

Configuration Manual

MSc Research Project
Data Analytics

Maria Raap
Student ID: x19141700

School of Computing
National College of Ireland

Supervisor: Majid Latifi

National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Maria Raap
Student ID:	x19141700
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Majid Latifi
Submission Due Date:	23/09/2021
Project Title:	Configuration Manual
Word Count:	1037
Page Count:	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	22nd September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Maria Raap
x19141700

1 Introduction

The configuration manual lays out hardware specifications, software requirements and different stages of implementation of the 'Vehicle Damage Detection using Semi-Supervised Object Detection' project in detail. Section 2 details about system requirements including the hardware specification and software requirements. Section 3 describes the data source followed by section 4 detailing the steps required to complete the data pre-processing. In section 5 outlines the required steps to execute the different models. The configuration manual concludes with section 6 outlining the evaluation metrics.

2 System Configuration

This section details system requirements and software required for implementation.

2.1 Hardware Requirement

- Virtual Machine: Azure Standard NC6 with 6 vCpus
- RAM: 56GB
- System Type: Ubuntu 18.04 LTS
- GPU: NVIDIA Tesla K80 GPU
- Storage: 512GB SSD

2.2 Software

- **Docker:** Docker is an open-source platform for developing, shipping, and running applications. Docker enables the separation of applications from the infrastructure so software can be delivered quickly. The application can be downloaded from the docker website¹.
- **Coco Annotator:** Coco Annotator is a web-based image annotation tool used for labelling images to create training data for image localization and object detection. The application can be downloaded from Github².

¹<https://www.docker.com/products/docker-desktop>

²<https://github.com/jsbroks/coco-annotator.git>

- **Cuda Toolkit:** Cuda Toolkit is an open-source parallel computing platform and the application programming interface model that allows the use of a CUDA-enabled graphics processing unit for general-purpose processing – an approach termed 'GPGPU'. The application can be downloaded from the CUDA website³.
- **Anaconda3:** Anaconda3 is an open-source platform that can be downloaded from anaconda distribution website ⁴. The platform supports various integrated design frameworks (IDD) for python programming. The models are built for particular environments using the below-listed libraries.

- Python 3.6.13
- Libraries
 - * numpy 1.19.5
 - * tensorflow 1.3.0
 - * keras 2.0.8
 - * GCC 7.5
 - * CUDA 10.0 (base Mask R-CNN)
 - * CUDA 10.2 (enhanced Mask R-CNN)
 - * detectron2 arch flags 3.7
 - * PyTorch 1.9.0+cu102
 - * Pillow 8.3.1
 - * torchvision 0.10.0+cu102
 - * iopath 0.1.9
 - * opencv-python 4.5.3
 - * h5py
 - * imgaug
 - * IPython[all]
 - * scipy
 - * matplotlib
 - * scikit-image

3 Datasource Description

The images for the project are downloaded from the website: <https://www.kaggle.com/>. The relevant datasets contain several images with varying degrees of vehicle damages. There is no uniform image size or quality. The overview below provides an overview of images imported downloaded.

³<https://developer.nvidia.com/cuda-toolkit>

⁴<https://www.anaconda.com/products/individual>

Table 1: Example images retrieved from Kaggle.com



4 Data Pre-Processing and Exploratory Analysis

Data pre-processing is undertaken in two parts. The first part is the data cleansing, including the review of and ultimately the selection of appropriate images. Since this project focuses on vehicle damage detection, images like total wreckage as well as low-quality images were removed from the overall dataset. This step was carried out in Ubuntu's default image viewer. If the decision was made to remove an image, it was simply removed from the folder. Part two is the annotation of the images which was carried out in Coco Annotator. Here the images are loaded to the relevant folder location and can then be viewed in the dataset view, see figure 1. The annotation of the image itself is done in the single image view. Before the annotation and labelling can commence, the object categories need to be defined, here only one category 'Damaged' is used. For this project the annotations were added as polygons to cater for the asymmetric character of vehicle damages, it is possible to label more than one damage object in one picture. Figure 2 as reference for the annotation tool view.

