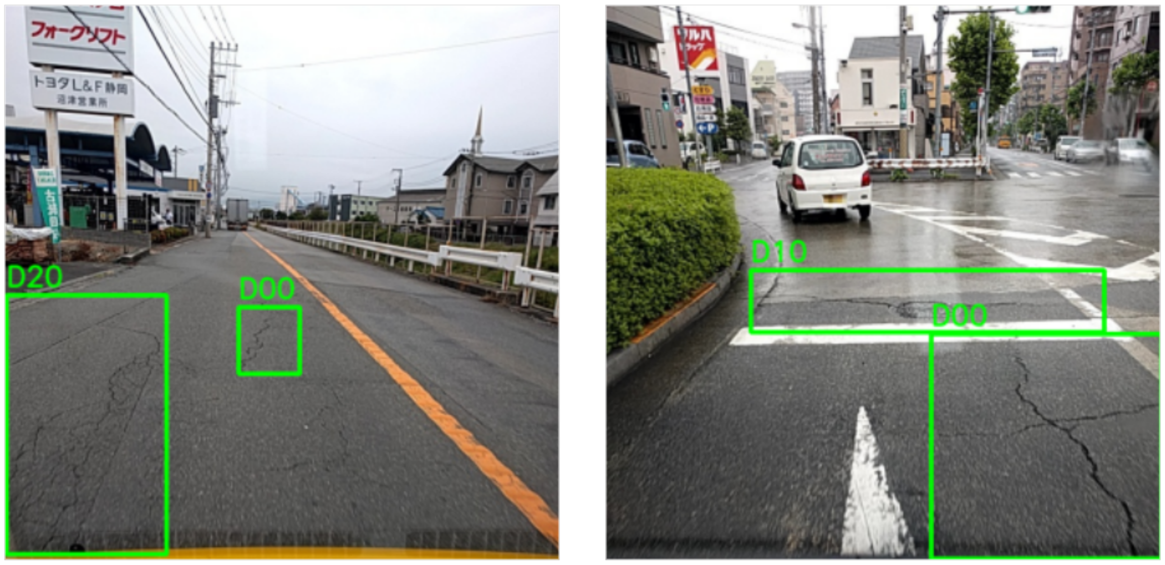


Figure 3: Images from the 2018 dataset and their labels

2.1.3 Train models using YoloV5

I was asked to use Yolo in order to train models on this dataset. I used the version 5 as it is the latest version at this date. [3] Every training I did is saved online thanks to a tool called WandB.

2.1.3.1 Using default hyperparameters

I started the training using default hyperparameters on the 4 different architectures of model available, whose information are detailed in the following figure.

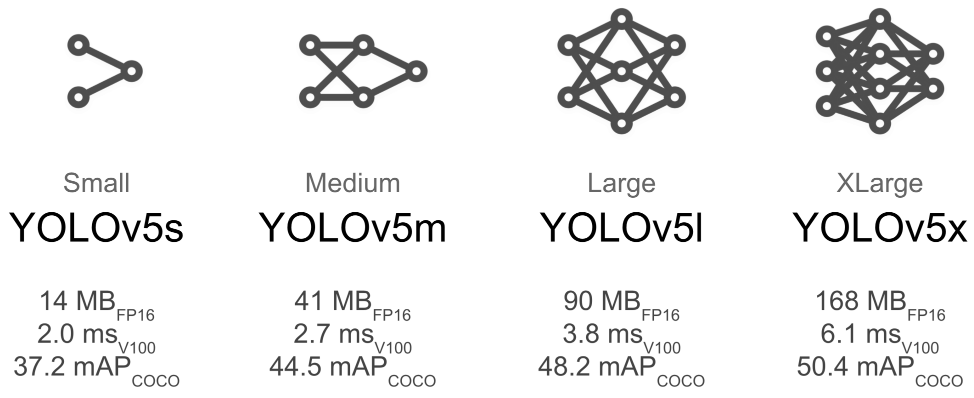


Figure 4: Comparison of the 4 models available in YoloV5

I tried to train these models with pretrained weights on the CoCo Dataset, as well as weights randomly initialised.

2.1.3.2 Modifying hyperparameters by hand

After getting the first results, I decided to modify some of the hyperparameters such as the number of epochs, the batch size, the learning rate, as well as hyperparameters having an influence on the augmentation of the images. (Image hue, value and saturation augmentation, scaling, translating)

2.1.3.3 Hyperparameter evolution

YoloV5 includes a feature called hyperparameter evolution, which is a method of hyperparameter optimisation using a genetic algorithm [4].

