ITI107 – Assignment

Adelene Ng

1 Introduction

In this project, we develop a solution to detect and classify which group of mammals an image belongs to. In particular, we want to identify **lions, tigers and cheetahs** in an image or video using the Tensorflow 2 Object Detection API.

2 Dataset

The images are collected by doing an image Google search (Figure 1).

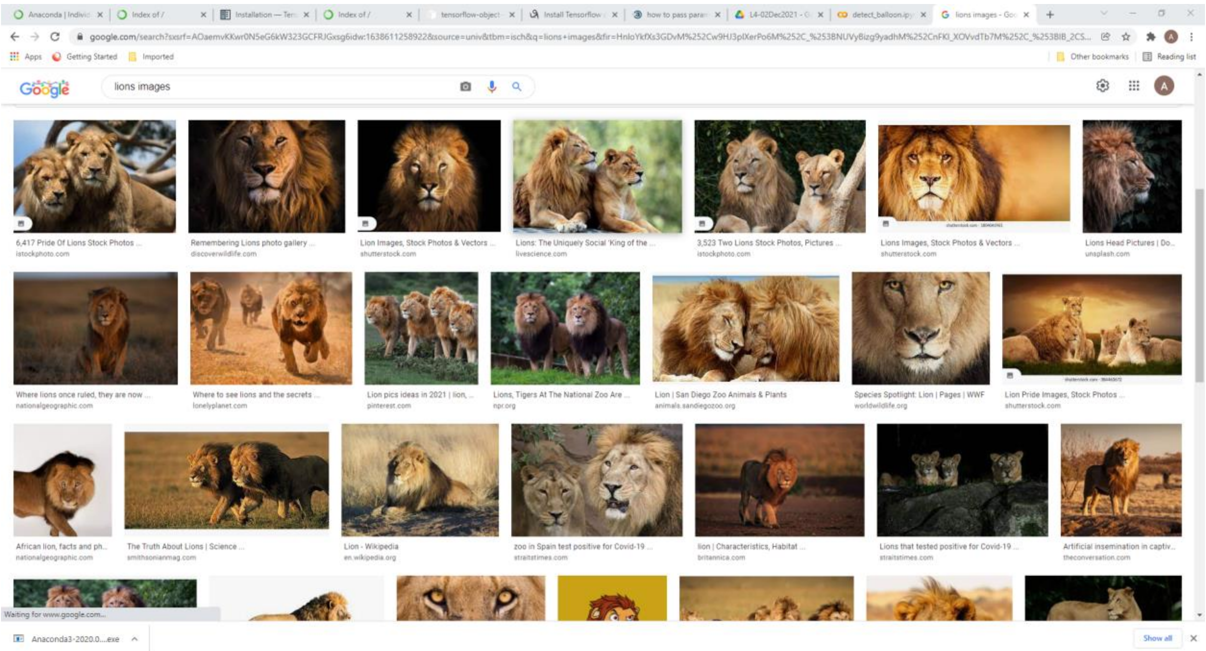


Figure 1 – Google Image Search for "Lions"

To download the images returned by Google, I installed a Chrome Extension, [Fatkun Batch Download Image](), from the Chrome Web Store (Figure 2).

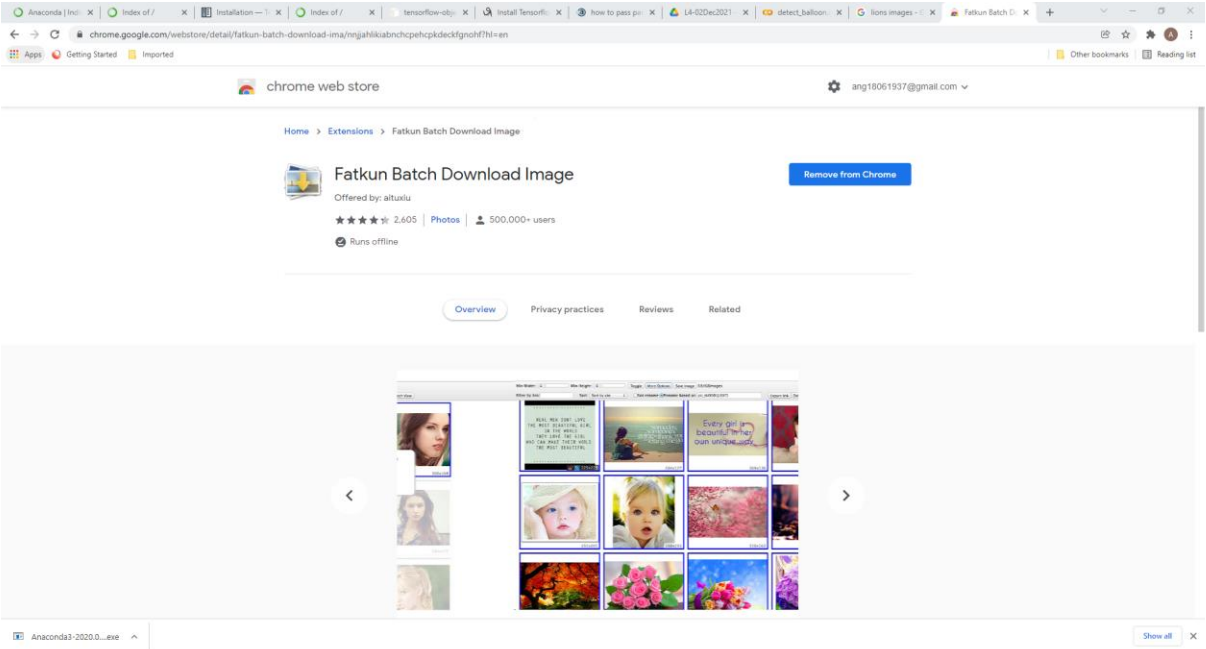


Figure 2 – Chrome Web Store – "Fatkun" Batch Download Image Extension Software

When installed, the "Fatkun" Batch Download Image can be invoked by clicking on the icon shown in (Figure 3) (red rectangle):