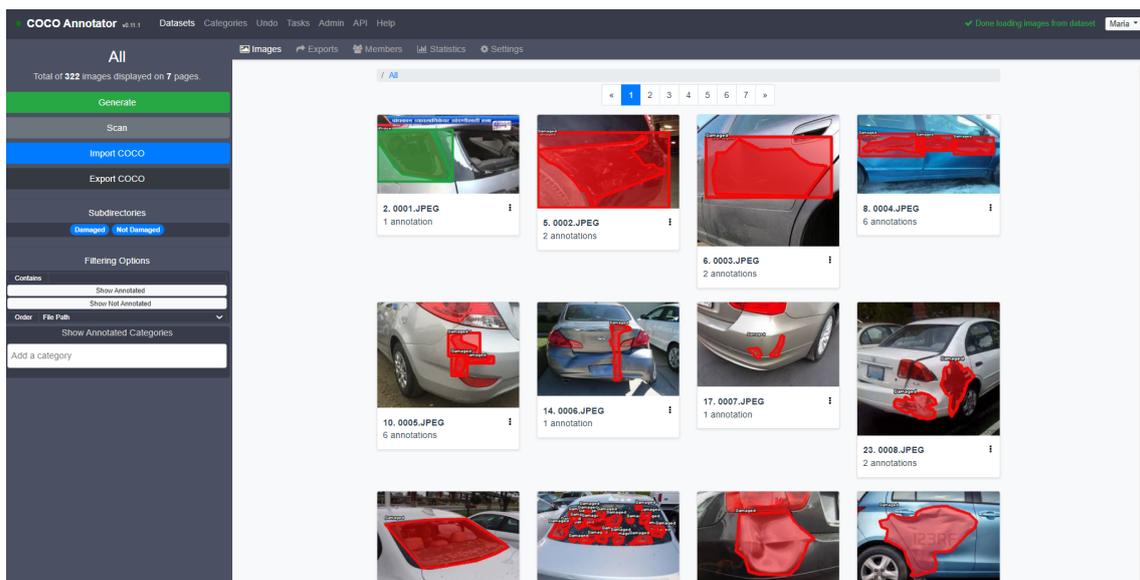


Figure 1: Coco Annotator Dataset Overview

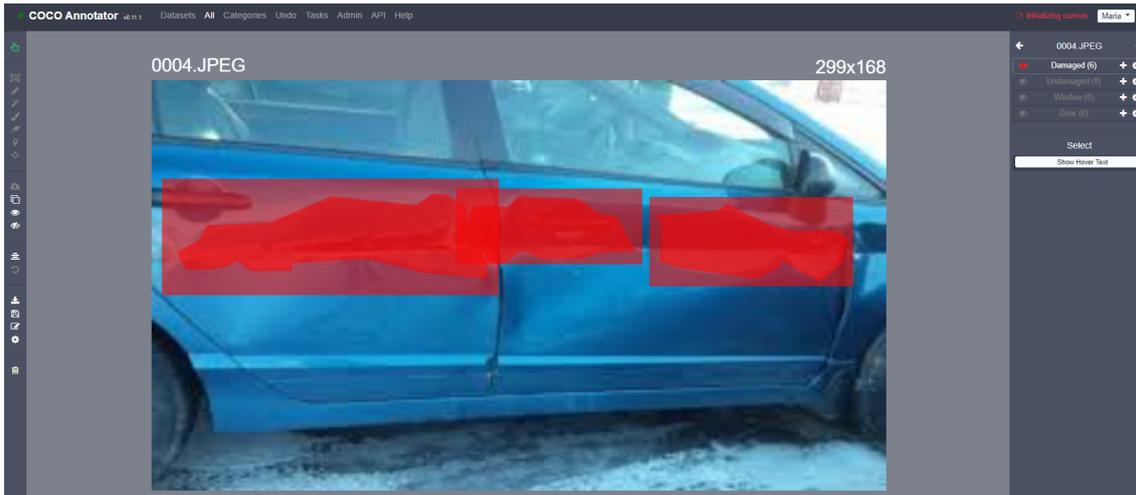


Figure 2: Coco Annotator Labelling tool

5 Model Training

Before the model can be run, anaconda3 must be installed and the relevant environment for the run be set up. The detailed setup is described in section 5.1 for the basic Mask R-CNN model and section 5.2 for the semi-supervised enhanced model.

