Reproduction: Blurring sensitive data in panoramic images

J.m.c.van Dijk, Y. Lo-Fo-Sang

April 2021

Figure 1: Picture of a local street

1 Introduction

This paper features a reproduction of the results displayed in the paper [1] Blurring sensitive data in panoramic images by Laurens Samson and Chris Eijgenstein. The reproduction has been commissioned by the 'Deep learning' course of the TU Delft. Not all results are replicated. We focus on the results of the Faster R-CNN (detectron2) network and the YOLOv5 network. The goal of the aforementioned paper is to detect license plates and people in panoramic images.

[1] https://drive.google.com/file/d/1nXeoM6p2fOlYaFwaMfjz_KnABdQDRYT0/view

2 Dataset

The dataset that was provided to us consisted of 3743 annotated images in the YOLOv5 format. The images were scaled down from a resolution of 8000 by 4000 to 2000 by 1000. The authors had to choose a fitting resolution for the data set images where a network would still be capable of recognizing far away objects without wasting a lot of performance. The authors decided upon the resolution of 2000 by 1000 with empirical testing. They found that a lower resolution results in a loss of detections for far away objects.

3 Process

The authors provided us with the dataset of 3743 annotated images and gave us a configuration file with the hyper parameters that they used for their YOLOv5 training. We decided to reproduce the YOLOv5 results and the Detectron2 results. Mainly due to the precision rates of these methods compared to the others.

3.1 YOLOv5

The YOLOv5 model used by the Amsterdam-AI group has been made by the Ultralytics. Ultralytics is a U.S.-based particle physics and AI startup with over 6 years of expertise. https://github.com/ultralytics. YOLO is an open source object detection model which has gone through several different versions, ranging from YOLOv1 to the most current YOLOv5 model. The model can be trained on image data with bounding box annotations. The bounding box annotation for each class instance consists of the class, the normalised x-center coordinate, the normalised y-center coordinate and the normalized width and height of the bounding box. Within the YOLOv5 model, there are different sizes which the user can choose from, namely YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x, which contain 7.3 to 87.7 million parameters respectively. For the reproduction, YOLOv5s was used as training the larger models takes a large amount of GPU hours and the results were deemed sufficient.

Training the model is done in a few lines of code.

- Cloning the repo
- Installing dependencies, it is advisable to install pytorch and cuda (in case the user has an Nvidia GPU) separately to avoid errors
- Specifying where to find the data in a coco.yaml file
- Specifying which model (YOLOv5s,.. , Yolov5x) has to be used and the number of classes the model has to be trained on

- Training the model specifying the image size, batch size, number of epochs and starting weights. Caching is optional as this takes quite some video-ram.
- Storing the results

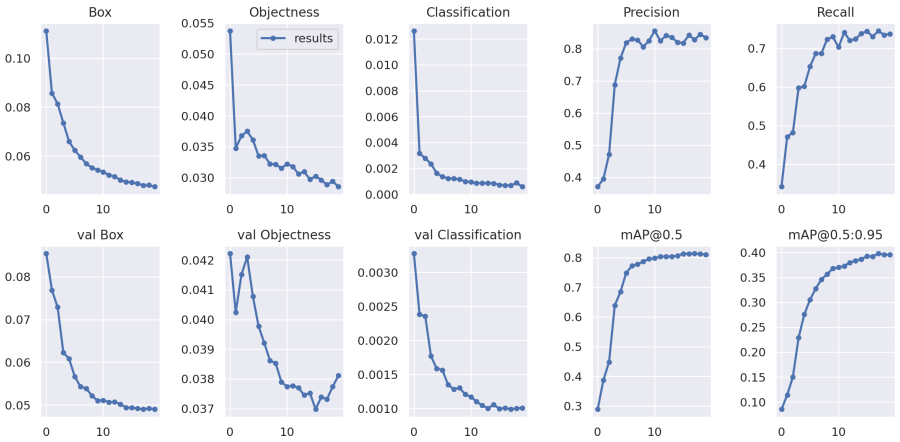

Figure 2: Results of YOLOv5 training epochs

3.2 Detectron2

Detectron2[2] is a software library that allows the usage of Faster R-CNN. It is developed by Facebook and supports a lot of features. We will be using it for object detection with bounding boxes.

The dataset had to be converted to a format that is compatible with Detectron2. We decided to use the COCO dataset format[3]. We used Yolo-to-COCO-format-converter[4] by Tae Young. This script converts the given YOLOv5 format to COCO format. It did however error on a few entries in the dataset train.txt due to some missing images. This was easily resolved by removing those paths. This resulted in a loss of 3 images making our dataset consist of 3740 images for the detectron2 dataset. The error was likely a client side error that happened during the process of managing folders, but it was ignored due to the slow file handling speed of the used PC.

How we trained the model:

- Cloning the repo[5]

[2]https://github.com/facebookresearch/detectron2
[3]https://cocodataset.org/#home
[4]https://github.com/Taeyoung96/Yolo-to-COCO-format-converter
[5]https://github.com/julian9499/amsterdam-ai-reproduction

- Installing the dependencies, it is advisable to install pytorch and cuda (in case the user has an Nvidia GPU) separately to avoid errors. The training can be done on CPU if necessary but it is a lot slower
- Installing detectron2 by either building it locally or downloading a prebuilt version. More information about this can be found on `https://github.com/facebookresearch/detectron2/blob/master/INSTALL.md`
- Specifying a model to start with. In our case we chose the faster_rcnn_R_101_C4_3x model.
- Specifying where to find the coco datasets and speciying the iterations, batch size and learning rate.
- Training the model by running it until all iteration are executed.
- The resulting weights are stored multiple times during a training session as are the final weights in the output folder

4 Results

The results differ a lot compared to the results from the paper. The YOLOv5 results are almost the same as the one displayed in the paper, but the detectron2 results are worse than the results of the paper. We contribute this to the missing of the starting weights. Our randomly selected starting weights are likely to be a bad fit for the problem since we did not have the time to look into all options for our problem.

	Precision	Recall	AP50	AP
Yolov5	0.8349	0.7373	0.8101	0.3958
Detectron2	0.259	0.471	0.477	0.188

Table 1: Results of training

5 Discussion

The reproduction was not trivial. The report did not go into detail about the amount of epochs that were used for the given results. The dataset was also only available in the Yolov5 format. Which is of course translatable to COCO format, but it does add an extra necessary step for reproduction. The detectron2 results are very dependent on the choice of starting model. This was also not specified in the paper. We did ask the original authors for specification, but got no response. The paper misses a lot of specific information which would be required for a perfect reproduction, but it does feature enough information to resemble the results in one way or another.