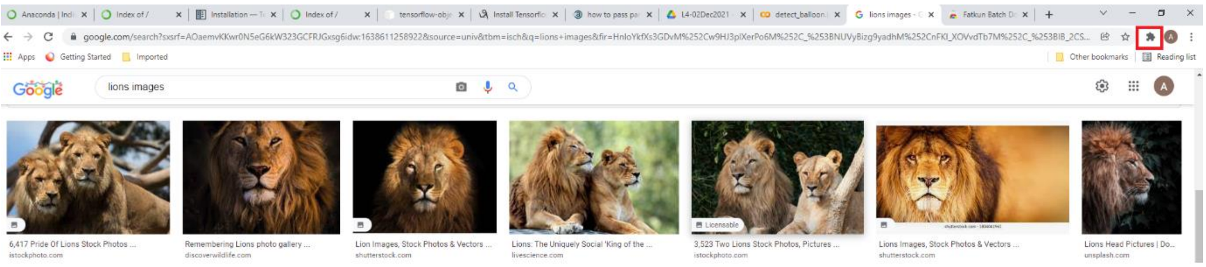


Figure 3 – "Fatkun" Batch Download Image

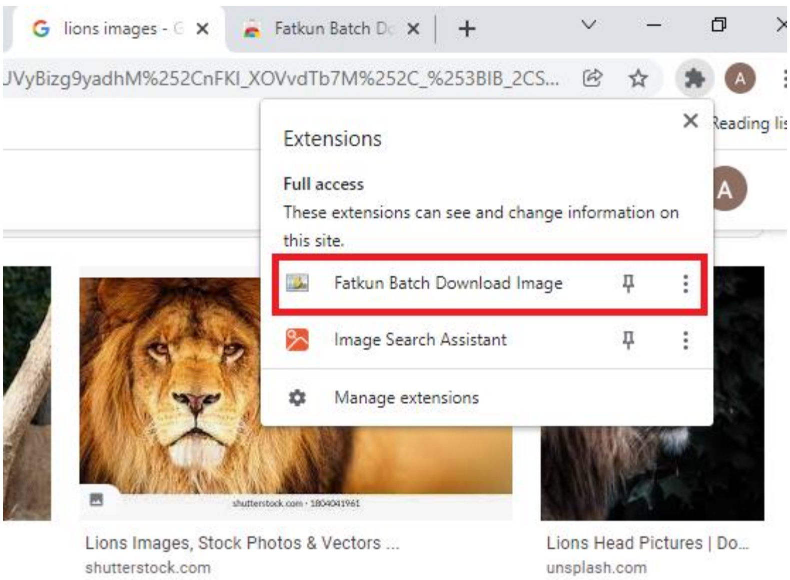


Figure 4 – Select "Fatkun" Batch Download Image (highlighted in RED)

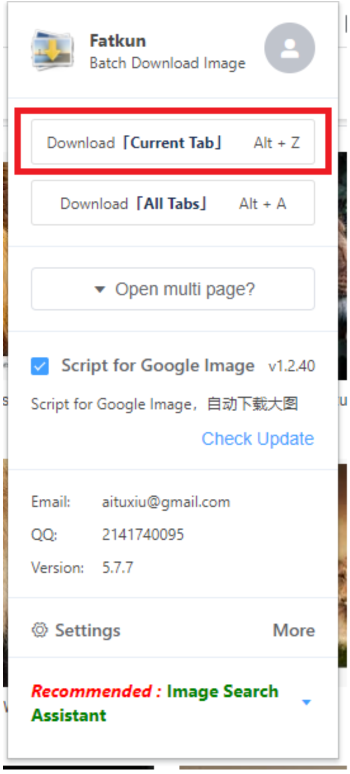


Figure 5 – Select "Download (Current Tab)"

Next, select the "Fatkun" Batch Download Image option from the drop down (Figure 4) and click "Download (Current Tab)" Figure 5

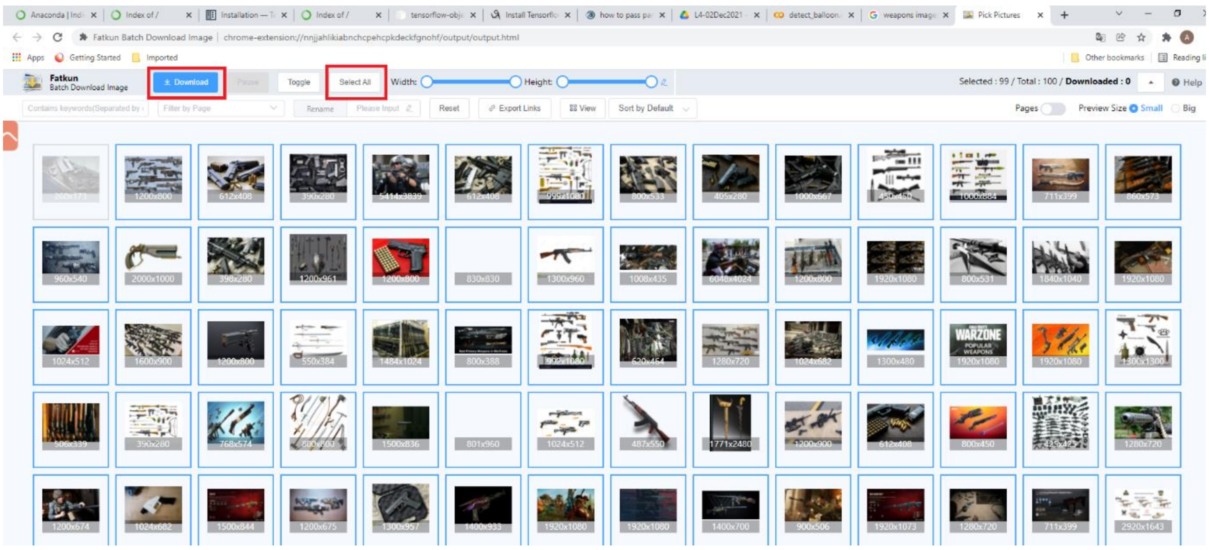


Figure 6 – Click "Select All" followed by "Download" to retrieve the images for training

Click "Select All" followed by pressing the "Download" button shown in Figure 6.

Once the 3 sets of image classes (lions, tigers and cheetahs) have been downloaded, these will have to be labelled. I use [LabelImg]() for this purpose. An example of the labeling is shown in Figure 7.