5.1 Model Implementation Mask R-CNN

- Conda environment setup:
 - `conda env create -f environment.yml`
 - `conda activate mask-rcnn`
- Training:
 - Train a new model starting from pre-trained weights


```
python3 training.py -dataset=/path/to/dataset
weight=/path/to/pretrained/weight.h5
```
 - Resume training a model


```
python3 training.py -dataset=/path/to/dataset
weight=/path/to/pretrained/weight.h5
```
- Testing
 - Image


```
python3 image_detection.py -dataset=/path/to/dataset
weights=/path/to/pretrained/weight.h5 -image=/path/to/image/directory
```
- Annotation generating:


```
python3 annotating_generation.py -dataset=/path/to/dataset
weights=/path/to/pretrained/weight.h5 -image=/path/to/image
```

```
(mask-rcnn) x19141700@UbuntuVM:/datadrive/Mask_R-CNN$ python3 training.py --dataset=dataset/custom_dataset --weight=trained_weight/mask_rcnn_custom_dataset_0020.h5
Using TensorFlow backend.
Pre-trained weight: /datadrive/Mask_R-CNN/trained_weight/mask_rcnn_custom_dataset_0020.h5
Dataset: /datadrive/Mask_R-CNN/dataset/custom_dataset
Logs: /datadrive/Mask_R-CNN/Logs
Continue Train: None

Configurations:
BACKBONE                resnet101
BACKBONE_STRIDES        [4, 8, 16, 32, 64]
BATCH_SIZE              1
BBOX_STD_DEV            [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE None
DETECTION_MAX_INSTANCES 50
DETECTION_MIN_CONFIDENCE 0.9
DETECTION_NMS_THRESHOLD 0.2
FPN_CLASSIF_FC_LAYERS_SIZE 1024
GPU_COUNT               1
GRADIENT_CLIP_NORM     5.0
IMAGES_PER_GPU         1
IMAGE_CHANNEL_COUNT     3
IMAGE_MAX_DIM           512
IMAGE_META_SIZE        14
IMAGE_MIN_DIM          800
IMAGE_MIN_SCALE         0
IMAGE_RESIZE_MODE      square
IMAGE_SHAPE             [512 512  3]
LEARNING_MOMENTUM      0.9
LEARNING_RATE           0.001
LOSS_WEIGHTS           {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE         14
MASK_SHAPE              [28, 28]
MAX_GT_INSTANCES       50
MEAN_PIXEL              [123.7 116.8 103.9]
MINI_MASK_SHAPE        (56, 56)
NAME                   custom_dataset
NUM_CLASSES             2
POOL_SIZE              7
POST_NMS_TOPK_THRESHOLD 1000
```