YoloV5 allows us to modify at will its 29 hyperparameters. For my part, I first tried to modify some hyperparameters by hand to see the influence of them in terms of performance. Then I used hyperparameter evolution, while freezing the less impactful hyperparameters to default, and making the others mutate.

The fitness function I used for this genetic algorithm is $0.9mAP@0.5 + 0.1mAP@0.5 : 0.95$ where mAP@0.5 is the mean average precision at 0.5 IoU (intersection over union) and mAP@0.5:0.95 is the average mAP over different IoU thresholds, from 0.5 to 0.95, with a step of 0.05.

I used this fitness function because the aim is not to find bounding boxes that fit exactly the damage, but more to detect the damages and make good classifications. Therefore we are more interested in mAP@0.5 than mAP@0.5:0.95. Moreover I didn't have time to test different fitness functions and compare the performances as this method is very time-consuming and I had power computation limits.

2.1.3.4 Other approaches

I also used some other approaches like test-time augmentation and model ensembling, that I'll be explaining in section 2.2

2.1.4 Conclusion on the 2018 Dataset

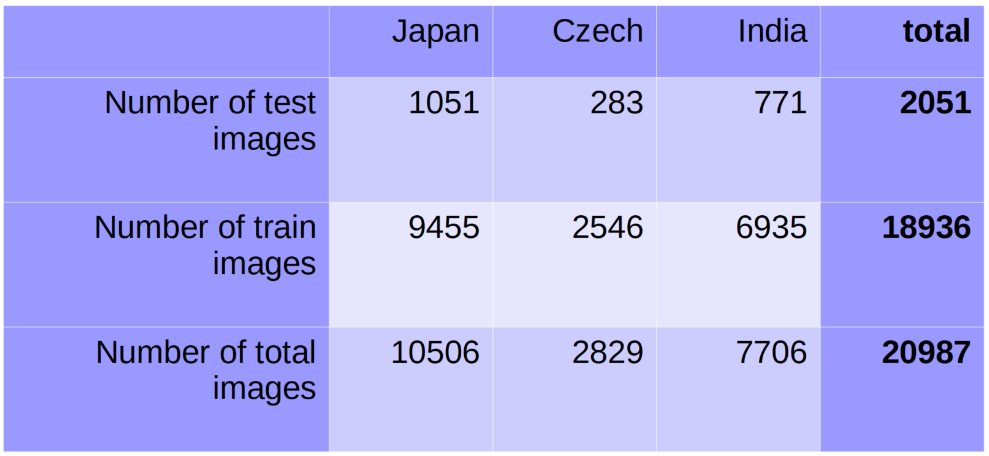
Although significant results have been achieved by some studies [1] , using this dataset to detect road damages all over the world is not a good idea. Indeed, the dataset only contains images of different cities of Japan and therefore can't be used to detect damages in every country because it is too specific to Japan roads which can be very different in other countries.[5] That is why GRDDC organizers decided to make a drastic change in the 2020 dataset, which we will be covering in the next section.

2.2 Work on the 2020 Dataset

This change is using images from different countries to see if the detection could still be possible with roads that can be very different in terms of shape, color, state, etc. Thus, the new dataset contains 20987 images, including 10506 images from Japan, but also 2829 images from Czech Republic and 7706 from India. Also the number of classes has been reduced to 4 (D00,D10,D20,D40).

2.2.1 About the dataset

I was asked to work on a new dataset, containing the previous images from Japan, as well as new images from Czech Republic and India, for a total of 20987 images and 25046 labels.



	Japan	Czech	India	**total**
Number of test images	1051	283	771	**2051**
Number of train images	9455	2546	6935	**18936**
Number of total images	10506	2829	7706	**20987**

Figure 5: Number of images according to the countries in the 2020 dataset

Moreover the number of classes was reduced to 4 as we can see on the following pie chart.

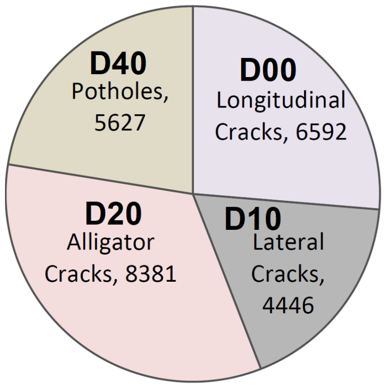


Figure 6: Number of instances of the 4 classes in the 2020 dataset

The size of the images varies from 600x600 to 720x720 depending on the country.
Here are some images belonging to this dataset.