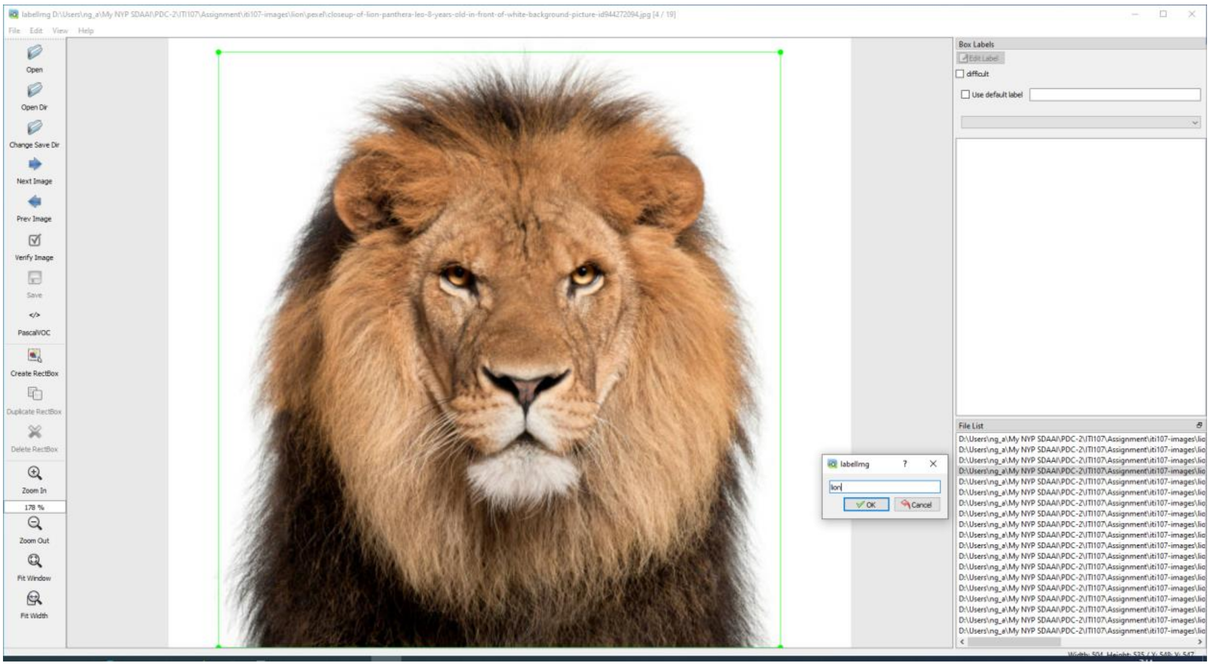


Figure 7 – Selecting an image for labelling

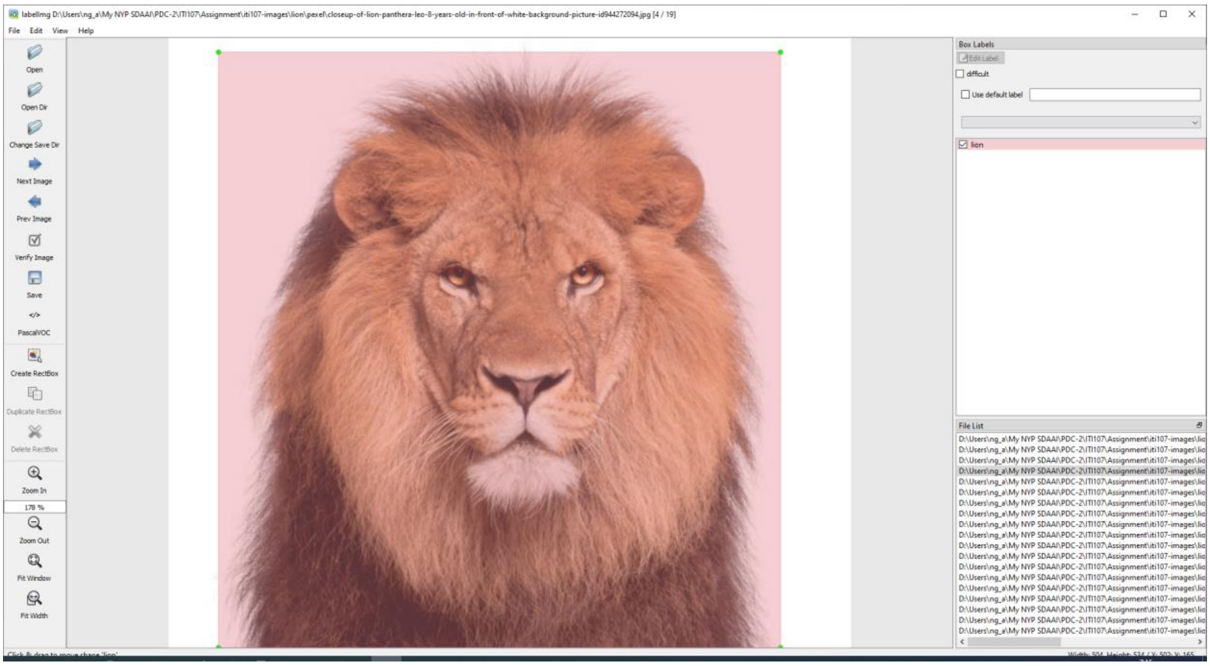


Figure 8 – Label Finalization

Table 1 shows a breakdown of the images collected from the Internet:

Image Class	Total
lions	265
tigers	201
cheetahs	328
TOTAL	794

Table 1 – Image Class Breakdown

3 Model Training

For this project, I am using the "EfficientDet D0 512x512" pre-trained model. The reason for choosing this is because I want to incorporate real time object detection. It is fast however; it will not be as accurate. I chose the "D0" family because of training speed as well over the other series "D1", …, "D7".

3.1 Baseline Parameters (Run 2)

For the baseline run, Table 2 shows the parameters that I have used for the initial baseline training.

In the baseline run, the `number of classes` is changed to **3**. We want to be able to detect: lions, tigers and cheetahs. The '`batch size`' was changed to **4** to fit the amount of GPU memory I had on my laptop. I also reduced the '`num_steps`' to **100000** and changed the '`fine_tune_checkpoint_type`' from `classification` to `detection`. This is because the images supplied usually have multiple objects (of different classes), and object detection allows us to identify all the objects and the positions of each of them. `use_bfloat16` was changed from `true` to `false` as I do not have a TPU.

I also had to change the directory to point to the location where `label_map.pbtxt`, `train.record`, `val.record`, `ckpt-0` was stored.

Hyperparameter Name	Old Value	New Value
num_classes	90	3
batch_size	128	4
fine_tune_checkpoint		D:/Users/ng_a/TensorFlow/workspace/training_custom/pre-trained-models/efficientdet_d0_coco17_tpu-32/checkpoint/ckpt-0
num_steps	300000	100000
fine_tune_checkpoint_type	classification	detection
use_bfloat16	True	false
train_input_reader/label_map_path		D:/Users/ng_a/TensorFlow/workspace/training_custom/data/label_map.pbtxt
train_input_reader/input_path		D:/Users/ng_a/TensorFlow/workspace/training_custom/data/train.record
eval_input_reader/label_map_path		D:/Users/ng_a/TensorFlow/workspace/training_custom/data/label_map.pbtxt
eval_input_reader/input_path		D:/Users/ng_a/TensorFlow/workspace/training_custom/data/val.record

Table 2 – Baseline Hyperparameters

3.2 Experimental Run 3

Hyperparameter Name	Old Value	New Value

	`loss {`
	`localization_loss {`
	`weighted_smooth_l1 {`
	`}`
	`}`
	`classification_loss {`
	`weighted_sigmoid_focal`
	`{`
		`loss {`
		`localization_loss {`
		`weighted_smooth_l1 {`
		`}`
		`}`
		`classification_loss {`
		`bootstrapped_sigmoid {`
		`alpha: 0.25`

	gamma: 1.5 alpha: 0.25 } } classification_weight: 1.0 localization_weight: 1.0 }	} } classification_weight: 1.0 localization_weight: 1.0 }

Table 3 – Hyper-parameters for experimental run3

In this experiment, I changed the `classification_loss` to `bootstrapped_sigmoid`, (Table 3) so that as the model improves over time, its predictions can be trusted more and I can use these predictions to mitigate the damage of noisy/incorrect labels.