Figure 3: Mask RCNN Training initiation sample

5.2 Model Implementation enhanced Mask R-CNN with 'Un-biased Teacher'

- Conda environment setup:
 - conda create -n detectron2 python=3.6
 - conda activate detectron2
 - conda install pytorch==1.9.0 torchvision -c pytorch
 - python -m pip install 'git+https://github.com/facebookresearch/detectron2.git'
- Training:
 - Train a new model with 1% labelled data and ResNet50


```
python train_net.py --num-gpus 1
config configs/coco_supervision/rcnn_R_50_FPN_sup01_run1_cd.yaml
SOLVER.IMG_PER_BATCH_LABEL 1
SOLVER.IMG_PER_BATCH_UNLABEL 1
```
 - Train a new model with 1% labelled data and ResNet101


```
python train_net.py --num-gpus 1
config configs/coco_supervision/rcnn_R_101_FPN_sup01_run1_cd.yaml
SOLVER.IMG_PER_BATCH_LABEL 1
SOLVER.IMG_PER_BATCH_UNLABEL 1
```
 - Train a new model with 5% labelled data and ResNet50


```
python train_net.py --num-gpus 1
config configs/coco_supervision/rcnn_R_50_FPN_sup05_run1_cd.yaml
SOLVER.IMG_PER_BATCH_LABEL 1
SOLVER.IMG_PER_BATCH_UNLABEL 1
```
 - Train a new model with 5% labelled data and ResNet101

- ```
python train_net.py --num-gpus 1
config configs/coco_supervision/rcnn_R_101_FPN_sup05_run1_cd.yaml
SOLVER.IMG_PER_BATCH_LABEL 1
SOLVER.IMG_PER_BATCH_UNLABEL 1
```
- Train a new model with 10% labelled data and ResNet50

```
python train_net.py --num-gpus 1
config configs/coco_supervision/rcnn_R_50_FPN_sup10_run1_cd.yaml
SOLVER.IMG_PER_BATCH_LABEL 1
SOLVER.IMG_PER_BATCH_UNLABEL 1
```
  - Train a new model with 10% labelled data and ResNet101

```
python train_net.py --num-gpus 1
config configs/coco_supervision/rcnn_R_101_FPN_sup10_run1_cd.yaml
SOLVER.IMG_PER_BATCH_LABEL 1
SOLVER.IMG_PER_BATCH_UNLABEL 1
```

```
(detectron2) x19141700@ubuntu18:/datadrive/unbiased-teacher$ python train_net.py --num-gpus 1 --config configs/coco_supervision/rcnn_R_101_FPN_sup50_run1_cd.yaml SOLVER.IMG_PER_BATCH
OLVER.IMG_PER_BATCH_UNLABEL 1
Command Line Args: Namespace(config_file='configs/coco_supervision/rcnn_R_101_FPN_sup50_run1_cd.yaml', dist_url='tcp://127.0.0.1:50152', eval_only=False, machine_rank=0, num_gpus=1, num_machines=1, opt:
[08/15 12:44:55 detectron2]: Rank of current process: 0. World size: 1
[08/15 12:44:55 detectron2]: Environment info:

sys.platform linux
Python 3.6.13 |Anaconda, Inc.| (default, Jun 4 2021, 14:25:59) [GCC 7.5.0]
numpy 1.19.5
detectron2 0.5 @/datadrive/anaconda3/envs/detectron2/lib/python3.6/site-packages/detectron2
Compiler GCC 7.5
CUDA compiler CUDA 10.2
detectron2 arch flags 3.7
DETECTRON2_ENV_MODULE
PyTorch 1.9.0+cu102 @/datadrive/anaconda3/envs/detectron2/lib/python3.6/site-packages/torch
PyTorch debug build False
GPU available Yes
GPU 0 Tesla K80 (arch=3.7)
CUDA_HOME /usr/local/cuda-10.2
Pillow 8.3.1
torchvision 0.10.0+cu102 @/datadrive/anaconda3/envs/detectron2/lib/python3.6/site-packages/torchvision
torchvision arch flags 3.5, 5.0, 6.0, 7.0, 7.5
fvcore 0.1.5.post20210804
logpath 0.1.9
cv2 4.5.3

PyTorch built with:
 - GCC 7.3
 - C++ Version: 201402
 - Intel(R) Math Kernel Library Version 2020.0.0 Product Build 20191122 for Intel(R) 64 architecture applications
 - Intel(R) MKL-DNN v2.1.2 (Git Hash 98c79878af4711c966c8f3594129c82913cdd)
 - OpenMP 201511 (a.k.a. OpenMP 4.5)
 - NNPACK is enabled
 - CPU capability usage: AVX2
 - CUDA Runtime 10.2
 - NVCC architecture flags: -gencode;arch=compute_37,code=sm_37;-gencode;arch=compute_50,code=sm_50;-gencode;arch=compute_60,code=sm_60;-gencode;arch=compute_70,code=sm_70
 - CUDNN 7.6.5
 - Magma 2.5.2
 - Build settings: BLAS_INFO=mkl, BUILD_TYPE=Release, CUDA_VERSION=10.2, CUDNN_VERSION=7.6.5, CXX_COMPILER=/opt/rh/devtoolset-7/root/usr/bin/c++, CXX_FLAGS=-Wno-deprecated -fvisibility-inlines-hidden
```