Figure 7: Image from Japan

Figure 8: Image from Czech Republic

Figure 9: Image from India

2.2.2 Train models using YoloV5

To train models on the 2020 dataset, I first used the same approach as in 2.1.3 for the 2018 dataset.

I also wanted to compare the performance of the models according to the different countries. That is why I split the test set in 3 sets (Japan, Czech Republic and India) and analyzed the results.

After this, I used some methods to improve the performance of the detection, that I will be covering in the next sections.

2.2.2.1 Filtering images of the dataset

Analysing the dataset, we can find out that a considerable proportion of the images don't contain any label (more than 40%), either because there is no damage on the road or because the damages that can be observed don't belong to our set of 4 classes. For example, there is a lot of Japan images from the 2018 dataset which contain labels referring to other damages (because there were 8 in the 2018 dataset, whereas now we are interested in only 4 classes).

Therefore, I had the idea of removing the unlabeled images and observe the performance this new filtered dataset.

Country	Japan	Czech	India
nb of images with no labels	2606	1757	4483

Figure 10: Number of images without any label in the 2020 dataset according to the different countries

2.2.2.2 Hyperparameter evolution

I then used hyperparameter evolution as in section 2.1.3.3, and on the filtered dataset. Furthermore I used 15 epochs, a batch size of 20, as well as 50 generations.

2.2.2.3 Test-time augmentation

Test-time augmentation is a method used during testing. For each image tested, it consists in making predictions on augmented images from this image, and averaging these predictions in order to make the final prediction.

It is similar to data augmentation during training, but this method is used after the network is trained, only on test images, during testing.

In YoloV5, the default augmentations are left-right flipping and scaling to 3 different resolutions.

2.2.2.4 Model ensembling

Ensemble learning is a method using several neural network models to make the prediction. Looking at the results of the 2020 Global Road Damage Detection Challenge, the 3 first teams used ensemble learning to make their predictions. Therefore, I tried to use model ensembling, following the top team approach according to which the different models used were trained on different image sizes.

This method is compatible with test-time augmentation.

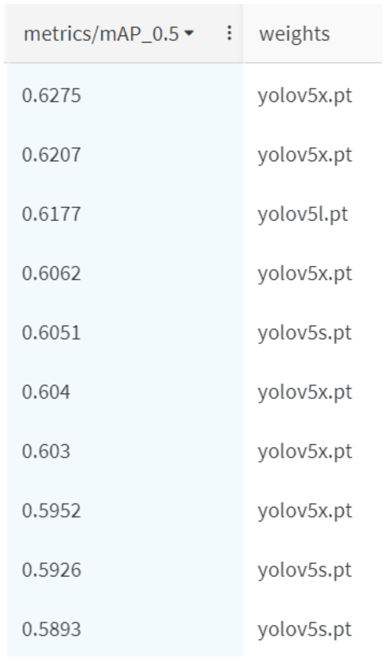
3 Results

This section is dedicated to the results I got when I tried the different approaches developed in the section 2. Therefore the structure of this section will be quite similar as section 2.

3.1 Work on the 2018 Dataset

3.1.1 Performance of the different architectures of YoloV5

I tested different sizes of models available in YoloV5 and compared the performances.



metrics/mAP_0.5	weights
0.6275	yolov5x.pt
0.6207	yolov5x.pt
0.6177	yolov5l.pt
0.6062	yolov5x.pt
0.6051	yolov5s.pt
0.604	yolov5x.pt
0.603	yolov5x.pt
0.5952	yolov5x.pt
0.5926	yolov5s.pt
0.5893	yolov5s.pt

Figure 11: mAP@0.5 of several training with different sizes of architectures

Looking at the Figure 11, we can observe that the Yolov5X model, that is to say the largest model available on YoloV5 seems to achieve the best results in terms of mAP@0.5 . That is why I chose to use only this model architecture in my work.

3.1.2 Weights initialisation

As mentioned in 2.1.3, we can train models with randomly initialised weights, but also with the weights of pre-trained models on the CoCo dataset.

After testing, I saw that the best results were achieved with the pre-trained weights. This is because the model trained on the CoCo dataset (which is a dataset of hundreds of thousands images) has the ability to well generalise to other datasets because it can extract good general features in the images. That is why I chose to continue my work only using pre-trained weights.