3.3 Experimental Run 4

Hyperparameter Name	Old Value	New Value

	anchor_generator {
	
	multiscale_anchor_generator {
	min_level: 3
	max_level: 7
	anchor_scale: 4.0
	aspect_ratios: 1.0
	aspect_ratios: 2.0
	aspect_ratios: 0.5
	scales_per_octave: 3
	}
	}
		anchor_generator {
		
		multiscale_anchor_generator {
		min_level: 3
		max_level: 7
		anchor_scale: 4.0
		aspect_ratios: 1.0
		aspect_ratios: 2.0
		aspect_ratios: 0.25
		aspect_ratios: 0.75
		scales_per_octave: 3
		}
		}

	data_augmentation_options {
	random_horizontal_flip {
	}
	}
	data_augmentation_options {
	
	random_scale_crop_and_pad_to_square {
	output_size: 512
	scale_min:
	0.10000000149011612
	scale_max: 2.0
	}
		data_augmentation_options {
		
		random_scale_crop_and_pad_to_square {
		output_size: 512
		scale_min: 0.5
		scale_max: 2.0
		}
		}

Table 4 – Hyper-parameters for experimental run 4

Here, I change anchors to best fit shapes of objects in my custom dataset. I added `aspect_ratios`, **0.25** and **0.75**. The former translates to 4 high whereas the latter translates to 1.33 high. I also removed `random_horizontal_flip` as images taken are usually not mirrored around the horizon. The `scale_min` attribute within `random_scale_crop_pad_to_square` was changed to **0.5** (Table 4).

3.4 Experimental Run 5

Hyperparameter Name	Old Value	New Value

	train_config {
	…
	
	…
	}
		train_config {
		…
		data_augmentation_options {
		normalize_image {
		original_minval: 0.0

<table>
<tr><th>Hyperparameter Name</th><th>Old Value</th><th>New Value</th></tr>
<tr><td></td><td></td><td><pre>   original_maxval: 255.0
        target_minval: -1.0
        target_maxval: 1.0
      }
    }
  …
}</pre></td></tr>
</table>

Table 5 – Hyper-parameters for experimental run5

In this experimental run, I try image augmentation, in particular, image normalization. I map the original values (0.0, 255.0) to (-1.0, 1.0). See Table 5

3.5 Experimental Run 6

<table>
<tr><th>Hyperparameter Name</th><th>Old Value</th><th>New Value</th></tr>
<tr><td></td><td><pre>post_processing {
      batch_non_max_suppression
{
        score_threshold:
9.99999993922529e-09
        iou_threshold: 0.5

max_detections_per_class: 100
        max_total_detections:
100
      }
      score_converter: SIGMOID
    }</pre></td><td><pre>post_processing {

batch_non_max_suppression {
        score_threshold: 0.2
        iou_threshold: 0.5

max_detections_per_class: 30
        max_total_detections:
90
      }
      score_converter:
SIGMOID
    }</pre></td></tr>
<tr><td></td><td><pre>train_config {
  …
  max_number_of_boxes: 100
  …
}</pre></td><td><pre>train_config {
  …
  max_number_of_boxes: 90
  …
}</pre></td></tr>
</table>

Table 6 – Hyper-parameters for experimental run6

In this run, I change the `batch_non_max_suppression` parameters.

`score_threshold`: defines a minimum confidence score for classification, which should be reached so the proposal won't be filtered out. The default value is a value close to 0, i.e. all proposals are accepted. I set the value to **0.2** because eliminating those proposals that are most likely to be incorrect leads to more stable training.

`max_detections_per_class`: I set the value to the maximum number of objects for a single class (10) times the number on anchors (number of aspect_ratios) (3): **30**

`max_total_detections` was set to `max_detections_per_class x total number of classes`: 30 x 3 = **90**

`max_number_of_boxes` is set to the same number as `max_total_detections`: **90**. See Table 6

4 Results

Experimental Run	mAP
Baseline	0.856
Run3	0.791

Experimental Run	mAP
Run4	0.847
Run5	0.001
Run6	0.727

Table 7 – mAP for the different experimental runs

Table 7 summarizes the mAP values for the different experiments that were carried out. The baseline run show the best results.

Sections 4.1 - 4.5 analyses the results from the different experimental runs. Each show the mAP, loss as well as the evaluation images.

