NATIONAL CHIAO TUNG UNIVERSITY

Institute of Computer Science and Engineering

Selected Topics in Visual Recognition using Deep Learning

Homework 4

AUTHOR: ALESSANDRO SAVIOLO

STUDENT NUMBER: 0845086

19th December 2019

Summary

1. Introduction

The purpose of this document is to present and analyze the model built for Homework 4. The dataset used for training the model is Tiny PASCAL VOC dataset. Due to the small size of the dataset, trasfer learning is used. In particular, ImageNet pre-trained weights are used as a starting point for training.

The framework used is Detectron2 due to its simplicity and modular design which makes it easy it use. The model used is Mask R-CNN with ResNet50 as backbone.

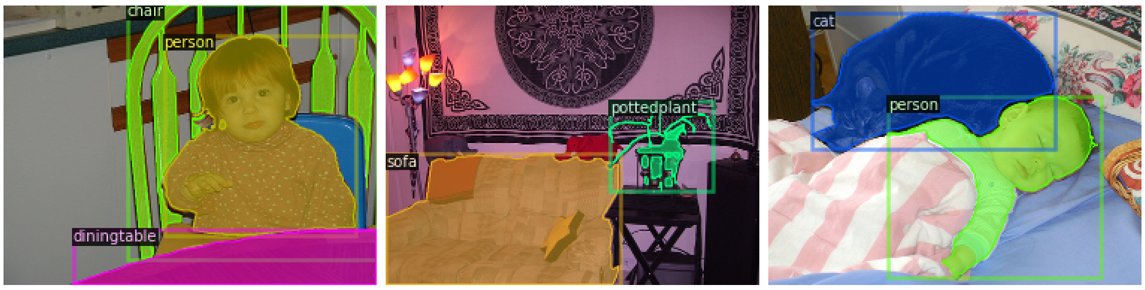
In Chapter 2, the dataset is analyzed. In Chapter 3, the model and the parameters are presented and discussed. In Chapter 3, the results obtained by using the trained model on the test set are illustrated. In particular, both correct and wrong results are presented and discussed. In Chapter 4, the conclusions are reported, also describing the first approach adopted to solve the problem proposed for Homework 4.

The code of the project has been uploaded to Github at the repository [5].

2. Methodology

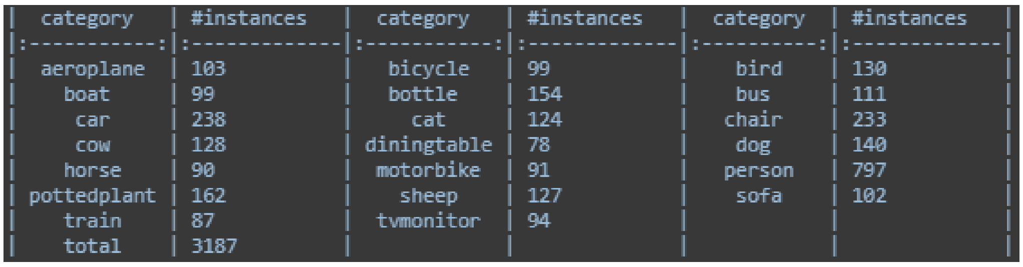
2.1 Dataset

This project uses the Tiny PASCAL VOC dataset. Tiny VOC dataset contains only 1349 training images and 100 test images with 20 common object classes (namely aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable, dog, horse, motorbike, person, pottedplant, sheep, sofa, train, tvmonitor). In Figure 2.1 some images from the training set are shown with the corresponding annotations. Each image may contain more than one class occurrence. In Figure 2.2 the number of occurrences of each class in the training images are listed.



(a) Train image (b) Train image (c) Train image

Figure 2.1: Training set images with corresponding annotations.