3.1.3 Results of the best model

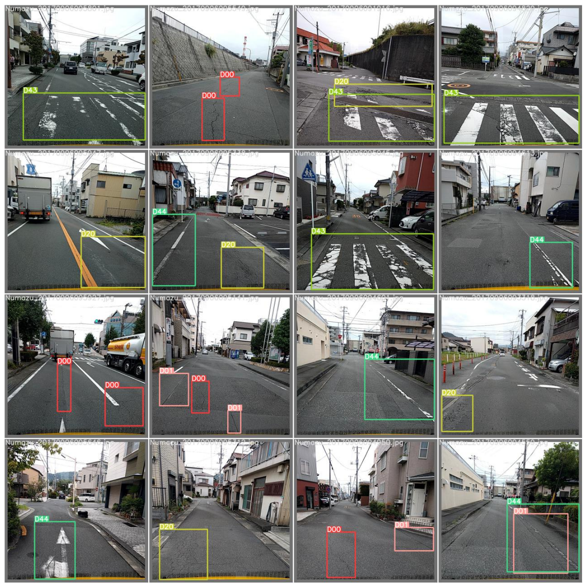


Figure 12: Labels of a batch of 16 images

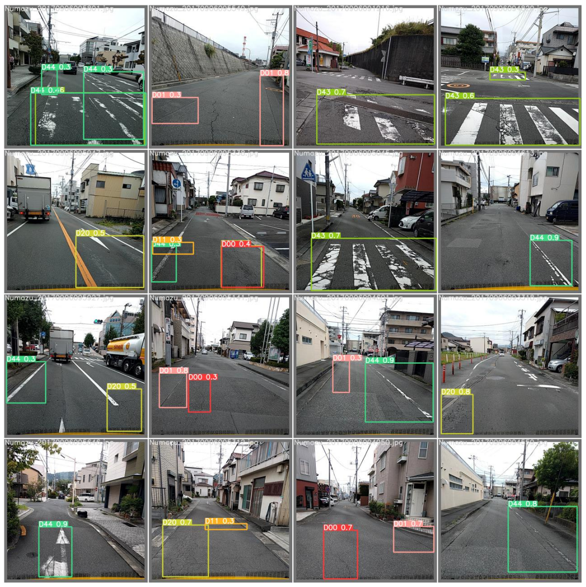


Figure 13: Predictions of a batch of 16 images

The best model I trained achieved an F1-score of 0.617. This is without any improvements like test-time augmentation or model ensembling.

Let's take a look at the confusion matrix.