4.1 Baseline Run 2

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.573
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.856
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.646
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.165
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.586
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.457
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.650
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.693
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.361
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.703
INFO:tensorflow:Eval metrics at step 29300
```

Figure 9 – Evaluation results from baseline experiment

The mAP results from this run are shown in Figure 9. This shows the best results from the 5 experimental runs.

One possible reason why not much tuning is really needed is because the objects to be detected are quite similar to the domain of the original dataset and is not from a completely different domain for example, X-ray images.

Also, the size of the target dataset is not very big (about 800 images). The network probably will not learn much from training more layers, so it will tend to overfit the new data. So, less fine-tuning is needed.

The loss is shown in Figure 10 and evaluation images are shown in Table 8.

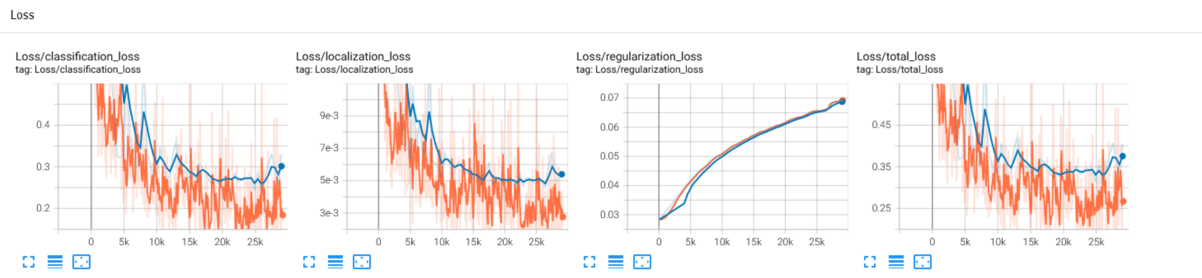
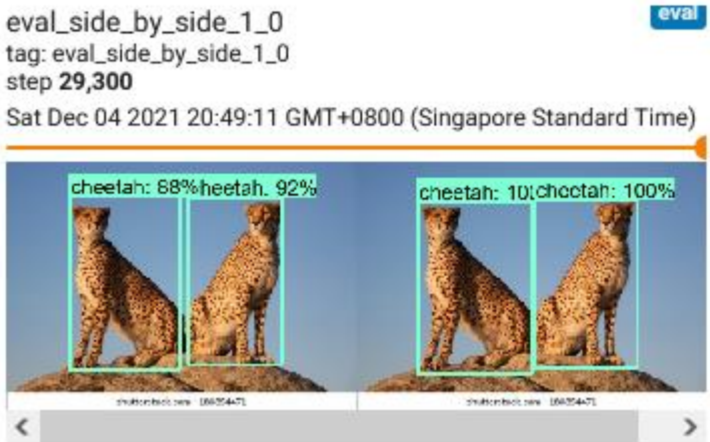
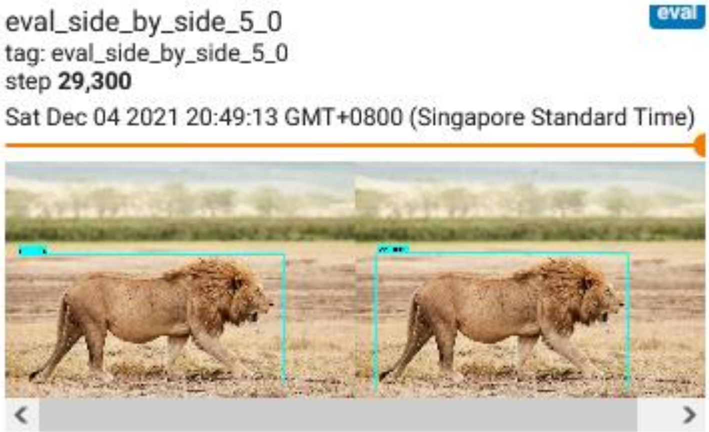


Figure 10 – Loss for baseline experiment

Evaluation #	Evaluation Images
0	
1	
2	

Evaluation #	Evaluation Images
3	
4	
5	

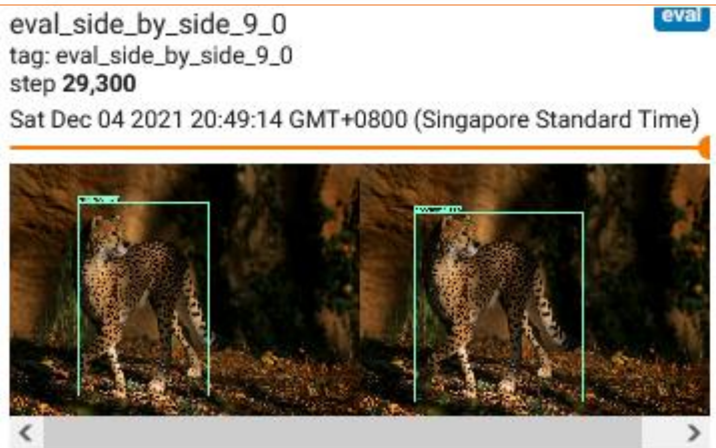
Evaluation #	Evaluation Images
6	
7	
8	
9	

Table 8 – Evaluation Images from baseline run

Since this experimental run shows the best mAP value from the 5 runs, I used the model from this run against a test image. Figure 11 show the result of running the detector on the test image. All objects (lion, cheetah and tiger) that we want this model to detect has been successfully detected. The scores are relatively good too: lion shows a probability of 0.89, cheetah 0.70 and tiger 0.91 respectively.

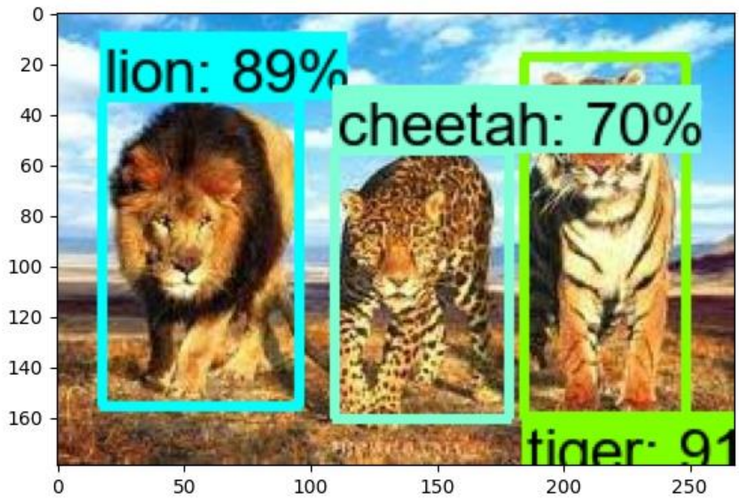


Figure 11 - Results of running this model against a test image

This model does not do too well on real time object detection as shown in the attached video. Lions were detected with scores between 50% - 80% but it also makes false detections in certain frames too (bottom of frame with no lion yet model says it is a lion @ 7s – 10s into video). Cheetah detection was not so good (about 50% - 80%). It was missed completely in certain frames (@ 43s, @52s – 56s). YOLO v4 might be a better pre-trained model to use in this case.

4.2 Run 3

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.461
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.791
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.490
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.210
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.472
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.407
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.588
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.641
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.400
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.652
INFO:tensorflow:Eval metrics at step 38800
```

Figure 12 - Evaluation results from experiment run 3

In this run, I changed the loss function to `bootstrapped_sigmoid`.

The mAP results from this run are shown in Figure 12.

Figure 13 shows the loss and Table 9 shows the evaluation images from experimental run 3.