Figure 4: enhanced Mask RCNN Training initiation sample

```
[08/15 12:45:33 d2.utils.events]: eta: 2 days, 23:42:28 iter: 19 total_loss: 1.737 loss_cls: 1.09 loss_box_reg: 0.009333 loss_rpn_cls: 0.6253 loss_rpn_loc: 0.03337 time: 1.4445 data_time: 0.030
0.0019981 max_mem: 2808M
[08/15 12:46:04 d2.utils.events]: eta: 3 days, 2:50:21 iter: 39 total_loss: 0.6181 loss_cls: 0.1973 loss_box_reg: 0.04502 loss_rpn_cls: 0.2996 loss_rpn_loc: 0.02774 time: 1.4869 data_time: 0.06
0.0039961 max_mem: 2808M
[08/15 12:46:35 d2.utils.events]: eta: 3 days, 4:05:41 iter: 59 total_loss: 0.4289 loss_cls: 0.2076 loss_box_reg: 0.0654 loss_rpn_cls: 0.1107 loss_rpn_loc: 0.01932 time: 1.5194 data_time: 0.006
0.0059941 max_mem: 2808M
[08/15 12:47:06 d2.utils.events]: eta: 3 days, 5:02:59 iter: 79 total_loss: 0.3035 loss_cls: 0.0926 loss_box_reg: 0.07925 loss_rpn_cls: 0.08683 loss_rpn_loc: 0.02522 time: 1.5245 data_time: 0.0
0.0079921 max_mem: 2808M
[08/15 12:47:38 d2.utils.events]: eta: 3 days, 5:04:54 iter: 99 total_loss: 0.2478 loss_cls: 0.05147 loss_box_reg: 0.09471 loss_rpn_cls: 0.07097 loss_rpn_loc: 0.02417 time: 1.5411 data_time: 0.0
0.0099901 max_mem: 2808M
[08/15 12:48:10 d2.utils.events]: eta: 3 days, 5:33:36 iter: 119 total_loss: 0.2877 loss_cls: 0.06114 loss_box_reg: 0.1406 loss_rpn_cls: 0.0613 loss_rpn_loc: 0.01786 time: 1.5491 data_time: 0.0
0.011988 max_mem: 2808M
[08/15 12:48:43 d2.utils.events]: eta: 3 days, 6:11:42 iter: 139 total_loss: 0.2773 loss_cls: 0.05356 loss_box_reg: 0.1433 loss_rpn_cls: 0.06134 loss_rpn_loc: 0.01677 time: 1.5637 data_time: 0.0
0.013986 max_mem: 2809M
[08/15 12:49:17 d2.utils.events]: eta: 3 days, 7:10:01 iter: 159 total_loss: 0.2487 loss_cls: 0.03802 loss_box_reg: 0.1464 loss_rpn_cls: 0.04818 loss_rpn_loc: 0.01264 time: 1.5775 data_time: 0.0
0.015984 max_mem: 2809M
[08/15 12:49:49 d2.utils.events]: eta: 3 days, 7:30:21 iter: 179 total_loss: 0.3101 loss_cls: 0.04005 loss_box_reg: 0.1649 loss_rpn_cls: 0.06056 loss_rpn_loc: 0.02755 time: 1.5828 data_time: 0.0
0.017982 max_mem: 2809M
[08/15 12:50:20 d2.utils.events]: eta: 3 days, 7:29:49 iter: 199 total_loss: 0.2631 loss_cls: 0.03587 loss_box_reg: 0.1571 loss_rpn_cls: 0.03963 loss_rpn_loc: 0.01938 time: 1.5811 data_time: 0.0
0.01998 max_mem: 2809M
[08/15 12:50:52 d2.utils.events]: eta: 3 days, 7:29:17 iter: 219 total_loss: 0.2970 loss_cls: 0.048 loss_box_reg: 0.1804 loss_rpn_cls: 0.03434 loss_rpn_loc: 0.02105 time: 1.5015 data_time: 0.06
0.021978 max_mem: 2809M
[08/15 12:51:24 d2.utils.events]: eta: 3 days, 7:28:46 iter: 239 total_loss: 0.3131 loss_cls: 0.04699 loss_box_reg: 0.214 loss_rpn_cls: 0.02754 loss_rpn_loc: 0.02166 time: 1.5832 data_time: 0.06
```