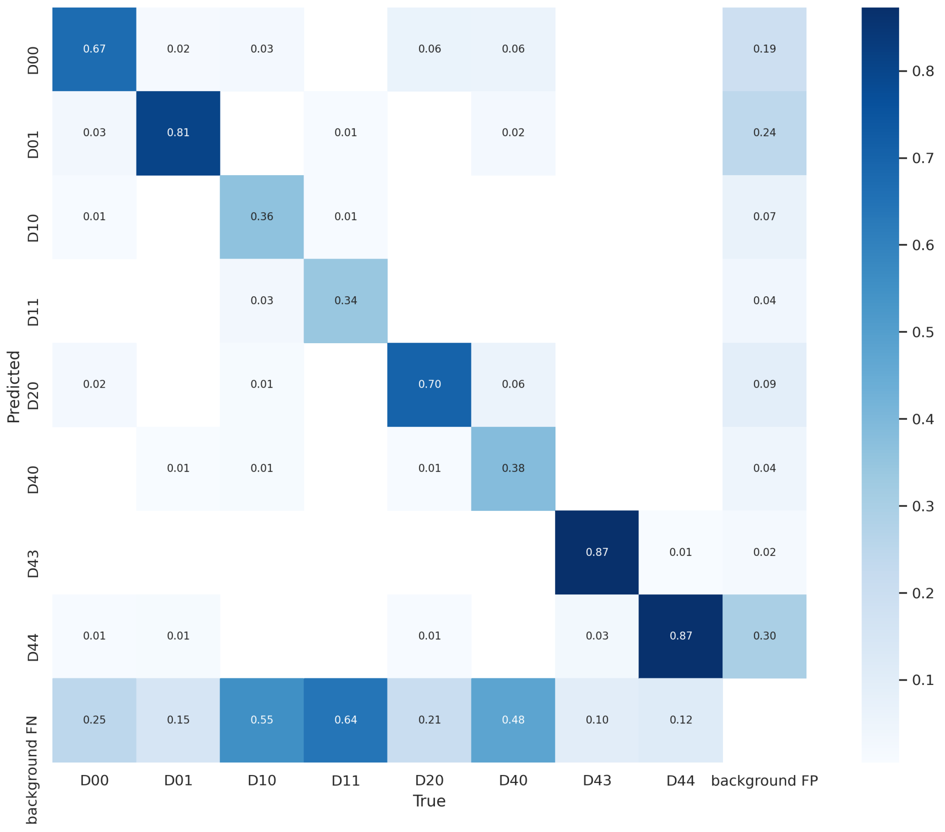


Figure 14: Confusion matrix for the best model trained on the 2018 dataset

We can first observe the low performance of the model on the D10, D11 and D40 classes : only between 30 and 40% of the damage of these class are well detected, and more than 50% are False Negative. Looking at the proportion of the classes in the dataset (cf Figure 2), we can see that these three classes have really few instances in the dataset compared to the other classes. YoloV5 has an augmentation process which seems not to lead to good results on these classes with few instances.

However we can also observe that the D43 class is very well predicted, although the dataset doesn't count a lot of instances of this class. This class as well as D44 are the classes on which the model performs the best. This is because these two classes correspond respectively to white line blur and cross walk blur, which are very easy to spot and hard to get mixed with other classes.

The other three classes are decently detected, with between 60 and 80% of good detections.

I then tried some improvements to increase the performance of the model.

3.1.4 Test-time augmentation

Here is the new confusion matrix when augmentation on the test data is used, as described in 2.2.2.3.

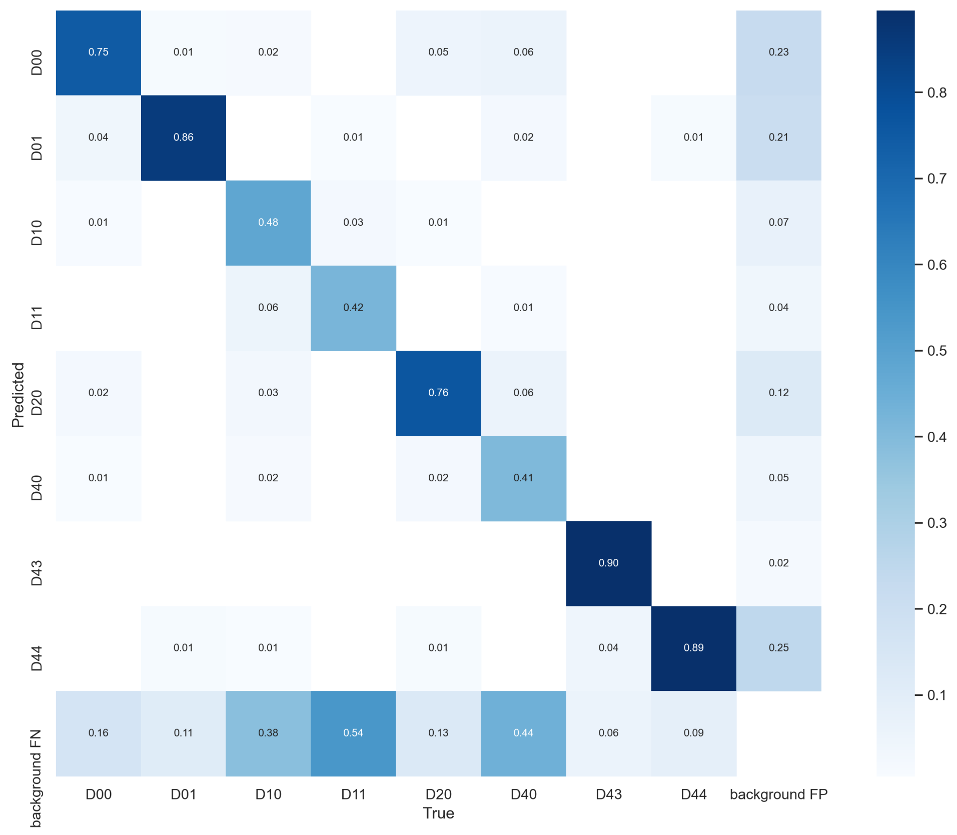


Figure 15: Confusion matrix for the best model trained on the 2018 dataset, with test-time augmentation

Using this method improved the F1-score by 0.01. We can see that the precision is increased for every class and the number of FN and FP is also reduced.

3.1.5 Model ensembling

For the model ensembling I used 3 different trained models : yolov5x trained on images of size 608 and 448, and yolov5l trained on images of size 608.

The following table shows the performance of the different trained models and of the model ensembling with these three models. TTA stands for test-time augmentation.