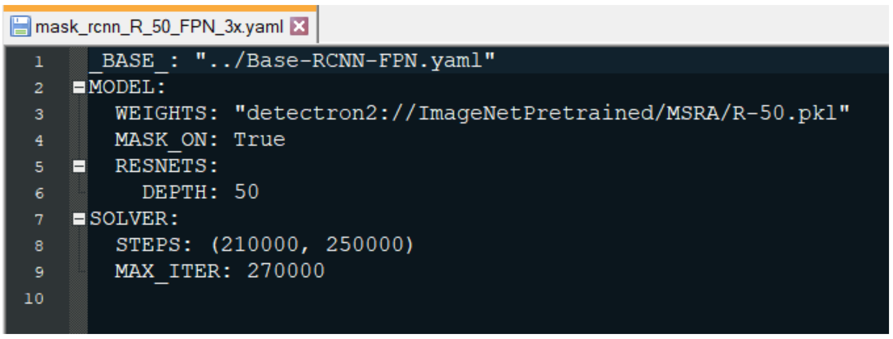
```
|  category   | #instances   |  category   | #instances   |  category  | #instances   |
|:-----------:|:-------------|:-----------:|:-------------|:----------:|:-------------|
|  aeroplane  | 103          |   bicycle   | 99           |    bird    | 130          |
|    boat     | 99           |   bottle    | 154          |    bus     | 111          |
|     car     | 238          |     cat     | 124          |   chair    | 233          |
|     cow     | 128          | diningtable | 78           |    dog     | 140          |
|    horse    | 90           |  motorbike  | 91           |   person   | 797          |
| pottedplant | 162          |    sheep    | 127          |    sofa    | 102          |
|    train    | 87           |  tvmonitor  | 94           |            |              |
|    total    | 3187         |             |              |            |              |
```

Figure 2.2: Classes occurrences in the training set.

2.2 Model and Parameters

In this project I used Detectron2 framework. Detectron2 is a Facebook AI Research's software system that implements state-of-the-art object detection algorithms, including Mask R-CNN. In particular, I used Mask R-CNN with ResNet50 as backbone.

To deal with the tiny dataset presented in Section 2.1, the proposed model is trained on top of ImageNet pre-trained weights. In Figure 2.3, the .yaml file used is shown.



```yaml
_BASE_: "../Base-RCNN-FPN.yaml"
MODEL:
  WEIGHTS: "detectron2://ImageNetPretrained/MSRA/R-50.pkl"
  MASK_ON: True
  RESNETS:
    DEPTH: 50
SOLVER:
  STEPS: (210000, 250000)
  MAX_ITER: 270000
```

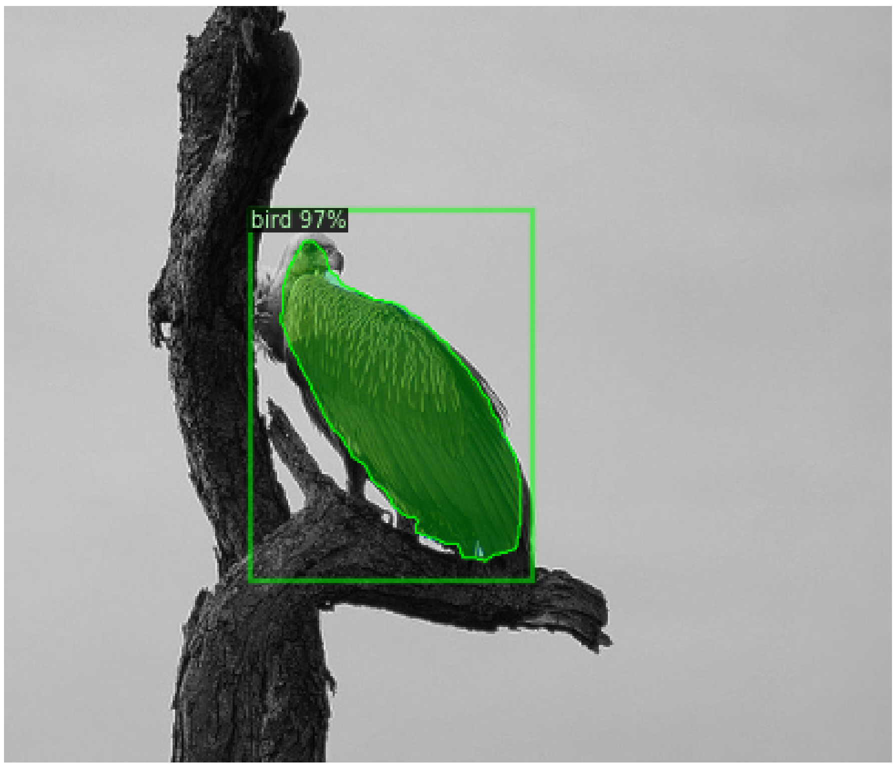
Figure 2.3: The .yaml file used for importing ImageNet pre-trained weights.