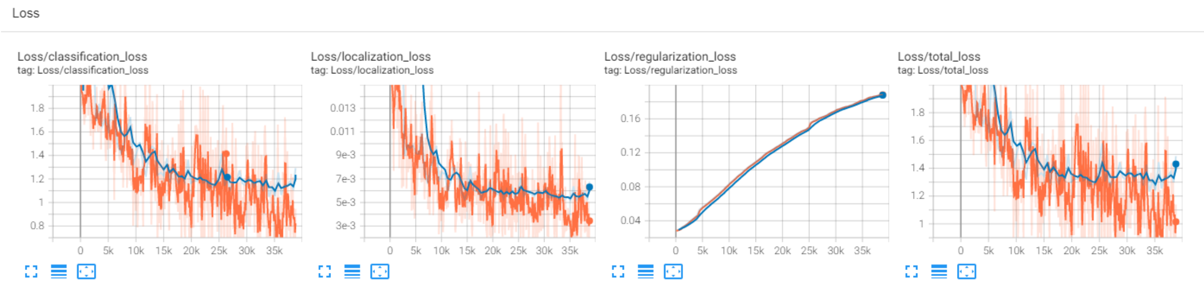
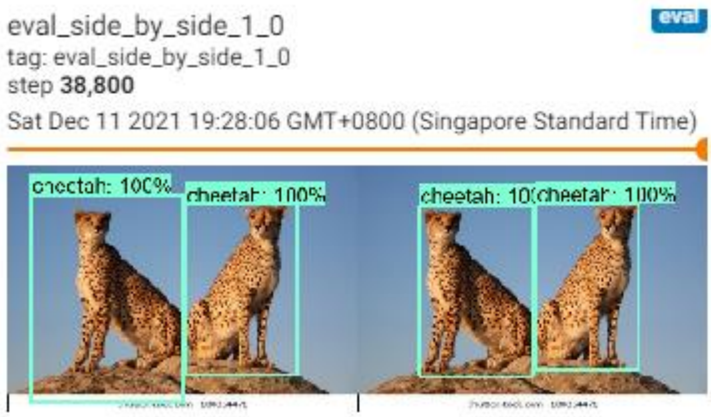
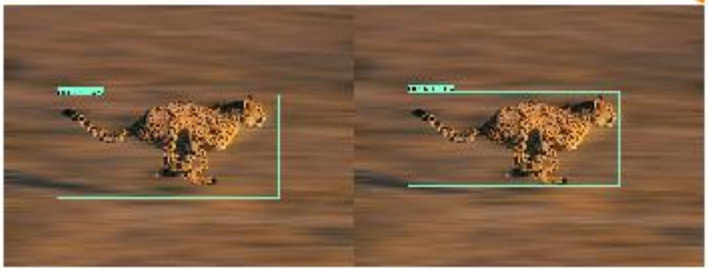
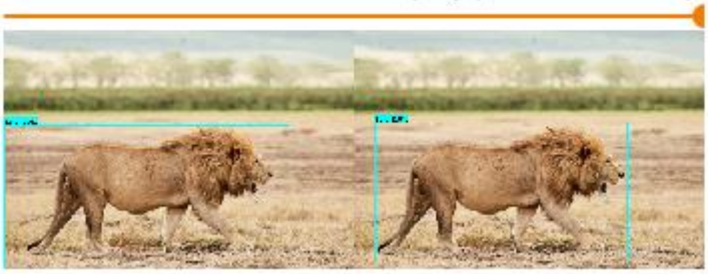
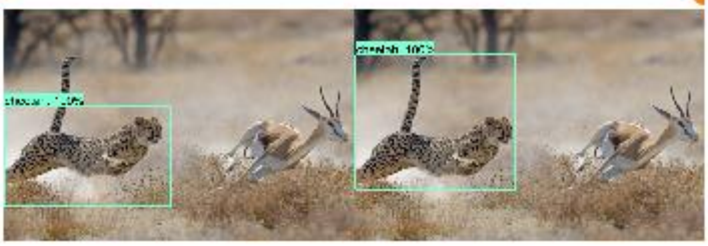


Figure 13 – Loss from experimental run 3

<table>
<tr><th>Evaluation #</th><th>Evaluation Images</th></tr>
<tr><td>0</td><td>
</td></tr>
<tr><td>1</td><td>
</td></tr>
<tr><td>2</td><td>
</td></tr>
</table>

Evaluation #	Evaluation Images
3	eval_side_by_side_3_0 eval tag: eval_side_by_side_3_0 step **38,800** Sat Dec 11 2021 19:28:06 GMT+0800 (Singapore Standard Time) 
4	eval_side_by_side_4_0 eval tag: eval_side_by_side_4_0 step **38,800** Sat Dec 11 2021 19:28:07 GMT+0800 (Singapore Standard Time) 
5	eval_side_by_side_5_0 eval tag: eval_side_by_side_5_0 step **38,800** Sat Dec 11 2021 19:28:07 GMT+0800 (Singapore Standard Time) 
6	eval_side_by_side_6_0 eval tag: eval_side_by_side_6_0 step **38,800** Sat Dec 11 2021 19:28:07 GMT+0800 (Singapore Standard Time) 

Evaluation #	Evaluation Images
7	eval_side_by_side_7_0 tag: eval_side_by_side_7_0 step **38,800** Sat Dec 11 2021 19:28:08 GMT+0800 (Singapore Standard Time) eval 
8	eval_side_by_side_8_0 tag: eval_side_by_side_8_0 step **38,800** Sat Dec 11 2021 19:28:08 GMT+0800 (Singapore Standard Time) eval 
9	eval_side_by_side_9_0 tag: eval_side_by_side_9_0 step **38,800** Sat Dec 11 2021 19:28:08 GMT+0800 (Singapore Standard Time) eval 