The precision and recall columns are the mean on all classes of the precision and recall.

v5x_608	v5x_448	v5l_608	TTA	Precision	Recall	mAP@0.5	mAP@0.5:0.95	F1-score
✓				0.626	0.609	0.621	0.344	0.617
✓			✓	0.622	0.627	0.636	0.354	**0.624**
	✓			0.647	0.587	0.626	0.341	0.616
	✓		✓	**0.659**	0.579	0.629	0.347	0.616
		✓		0.640	0.597	0.61	0.337	0.618
		✓	✓	0.599	0.637	0.636	0.353	0.617
✓	✓		✓	0.601	0.640	0.642	0.351	0.620
✓		✓	✓	0.578	**0.648**	**0.645**	**0.359**	0.611
	✓	✓	✓	0.595	0.643	0.643	0.354	0.618
✓	✓	✓	✓	0.599	0.637	0.636	0.357	0.617

Looking at the previous table, we can make some observations :

- The tests using test-time augmentation lead to better predictions in terms of mAP especially, but also in terms of F1-score.
- The model ensembling with yolov5x_608 and yolov5l_608 is the best in terms of recall, mAP@0.5 and mAP@0.5:0.95, which means that ensemble learning with different architectures can be a viable solution because the different models learned differently so they can spot different aspects of the images.
- The difference in terms of values are very small because the models trained have not a lot of differences and also because I only kept the best models trained and there is obviously an upper limit in terms of performance. We could have observed higher differences if the augmentation of the images on which each model is trained were completely different.

3.1.6 Conclusion on the 2018 dataset and possible improvements

The results achieved using yolov5 were not as good as the state of the art performance, which are accessible in [1]. When using yolov5, I'm sure improvements can be achieved by playing on the augmentation system. However most of the studies with the best models on this datast don't use Yolo, or at least don't use Yolo alone. [1]

3.2 Work on the 2020 Dataset

3.2.1 Using default hyperparameters and comparing the performance on the different countries

To understand the results achieved on this 2020 dataset, I split the test dataset into three subsets, each one containing images from one country.

The following bar chart represents the proportion of True Positives, False Negatives and False Positives according to the classes and the different countries.

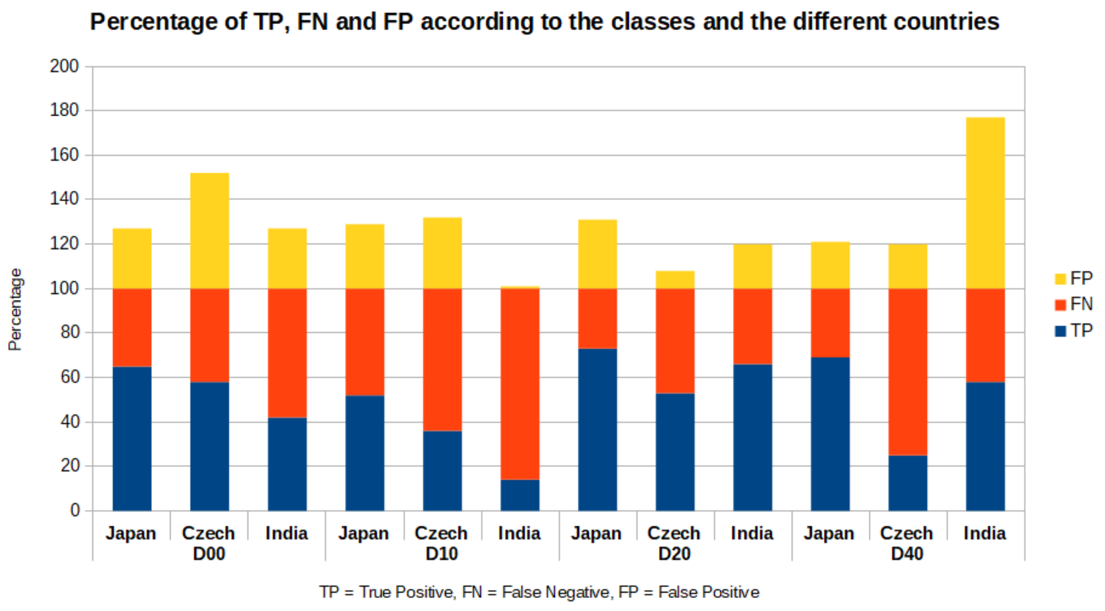


Figure 16: Percentage of True Positive (TP), False Positive (FP) and False Negative (FN) according to the different countries and the different classes

Note on how to read this bar chart : Let's take a look at the first bar : The class is D00 and the country is Japan. Around 60% of True Positives means that amongst all the damage corresponding to the D00 class in the test set, 60% of them were well detected. On the contrary, 40% were not detected. Finally the number of False Positives is around 20% of the total number of D00 damage.