The parameters chosen for training the model are listed in Table 2.1. The learning algorithm used is Stochastic Gradient Descent (SGD). The initial learning rate is set equal to 0.00025. The warmup strategy presented in [3] is used. This learning schedule is proved to be particularly helpful for prototyping object detection and segmentation methods that fine-tune pre-trained layers together with newly initialized layers [1, 4]. The number of iterations used is equal to 20000. By observing the log produced by the model during the training process, it seems to converge to a good solution after around 18000 iterations. By using this setting, the time required by the model to fully train is about 4 hours on Google Colab GPU [7].

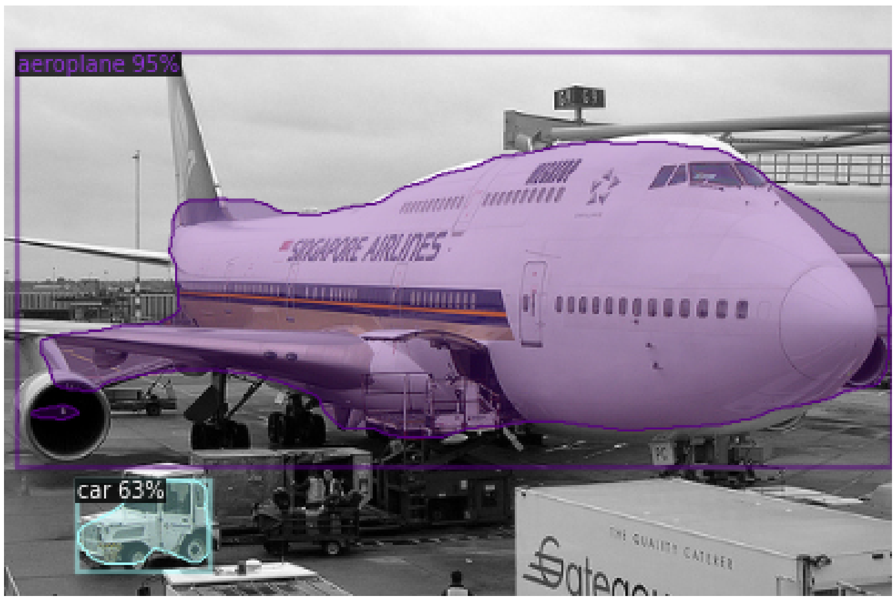
(Hyper) Parameter	Value
Backbone Model	ResNet50
Learning Algorithm	SGD
Starting Learning Rate	0.00025
Images x Batch	2
Batch Size x Image	128
# Iterations	20000