Figure 5: Epoch Training Run Sample

## 6 Evaluation Metrics

The evaluation of the relevant model run is triggered through the command ‘python3 evaluation.py –dataset=’/path to dataset’ –weights=’/path to pretrained weight.h5’’. Relevant output metrics are displayed automatically in separate windows, see figure 6. A

confusion matrix of the predictions and ground truth together with the precision-recall regression is available.

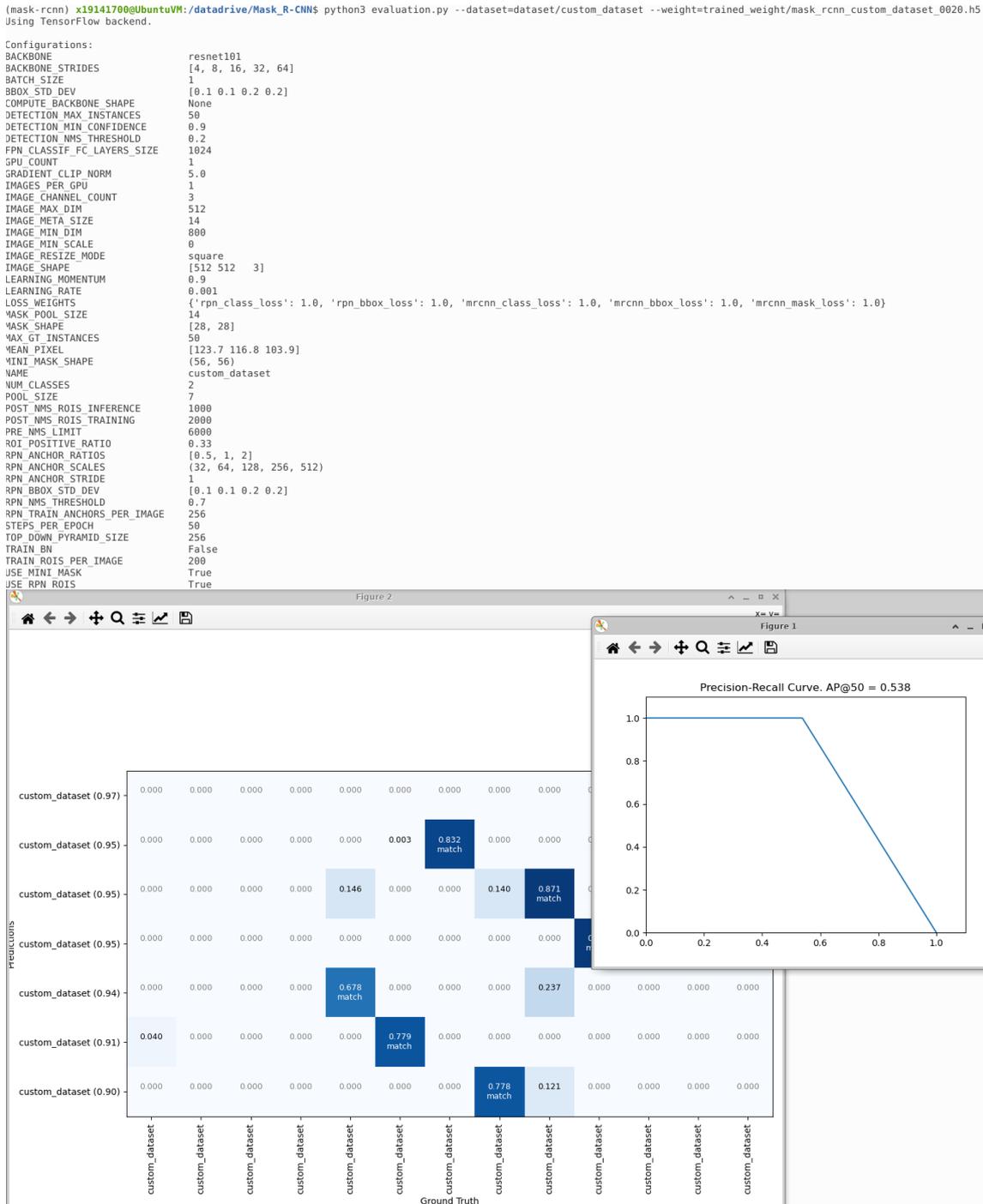


Figure 6: Mask R-CNN Evaluation code screenshot

