## IT107 Assignment on Object Detection

Student ID: 23A054Q  
Student Name: Allen Lee

### Objects for Detection

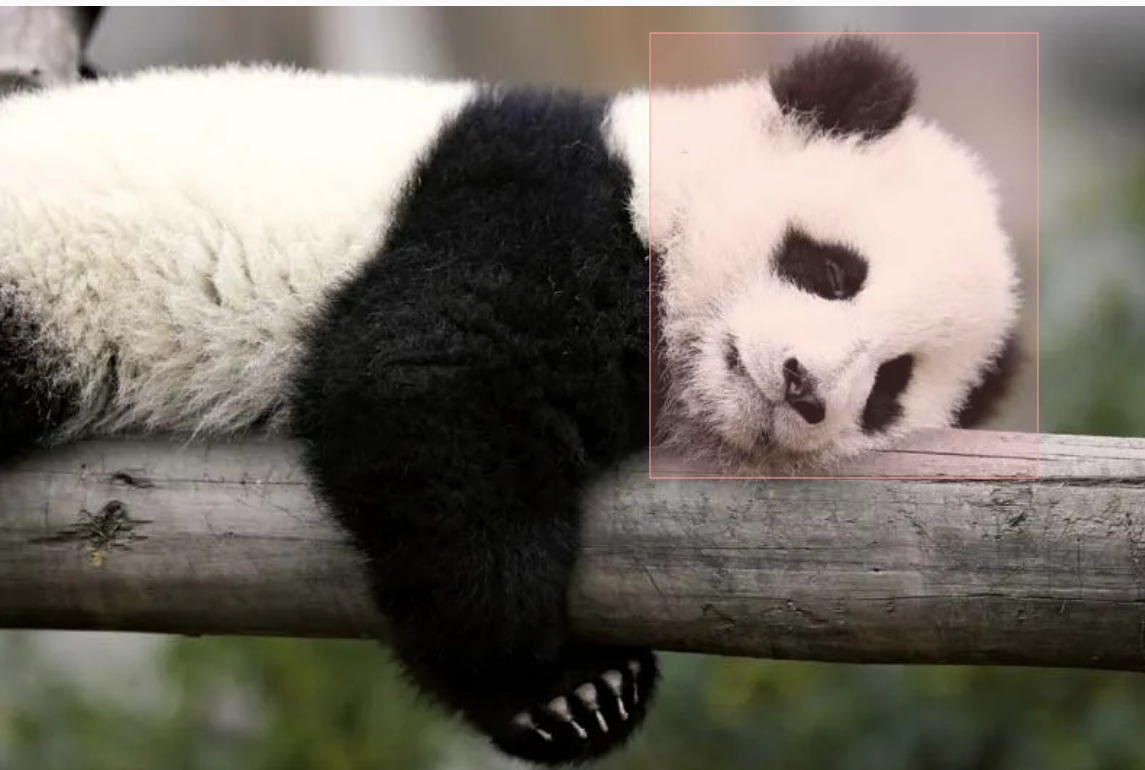
* Giant Panda
* Red Panda

### Data Collection and Annotations

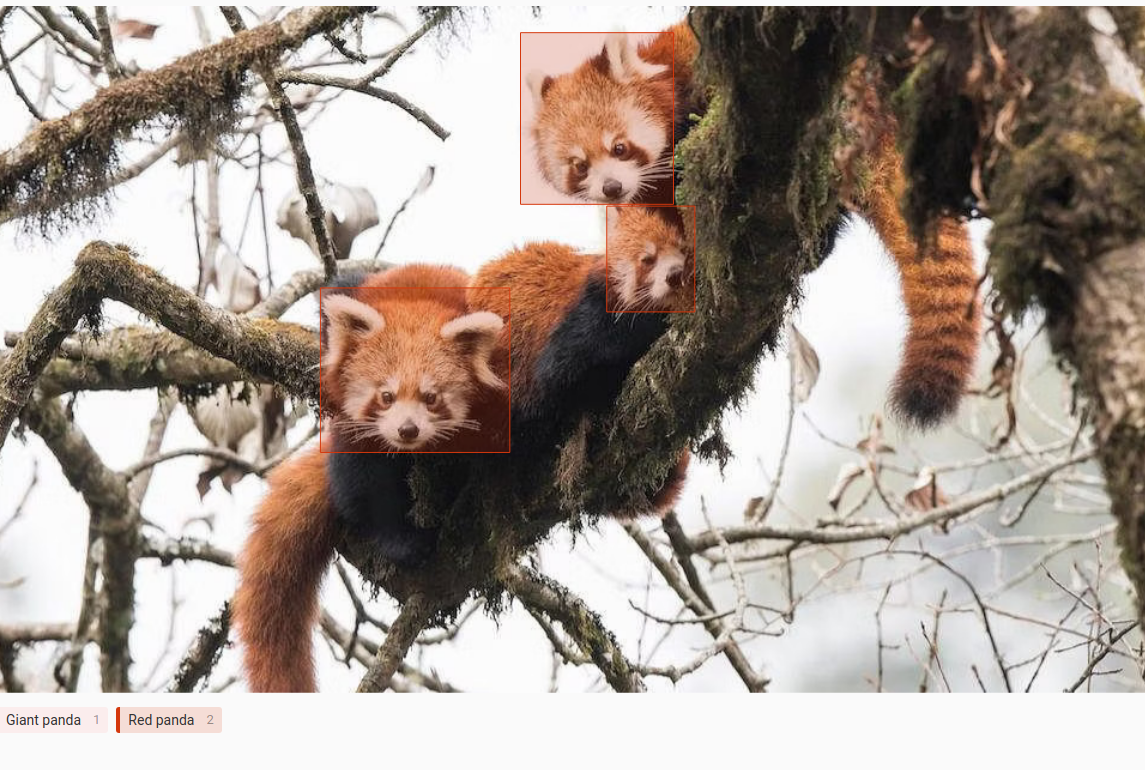
Photo images of giant pandas and red pandas are extracted from the web from Google images and some zoo web sites. The images are all converted into JPEG format using the open-source image processing tool GIMP, if they are not already in JPEG format. The annotations of these images are then carried out using Label Studio and exported as the Pascal VOC xml format.