Table 2.1

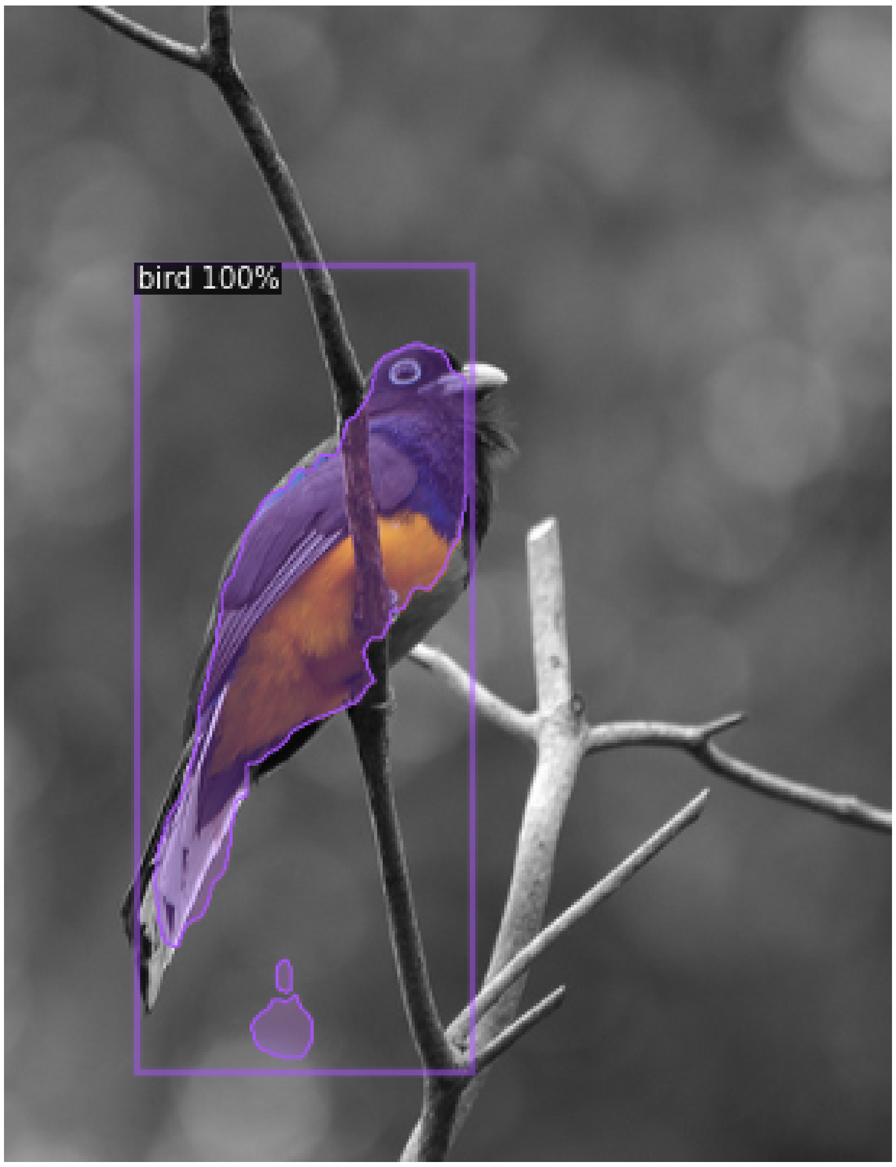
3. Results



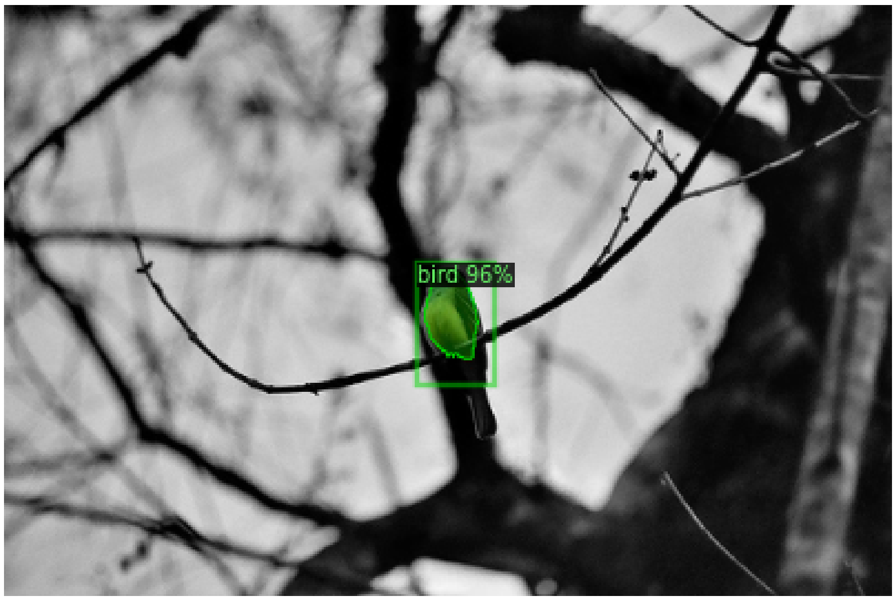
(a)