Taking a look at the bar chart, we can make several observations :

- Japan is by far the country for which the predictions are the best. Indeed the percentage of TP is the highest for every class and the number of FP stays relatively low.
- On the contrary, India is the country with the less accurate predictions. We can see than less than 20% of the D10 damage are well predicted. Furthermore, there is a considerable number of FP for the D40 class in India, proving that the neural network tends to predict this class often, but the proportion of TP is still less than Japan.
- For Czech Republic, the results are not that good either, when we look at the D00 class (more than 50% of FP) and the two other classes have quite low TP proportion. (40% and 20%).

The observations show that the quality of the predictions depends a lot on the country where the images were taken. This is because of the differences in shape and color of the roads that we mentioned earlier.

3.2.2 Filtering images of the dataset

Filtering images didn't lead to better results than with the complete dataset.

3.2.3 Hyperparameter evolution

On the Figure 17, we can see the mean average precision at 0.5 IoU (mAP@0.5) according to the different values of the hyperparameters. The hyperparameters for which the cloud of points is a line have their value unmodified during the evolution.

As said in 2.1.3.3, I used 50 generations and each training was 15 epochs with a batch size of 20. Note that YoloV5 developers suggest to use at least 300 generations to have best results, but unfortunately I was limited in time and also in terms of computational power so I couldn't do more than 50 generations.