The images, annotations and the source code files for the models are stored on GitHub ([https://github.com/sokonana/nyp-it107-assignment/](https://github.com/sokonana/nyp-it107-assignment)) for easy reference and maintenance.

#### Sample annotation image of Giant Panda



#### Sample annotation image for Red Pandas



### Pretrained Model Selection and Training

After all the images are labelled, the Python program from our class practical exercises is used to run and generate the images into training and validation sets in TFRecord format. The following 2 new files are generated after running the Python program ‘create\_tf\_voc.py’ via a Bash script.

* ‘train.record’ (number of records:108)
* ‘val.record’ (number of records: 26)

The program shuffled all the images and their accompanying annotation XML files and split them into training and validation data sets in 80/20 ratio.

A pretrained model for image detection (MobileNet V2 320x320) was selected and downloaded from ‘Tensorflow 2 Detection Model Zoo’ site for initial assessment.

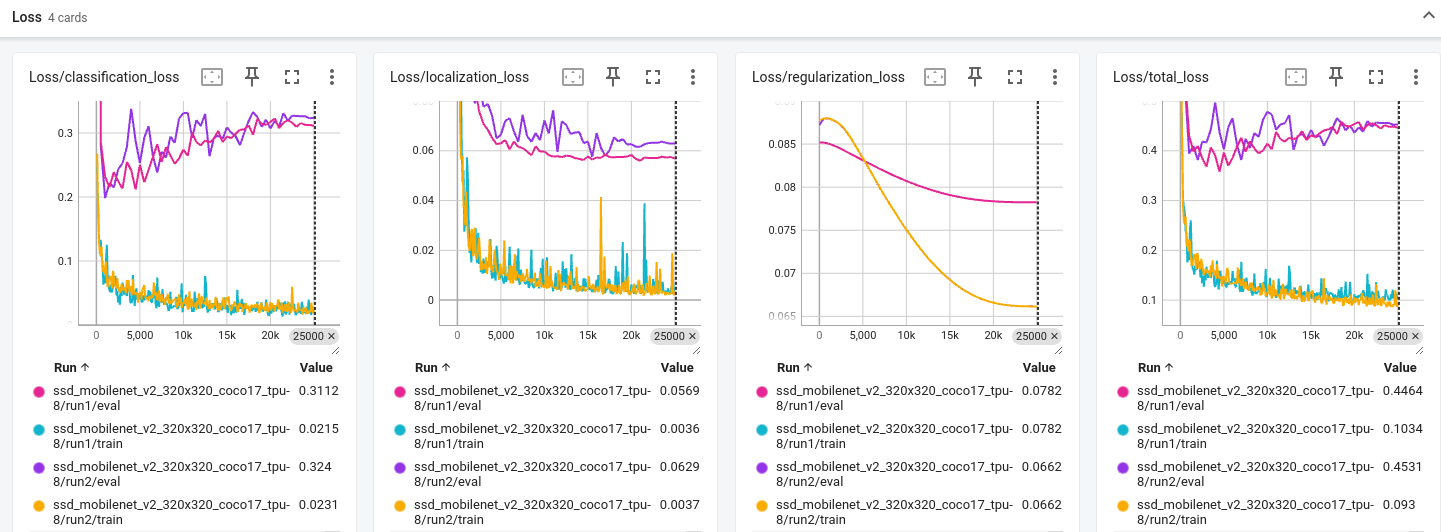
Link -> <http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_320x320_coco17_tpu-8.tar.gz>

The ‘model\_main\_tf2.py’ file was modified to keep the last 20 checkpoints. Evaluation on validation set is done every 500 steps, while training set loss calculation is done every 100 steps.