(b)

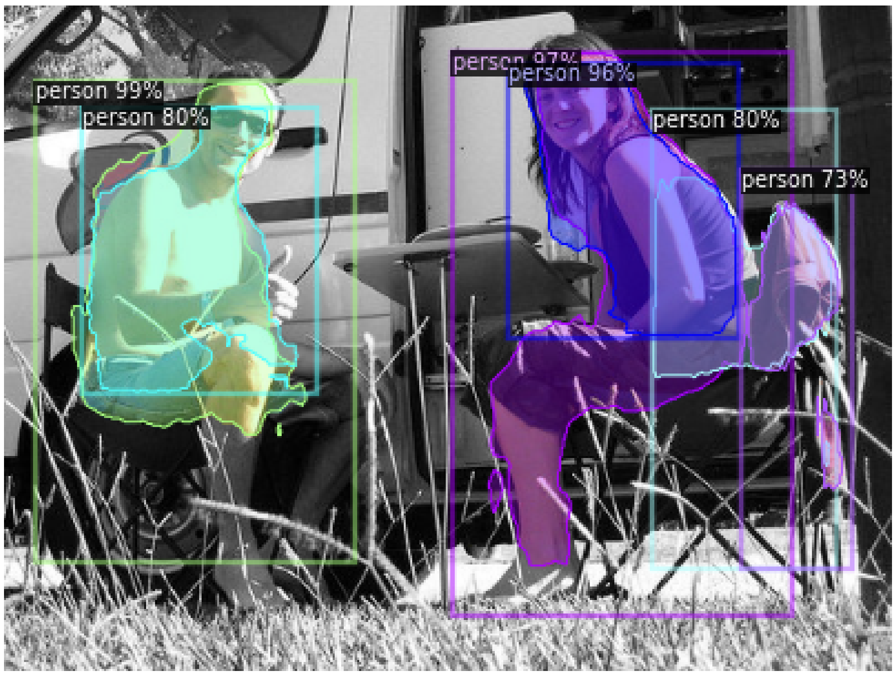


(c)

