National College of Ireland

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

National College of Ireland

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).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
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.	☐

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 website[1].
- **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 Github[2].

[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 website[3].

- **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.

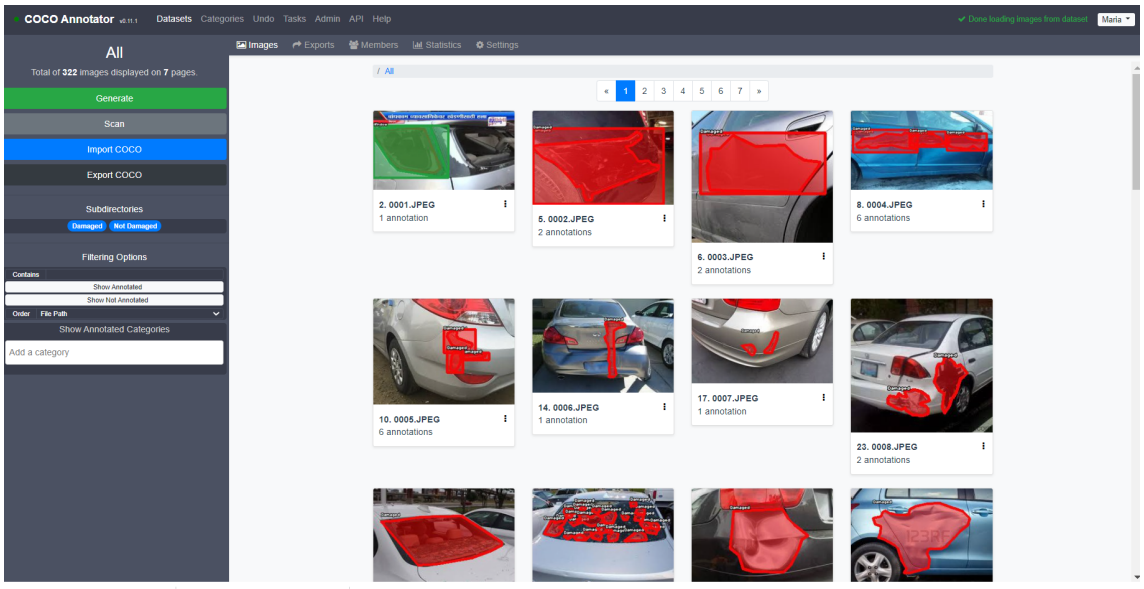

Figure 1: Coco Annotator Dataset Overview

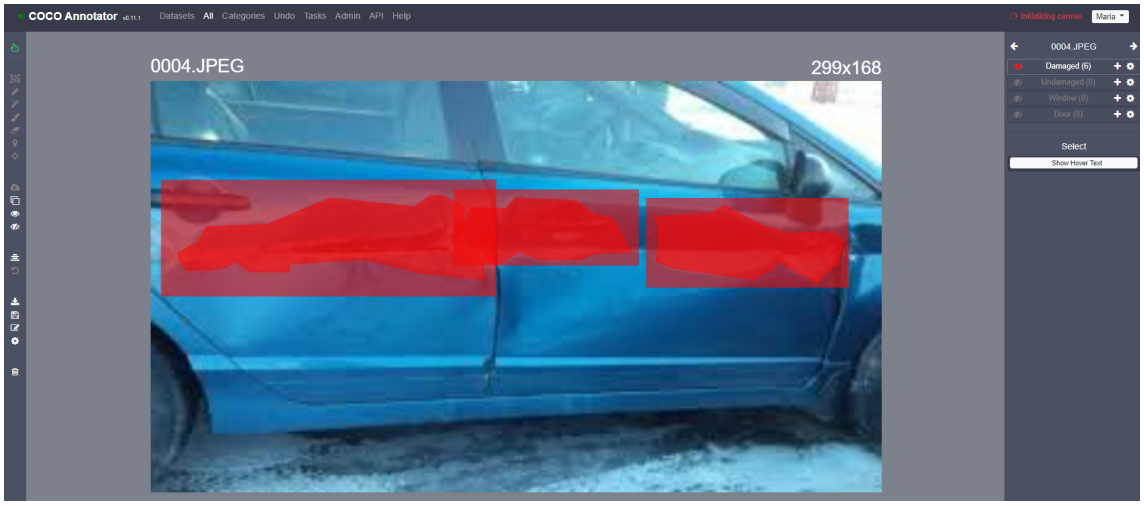

Figure 2: Coco Annotator Labelling tool

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_ROIS_INFERENCE        1000
```

Figure 3: Mask RCNN Training initiation sample

5.2 Model Implementation enhanced Mask R-CNN with 'Unbiased 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