### Configurations (pipeline.config)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Run ID | Train Batch Size | Learn Rate (warm-up / base) | NMS Score | Anchor Box Aspect Ratios | Dropout Rate |
| run1 | 32 | 0.002/0.01 | 9.9e-9 | 1.0/2.0/0.5/3.0/0.33 | Not used |
| run2 | 48 | 0.002/0.04 | 9.9e-9 | 1.0/1.5/0.6667/2.0/0.5 | 0.25 |
| run3 | 32 | 0.0005/0.005 | 0.02 | 1.0/1.5/0.667/2.0/0.5/3.0/0.333 | 0.40 |
| run4 | 32 | 0.002/0.02 | 9.9e-9 | 1.0/2.0/0.5/3.0/0.33/4.0/0.25 | 0.30 |

Initial run results show evidence of overfitting from the divergence between training and evaluation classification loss after 25000 training steps. This is seen in both run1 & run2 below. Localisation loss chart does not indicate such divergence. The lower learning rate used for run1 shows a more gradual and stable loss convergence.



However, the average mAP for run1 and run2 does not show significant difference in performance.

Run3 was set up to attempt to reduce overfitting via higher dropout rate and tweaking on non-maximum suppression settings. A lower learning rate for warm-up and base is also set for run3. The performance for run3 is significantly lower than run1 and run2. The training was stopped after 20,000 steps.

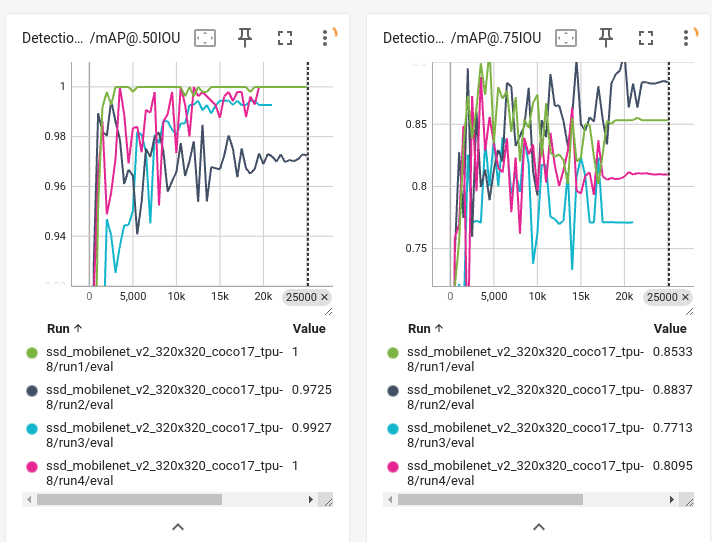
Run4 added 4.0 & 0.24 aspect ratio for more anchor box generation and more image augmentation methods are added to generate more varied training samples. The dropout rate and learning rate is set to a moderate level between run2 and run3. The batch normalisation epsilon and decay rate are also tweaked slightly.