Table 9 – Evaluation Images from experimental run 3

4.3 Run 4

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.546
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.847
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.619
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.334
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.553
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.446
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.636
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.686
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.461
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.694
INFO:tensorflow:Eval metrics at step 40300
```

Figure 14 - Evaluation results from experiment run 4

In this run, I changed the aspect ratios and removed random horizontal flip which randomly flips input images. I remove this since this is not relevant in this case. For aspect ratio, I removed the anchor box that is 2 high and added anchor boxes 1.33 and 4 high. This may not be that relevant as most of the images are more wide than high. Having said that, this set of hyper-parameters came close to the mAP values of the baseline run (Figure 14).

Figure 15 shows the loss and Table 10 shows the evaluation images from experimental run 4.

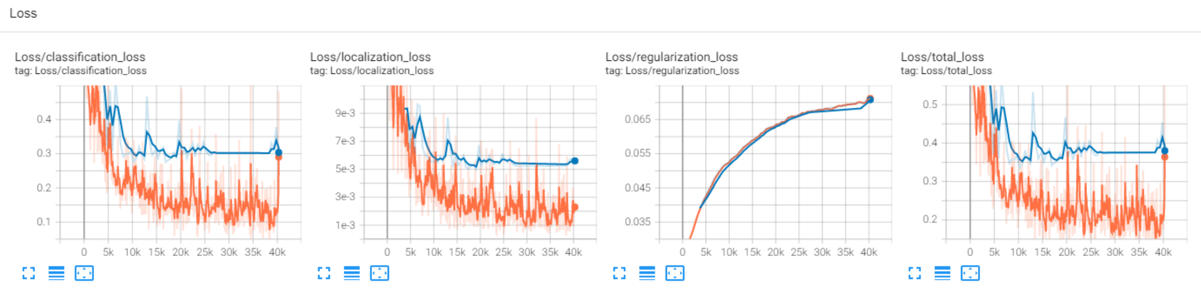
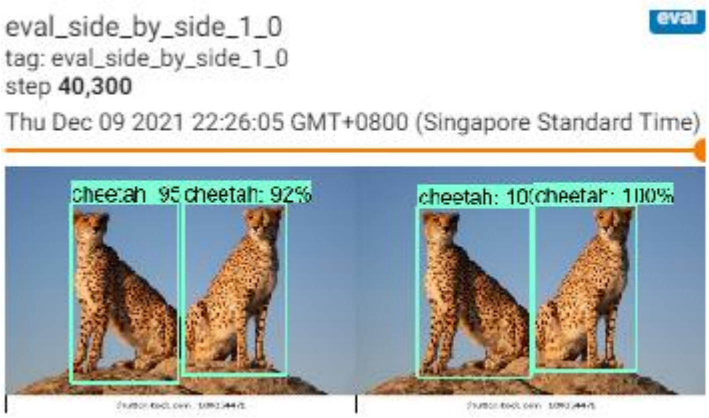
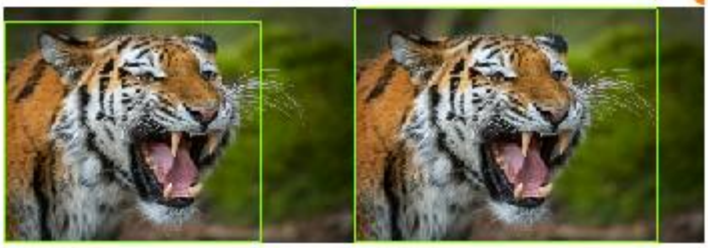
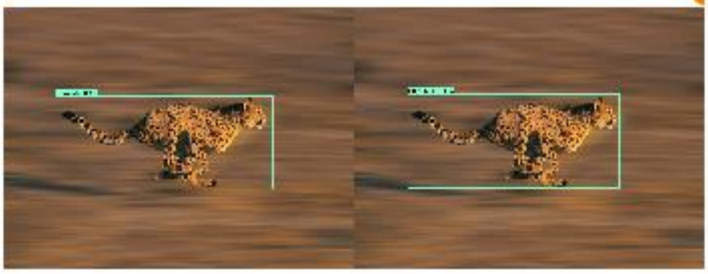
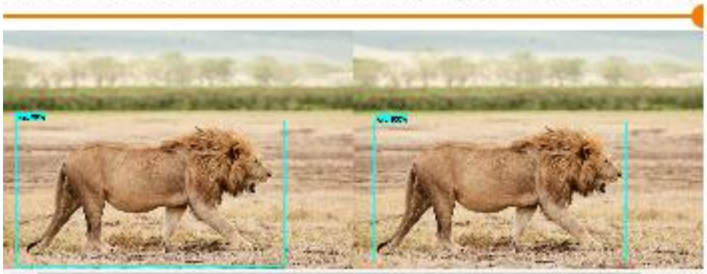


Figure 15 – Loss from experimental run 4

<table>
<tr><th>Evaluation #</th><th>Evaluation Images</th></tr>
<tr><td>0</td><td>
</td></tr>
<tr><td>1</td><td>
</td></tr>
</table>

Evaluation #	Evaluation Images
2	eval_side_by_side_2_0 eval tag: eval_side_by_side_2_0 step **40,300** Thu Dec 09 2021 22:26:05 GMT+0800 (Singapore Standard Time) 
3	eval_side_by_side_3_0 eval tag: eval_side_by_side_3_0 step **40,300** Thu Dec 09 2021 22:26:05 GMT+0800 (Singapore Standard Time) 
4	eval_side_by_side_4_0 eval tag: eval_side_by_side_4_0 step **40,300** Thu Dec 09 2021 22:26:06 GMT+0800 (Singapore Standard Time) 
5	eval_side_by_side_5_0 eval tag: eval_side_by_side_5_0 step **40,300** Thu Dec 09 2021 22:26:06 GMT+0800 (Singapore Standard Time) 

Evaluation #	Evaluation Images
6	
7	
8	
9	

Table 10 – Evaluation Images from experimental run 4

4.4 Run 5

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.001
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.001
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.024
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.104
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.105
INFO:tensorflow:Eval metrics at step 22900
```

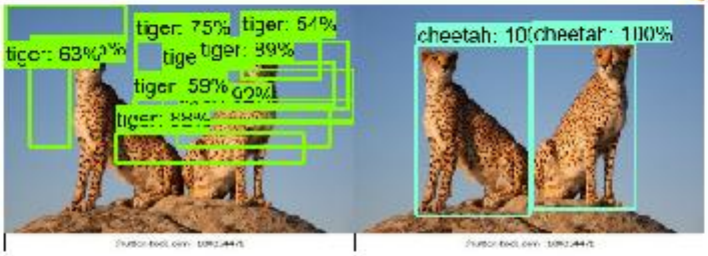
Figure 16 - Evaluation results from experiment run 5

In this experimental run, image normalization was applied. However, the mAP values became worse instead of improving as training times increased (Figure 16). This is because for EfficientDet, input pre-processing is included as part of the model, and EfficientDet models expect their inputs to be float tensors of pixels with values in the [0-255] range.

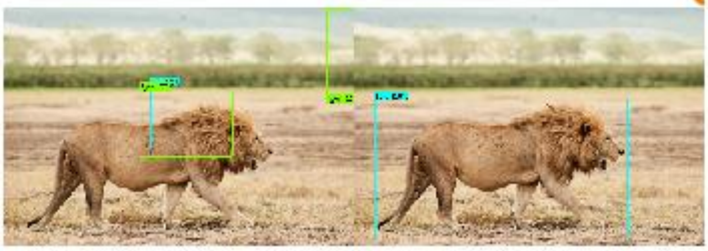
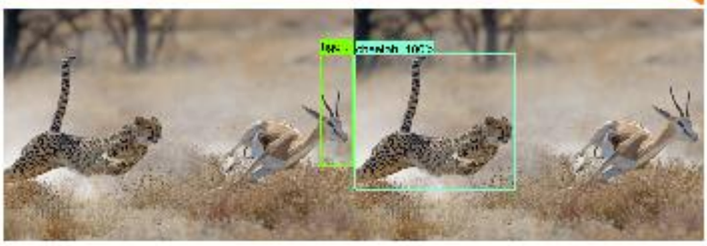
Figure 17 shows the loss and Table 11 shows the evaluation images from experimental run 5.

Figure 17 – Loss from experimental run 5

<table>
<tr><th>Evaluation #</th><th>Evaluation Images</th></tr>
<tr><td>0</td><td>
</td></tr>
</table>

<table>
<tr><th>Evaluation #</th><th>Evaluation Images</th></tr>
<tr><td>1</td><td>eval_side_by_side_1_0 eval
tag: eval_side_by_side_1_0
step 22,900
Sat Dec 11 2021 19:53:42 GMT+0800 (Singapore Standard Time)

</td></tr>
<tr><td>2</td><td>eval_side_by_side_2_0 eval
tag: eval_side_by_side_2_0
step 22,900
Sat Dec 11 2021 19:53:42 GMT+0800 (Singapore Standard Time)

</td></tr>
<tr><td>3</td><td>eval_side_by_side_3_0 eval
tag: eval_side_by_side_3_0
step 22,900
Sat Dec 11 2021 19:53:42 GMT+0800 (Singapore Standard Time)
</td></tr>
<tr><td>4</td><td>eval_side_by_side_4_0 eval
tag: eval_side_by_side_4_0
step 22,900
Sat Dec 11 2021 19:53:43 GMT+0800 (Singapore Standard Time)
</td></tr>
</table>