Further evaluation matrix can be also retrieved from tensorboard with the command 'tensorboard --logdir=logs/'path to trained dir''. Tensorboard can be viewed under port 6006, graphics can also be exported from there. Here metrics such as loss function in general and specific for mask and box loss is available in graph format as well as the overall progression of the mean average precision.

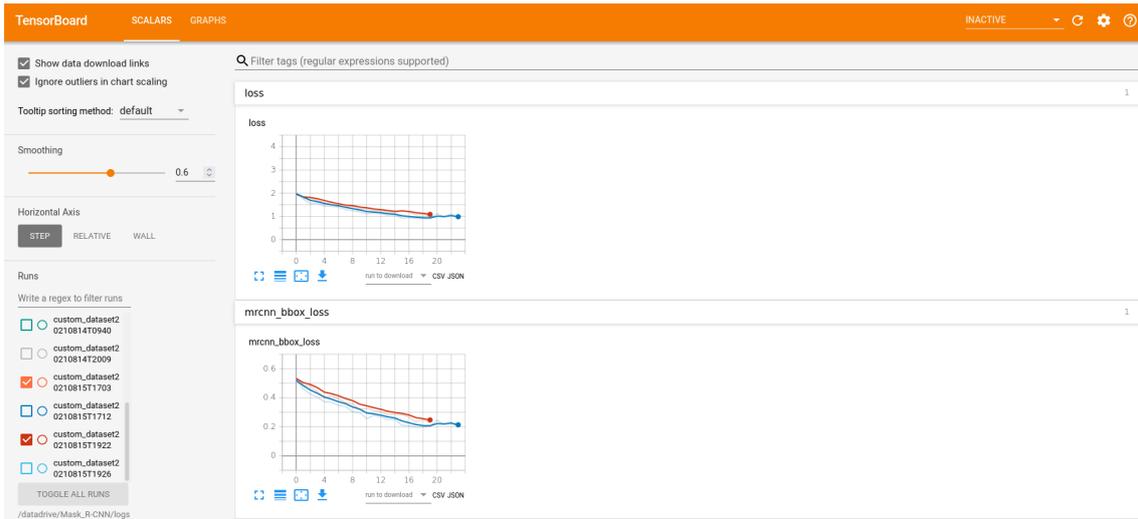


Figure 7: Tensorboard Screenshot with loss graph