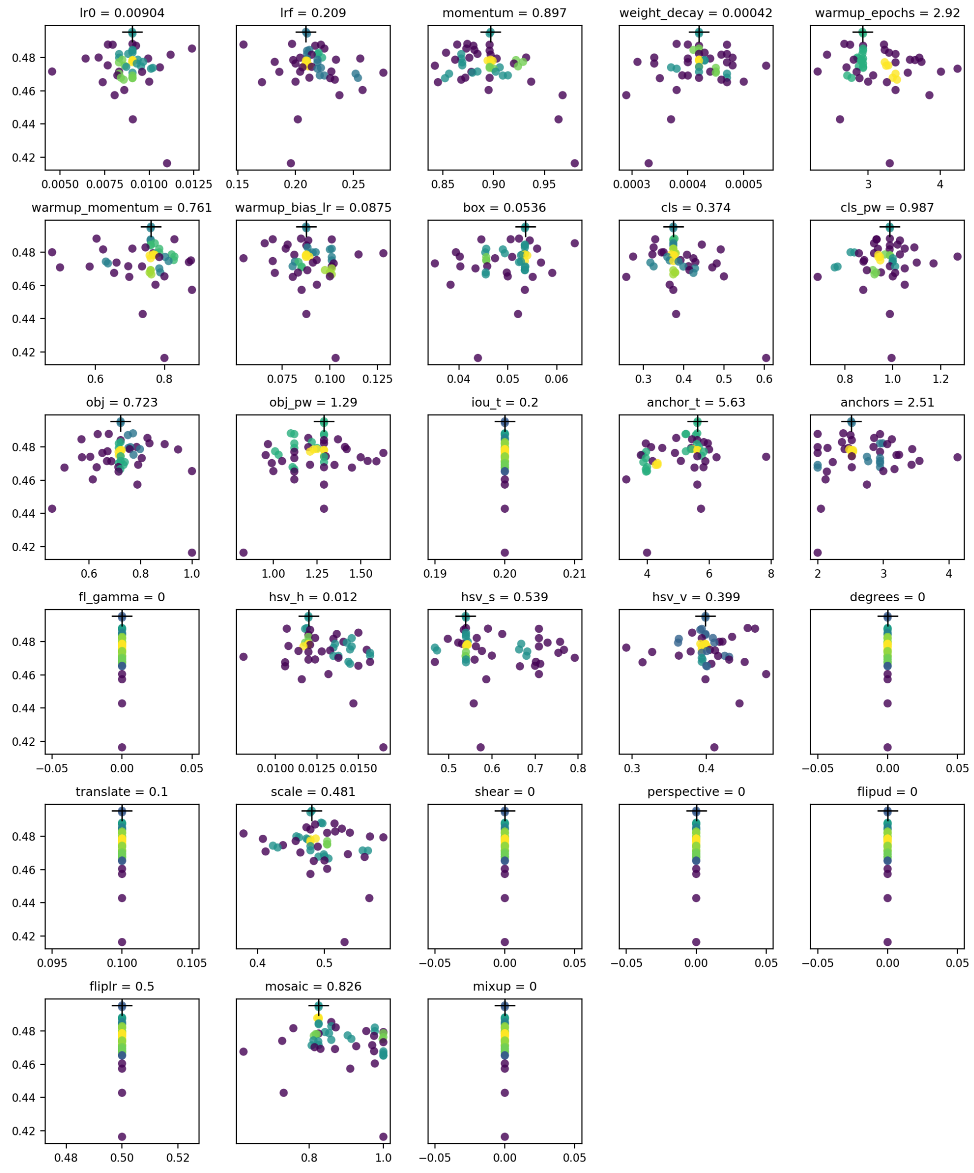


Figure 17: Results of the hyperparameter evolution on the 2020 all labeled dataset

Although the number of generations is not enough, we can see that some extreme values for some hyperparameters make the mAP fall. We can mention the training with a momentum higher than 0.95 for example.

Unfortunately it's hard to analyse such results, as changing only the number of epochs for example could produce completely different results in the training.

3.2.4 Model ensembling and test-time augmentation

I used three different trained models : 2 models trained with yolov5x with 608 image size, and one with yolov5x trained with 448 image size.

v5x_608	v5x_608_2	v5x_448	TTA	Precision	Recall	mAP@0.5	mAP@0.5:0.95	F1-score
✓				0.599	0.544	0.567	0.248	0.570
✓			✓	0.569	**0.573**	0.572	0.254	0.571
	✓			**0.608**	0.532	0.563	0.245	0.567
	✓		✓	0.585	0.562	**0.581**	**0.257**	**0.573**
		✓		0.545	0.540	0.541	0.231	0.542
		✓	✓	0.523	0.564	0.547	0.234	0.543
✓	✓		✓	0.584	0.553	0.576	0.257	0.568
✓		✓	✓	0.582	0.538	0.568	0.252	0.559
	✓	✓	✓	0.581	0.542	0.572	0.253	0.561
✓	✓	✓	✓	0.559	0.565	0.567	0.249	0.562

We can first notice that as for the 2018 dataset, test-time augmentation is improving the performance of the predictions. However model ensembling seems not to increase nor the mAP nor the F1-score. The best result we get when testing is achieved using test-time augmentation on one of the two models trained on yolov5 with 608x608 images.The F1-score is then 0.573 which can seem very low. Nonetheless, a competition called Global Road Damage Dataset Challenge has been organized on this dataset and only the top 6 teams achieved an F1-score above 0.56, on the 2 test sets that were available for the challenge. [6]

Unfortunately, the annotations to these 2 datasets are not available publicly so I can't test my own models on them.

4 Conclusion on the 2020 dataset and possible improvements

After analysing the results of the experiment, we can say that our models trained on YoloV5 are decent when compared to other models trained for the Global Road Damage Dataset Challenge. To improve the performance of the training, an hyperparameter evolution with at least 100 generations and 300 epochs per training could be a real help to find promising values for the hyperparameters. Moreover, this study heads in the same direction as the one made on this 2020 dataset [5], according to which mixing a dataset with road images from different countries is a good idea because it can increase the number of images and examples of damage, but can also be risky if the country we are interested in is too different from the other countries we use, as we saw for India.

References

[1] Hiroya Maeda, Yoshihide Sekimoto, Toshikazu Seto, Takehiro Kashiyama, Hiroshi Omata. *Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images: Road damage detection and classification*
`https://www.researchgate.net/publication/326087983_Road_Damage_Detection_and_Classification_Using_Deep_Neural_Networks_with_Smartphone_Images_Road_damage_detection_and_classification`

[2] Hiroya Maeda et al.
Github Road Damage Detector
`https://github.com/sekilab/RoadDamageDetector`

[3] Glenn Jocher et al.
YoloV5 GitHub repository
`https://github.com/ultralytics/yolov5`

[4] Glenn Jocher. *Hyperparameter Evolution*
`https://github.com/ultralytics/yolov5/issues/607`

[5] Deeksha Arya et al.
Transfer Learning-based Road Damage Detection for Multiple Countries
`https://arxiv.org/abs/2008.13101`

[6] Deeksha Arya et al.
Global Road Damage Detection: State-of-the-art Solutions
`https://arxiv.org/abs/2008.13101` https://arxiv.org/abs/2011.08740