Evaluation #	Evaluation Images
5	eval_side_by_side_5_0 eval tag: eval_side_by_side_5_0 step **22,900** Sat Dec 11 2021 19:53:44 GMT+0800 (Singapore Standard Time) 
6	eval_side_by_side_6_0 eval tag: eval_side_by_side_6_0 step **22,900** Sat Dec 11 2021 19:53:44 GMT+0800 (Singapore Standard Time) 
7	eval_side_by_side_7_0 eval tag: eval_side_by_side_7_0 step **22,900** Sat Dec 11 2021 19:53:45 GMT+0800 (Singapore Standard Time) 
8	eval_side_by_side_8_0 eval tag: eval_side_by_side_8_0 step **22,900** Sat Dec 11 2021 19:53:45 GMT+0800 (Singapore Standard Time) 

Evaluation #	Evaluation Images
9	

Table 11 – Evaluation images from experimental run 5

4.5 Run 6

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.479
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.727
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.529
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.177
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.491
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.440
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.558
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.559
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.211
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.571
INFO:tensorflow:Eval metrics at step 28800
```

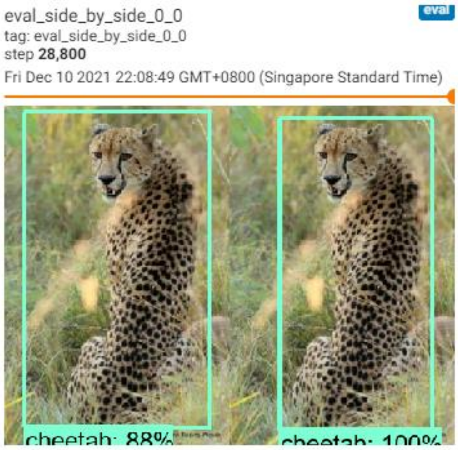
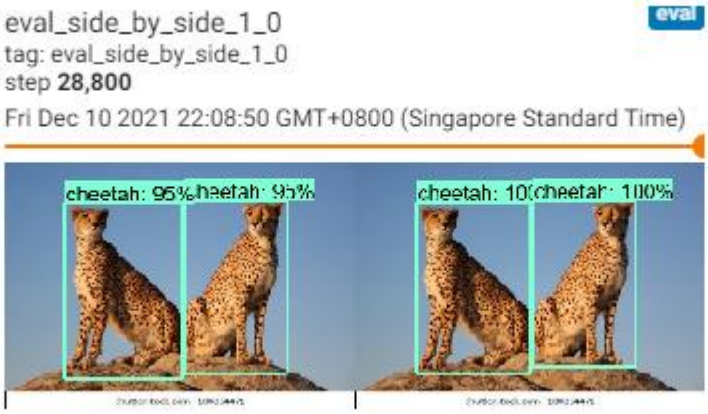
Figure 18 - Evaluation results from experiment run 6

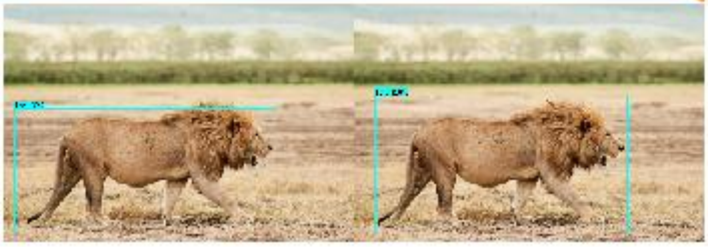
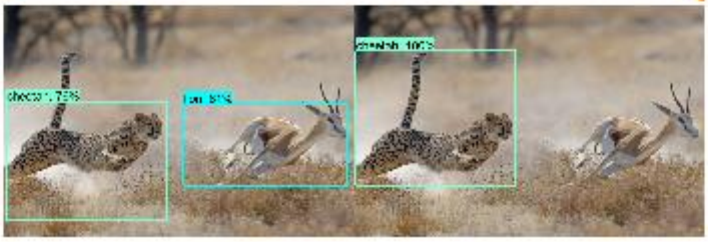
In the last experimental run, I decided to change the parameters in the post-processing step. Object detectors tend to generate hundreds of proposals. Most of them will not be accepted and will be eliminated. I decided to change the parameters that control model proposals. I changed the `score_threshold` from the default value close to 0 to 0.2. Setting the parameters allow the model to have an idea how many objects to expect and what their density is. This leads to better performance and will lower the chance of overfitting.

The mAP values from this run are shown in Figure 18. However, the values are not as good as the baseline run.

Figure 19 show the loss and Table 12 show the evaluation images from experimental run 6.

Figure 19 – Loss from experimental run 6

Evaluation #	Evaluation Images
0	
1	
2	
3	

Evaluation #	Evaluation Images
4	eval_side_by_side_4_0 eval tag: eval_side_by_side_4_0 step **28,800** Fri Dec 10 2021 22:08:51 GMT+0800 (Singapore Standard Time) 
5	eval_side_by_side_5_0 eval tag: eval_side_by_side_5_0 step **28,800** Fri Dec 10 2021 22:08:51 GMT+0800 (Singapore Standard Time) 
6	eval_side_by_side_6_0 eval tag: eval_side_by_side_6_0 step **28,800** Fri Dec 10 2021 22:08:52 GMT+0800 (Singapore Standard Time) 
7	eval_side_by_side_7_0 eval tag: eval_side_by_side_7_0 step **28,800** Fri Dec 10 2021 22:08:52 GMT+0800 (Singapore Standard Time) 

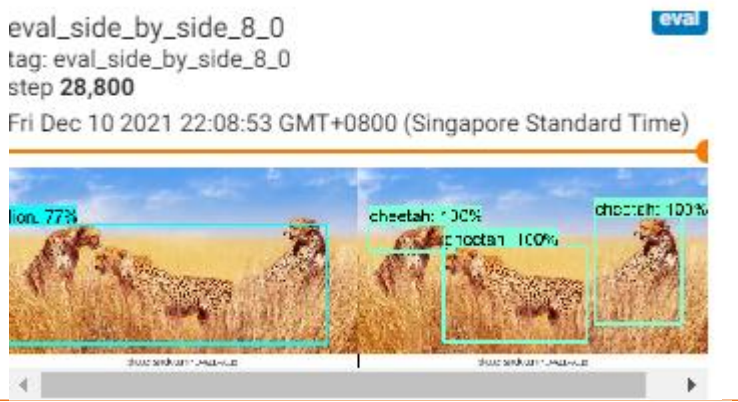
Evaluation #	Evaluation Images
8	
9	

Table 12 – Evaluation images from experimental run 6

5 Conclusion

To conclude, I used the pre-trained model, 'EfficientDet D0 512x512' to train a lion, tiger and cheetah detector. Five experiments were run.

In the baseline experiment, I only changed the batch size, number of steps to train, number of classes (equal to the number of objects I want to detect), changed 'classification' to 'detection'.

For each of the different experimental runs, I changed the following:

- Run 3 – loss function
- Run 4 – anchor boxes
- Run 5 – image normalization
- Run 6 – NMS parameters

The best mAP value was obtained using the baseline run. Minimal tuning was required because the objects to be detected were quite similar to the domain of the original dataset and is not from a completely different domain for example, X-ray images.