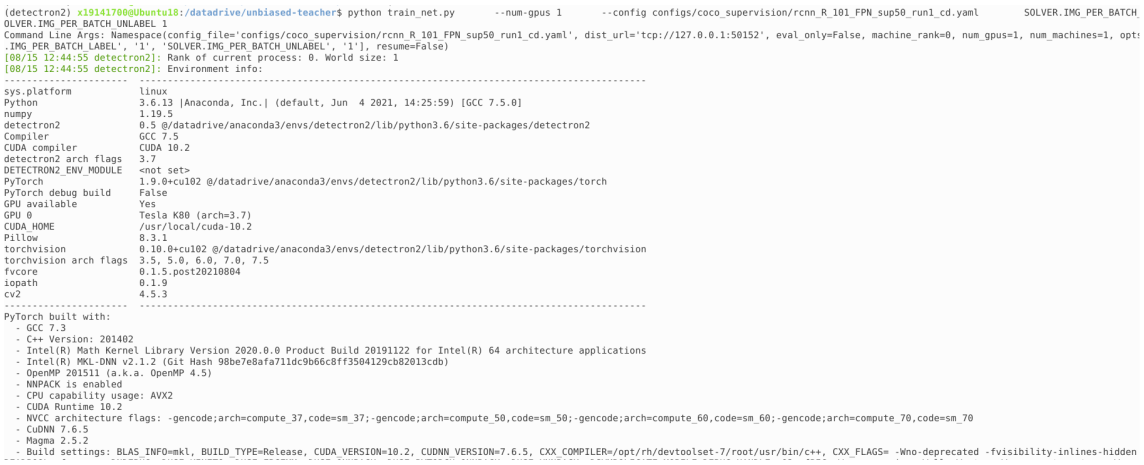

Figure 4: enhanced Mask RCNN Training initiation sample

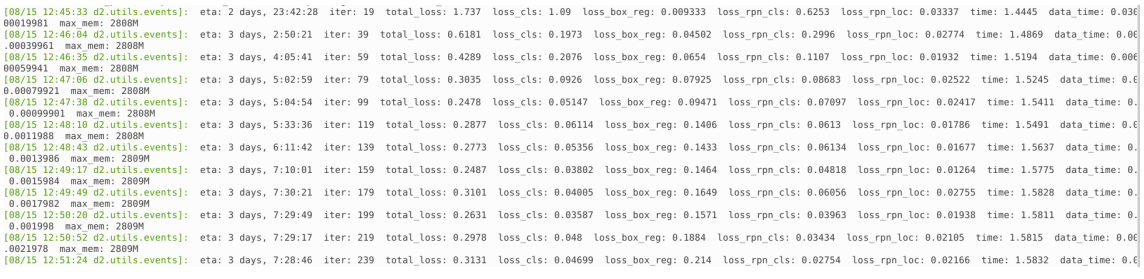

Figure 5: Epoch Training Run Sample

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.

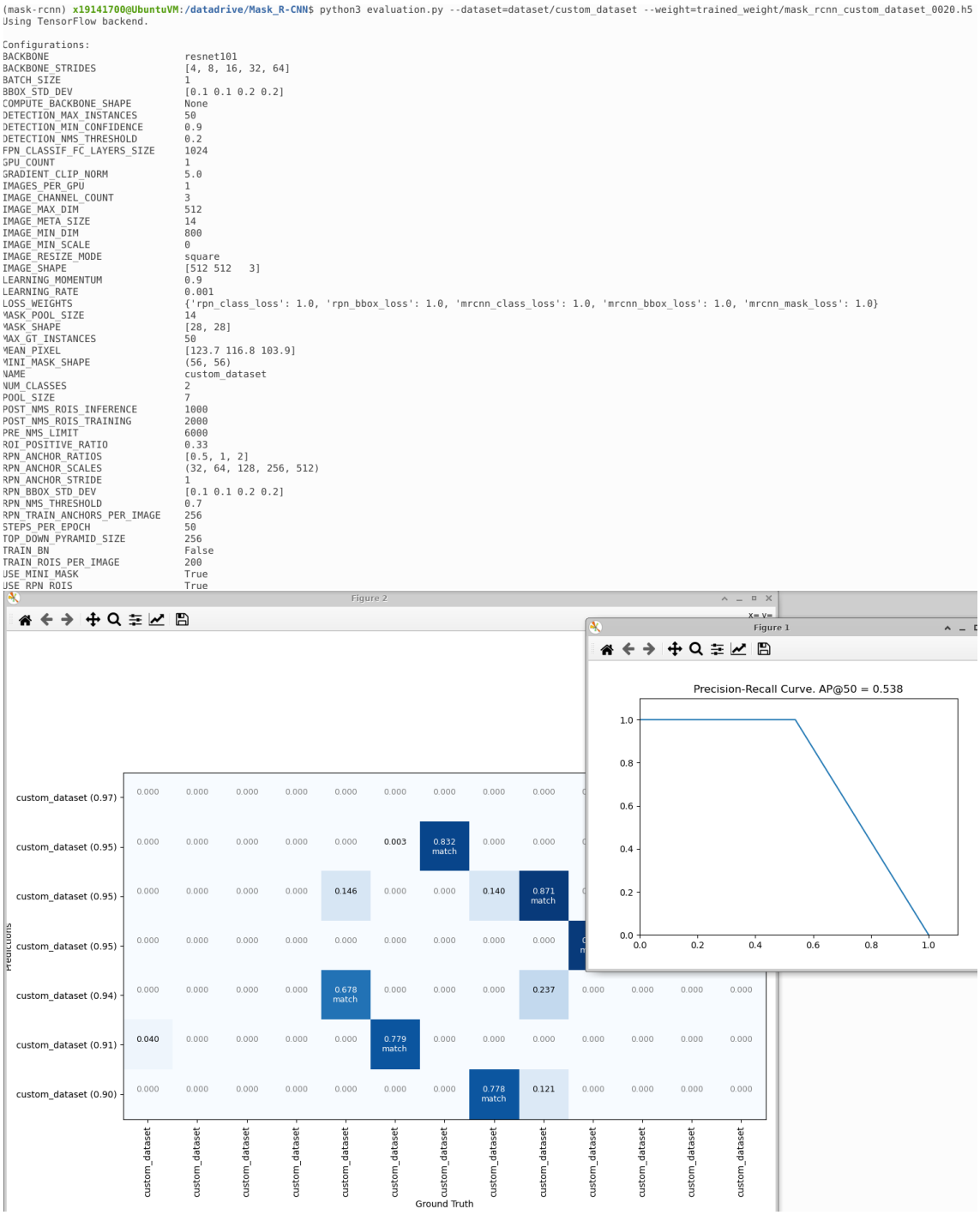

Figure 6: Mask R-CNN Evaluation code screenshot

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.

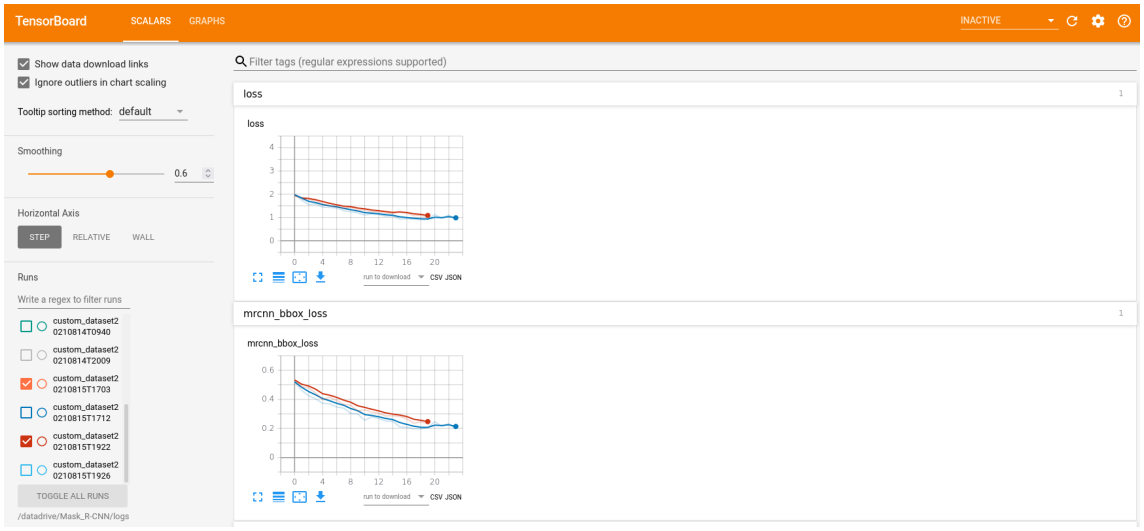

Figure 7: Tensorboard Screenshot with loss graph