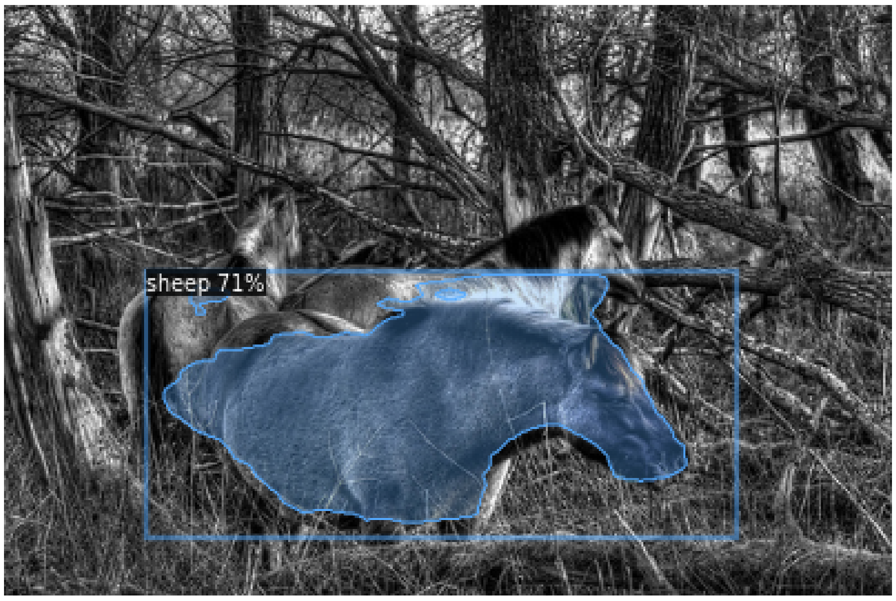


(d)

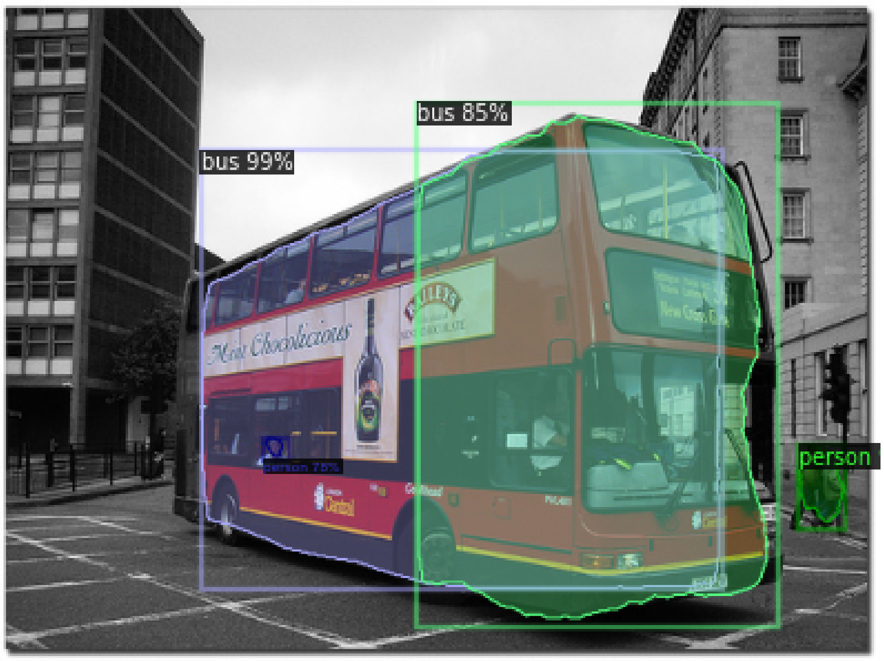
Figure 3.1: The model is capable of detecting and correctly labeling objects when they are not overlapping with other objects. Moreover, in such cases the detection scores are very accurate.