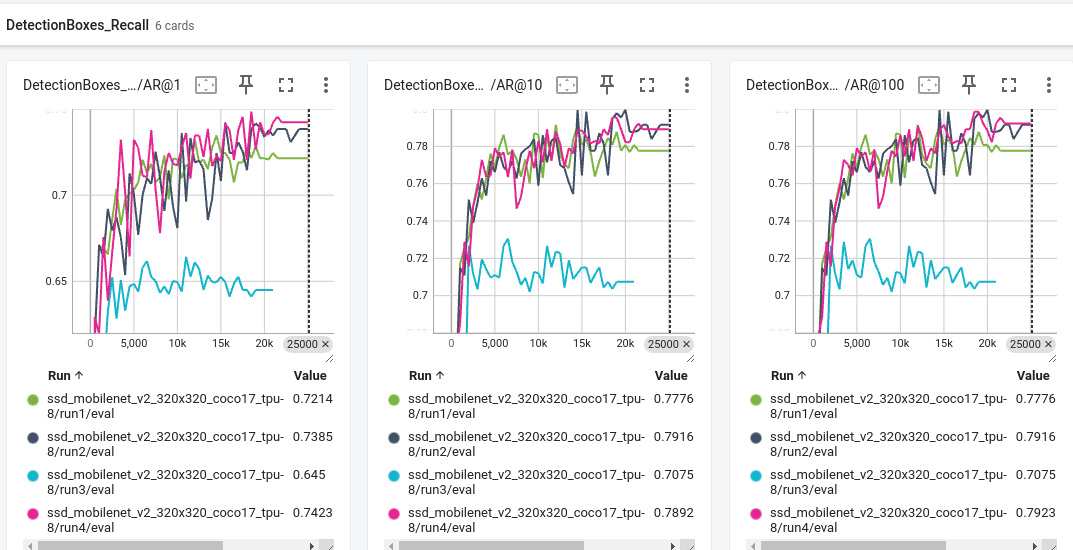
### Selection and Export of Saved Model

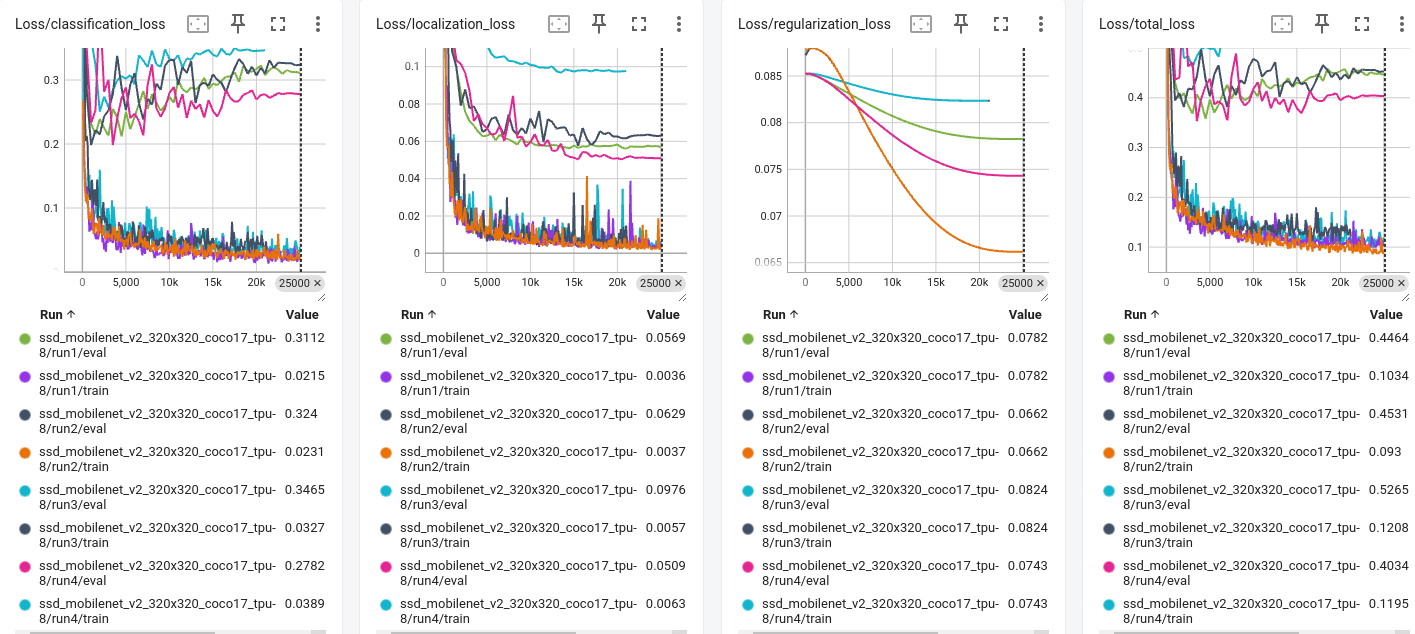
Run4 gives the best performance in mAP and detection Recall metrics, as well as the lowest in total loss, though training took the longest time among all runs. This model with checkpoint corresponding to the highest mAP score is then exported as a trained model using the checkpoint parameter after 21,500 steps (ckpt-44).



Detection mAP @0.5 for run1 & run4 are at 1.0 after 20k steps.







### Testing on Saved Model

Testing is conducted using a few test images and videos to assess the capability of the model on new data using a Jupyter Notebook ‘detect\_panda.ipynb’. On the videos, the detection of Red Panda is not as good as that for the Giant Panda. Even though the model has successfully detected the location, there are some misclassifications of the Red Panda for Giant Panda in some frames.

This might be due to the more variation in the hues of the Red Panda compared to the Giant Panda under different lighting conditions.

### Deployment of Object Detection Model

Deployment of the saved model from run4 is done on HuggingFace space under 23A054Q. The model is uploaded to <https://huggingface.co/sokonana/it107model>

A template Python application using Gradio is used and hosted to run inference on the saved model. Source code for the Python program can be found at <https://huggingface.co/spaces/ITI107-2023S2/23A054Q/tree/main>

Url to the space is as below.

<https://huggingface.co/spaces/ITI107-2023S2/23A054Q>