person 99%
person 80%
person 97%
person 96%
person 80%
person 73%

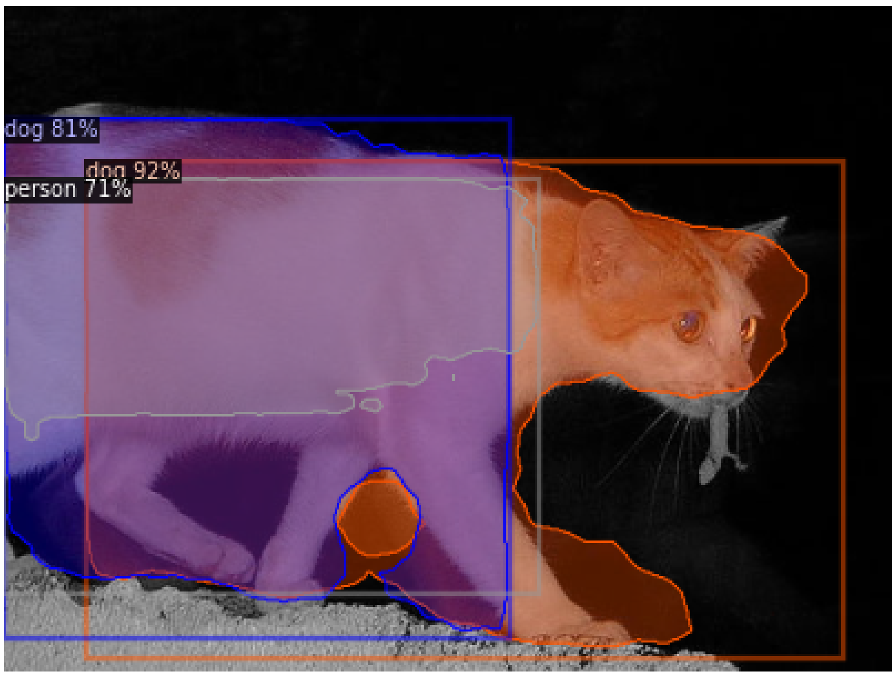
(a)


sheep 71%

(b)



(c)



(d)

Figure 3.2: The model tends to make wrong predictions when the objects in the image are overlapping with other objects. This may be due to the tiny training set used to train the model, which makes it return poor results when the objects are not clearly defined in the image.

4. Conclusion

The first attempt to execute this task I have tried consisted in directly using Mask R-CNN built on Feature Pyramid Network (FPN) and ResNet101 as backbone. The implementation of such model followed the Mask R-CNN paper [2] for the most part. I tried to improve the paper implementation by modifying the learning schedule. In particular, the paper uses a learning rate of 0.02, but I found it to be too high since it often caused the weights to explode (i.e., exploding gradient problem). This is enhanced especially when using a small batch size, which is the case of this task, since a batch size bigger than 4 causes Out Of Memory error on Google Colab GPU. Therefore, I used a smaller learning rate, defining a schedule which consisted of using a learning rate equal to 0.001 for firstly training the last layers of the model (keeping the other layer weights frozen), and then training the whole model with a learning rate equal to 0.0001. By using this schedule, it was possible to firstly adjust the pretrained model to the considered task/dataset, and then fine tune the whole model.

The model was returning promising performances, but the computational time for training it for 10 epochs only, with 375 iterations for each epoch, was about 10 hours running the process on Google Colab. This computational bottleneck made me try to use another framework like Detectron2, which revealed to be very powerful and computational less expansive.

Detectron2 has a simple and modular design, which makes it easy to use. It is written using pytorch, which is one of the reasons why it offers good performances. Future works include using more data augmentation to improve the generalization capabilities of the model and try to use ResNet152 to use a deeper and more powerful model.

APPENDIX

A.1 GitHub Repository

The code of the project is available on GitHub at the repository [5].

A.2 Credits

The Detectron2 GitHub Repository [6] has deeply helped the development of this project.

Bibliography

[1] Girshick R. *Fast R-CNN*. ICCV, 2015.

[2] He K., Gkioxari G., Dollar P., and Girshick R. *Mask R-CNN*. Facebook AI Research (FAIR).

[3] He K., Zhang X., Ren S., and Sun J. *Deep residual learning for image recognition*. CVPR, 2016.

[4] Ren S., He K., Girshick R., and Sun J. *Faster R-CNN: Towards real-time object detection with region proposal networks*. NIPS, 2015.

Sitography

[5] AlessandroSaviolo Github Repository (2019), last access 19 Dec. 2019, [Online]. *https://github.com/AlessandroSaviolo/CS_IOC5008_0845086_HW4*

[6] Detectron2 Github Repository (2019), last access 19 Dec. 2019, [Online]. *https://github.com/facebookresearch/detectron2*

[7] Google Colab (2015), last access 19 Dec. 2019, [Online]. *https://colab.research.google.com*