ITI107 – DEEP LEARNING NETWORKS

DECEMBER 2021

AFRICA BIG FIVE DETECTION

Yap Jing Yang (21B583W)

Ng Wei Xiang (21B576M)

Table of Contents

Member Contributions

Name	Admin No.	Contributions
Yap Jing Yang	21B583W	● ResNet50 Training and Fine-tuning ● EfficientDet D1 Training and Fine-tuning ● Setup of GITHUB project repository
Ng Wei Xiang	21B576M	● Image Scraping ● Image Annotation ● Image EDA ● EfficientDet Training and Finetuning

1. Introduction (0.5 pages)

For this assignment, we chose to detect the Big Five of Africa, namely the buffalo, elephant, leopard, lion and rhino.

1.1. Dataset

The dataset consists of 1000 annotated images, with about 200 images for each class, and a 80-20 train-validation split. The dataset can be downloaded from our project repository.

The buffalo, elephant and rhino annotated images are taken from the Kaggle African Wildlife dataset. As these images were annotated also with zebra class and also did not include the lion and leopard classes, the images were scanned through to remove the zebra class, and to add in the lion and leopard classes. As for the lion and leopard images, they are scraped using the jmd_imagescraper library and annotated. All annotations, annotation format conversions and train-validation split are done using roboflow.

2. Yap Jing Yang (2 pages)

2.1. SSD ResNet50 V1 FPN 640x640

2.1.1. Setting up & Training the Model

For the training of the model, I followed the guide as per the lab session number 4 covered by Mr Mar. The link can be found here.

I configured the pipeline.config file as per my needs. I chose a batch size of 10 and use a learning rate of 0.04. I trained the model for about five hours and stopped training when the mean mAP reached a pretty high value.

2.1.2. Results of the training

I had tabulated the results in a table and provided screenshots of the mAP, loss and evaluation images from Tensorboard. All the screenshots are stored in a separate folder labelled "screenshots".

mAP@0.5IOU	mAP@0.75IOU	mAP@small objects	mAP@large objects
0.998	0.994	0.95	0.907

These values are extracted from the screenshot of the evaluation results.
The model does not seem to be overfitting as the training and validation loss curves almost converge at the end of the training session.

2.1.3. Testing the model on the test image and video clip

After the training, I evaluated the model using an image (sample_big_five_image.jpg) that contains all the five animals and tabulated the results in the table below.

Buffalo	Elephant	Lion	Rhino	Leopard
93%	97%	46%	51%	79%

The accuracy for the lion and rhino is not so good and the reason might be due to the colour contrast of the lion and the rhino is similar to their respective background. When I evaluate the lion accuracy again in another picture (test_image_lion.jpg) which has a greener background, I got a higher accuracy of 85%.

Unfortunately for the test video(sample_big_five_vid.mp4), the results are not that good. Although it can detect most of the animals, but it cannot detect the lion class well. The reason might be due to the background noise being dominant.

2.2. EfficientDet D1 640x640

2.2.1. Setting up & Training the Model

For the training of the model, I followed the guide as per the lab session number 4 covered by Mr Mar. The link can be found here.

I configured the pipeline.config file as per my needs. I chose a batch size of 10 and use a learning rate of 0.04. During the training, I encountered several disconnects and decide to stop after about three hours. The disconnects might be due to high memory usage held by the training of the model.

2.2.2. Results of the training

I had tabulated the results in a table and provided screenshots of the mAP, loss and evaluation images from Tensorboard. All the screenshots are stored in a separate folder labelled "screenshots".

mAP@0.5IOU	mAP@0.75IOU	mAP@small objects	mAP@large objects
0.993	0.775	0.302	0.675

These values are extracted from the screenshot of the evaluation results.
The model does not seem to be overfitting as the training and validation loss curves almost converge at the end of the training session.

2.2.3. Testing the model on the test image and video clip

After the training, I evaluated the model using an image (sample_big_five_image.jpg) that contains all the five animals and tabulated the results in the table below.

Buffalo	Elephant	Lion	Rhino	Leopard
97%	86%	90%	99%	92%

Surprisingly, I observed that this model did well on the test image, achieving an overall accuracy of more than 85% for all the big five, despite it having a lower mean mAP than the ResNet50.

As for the test video, the detections and accuracy is about the same as my ResNet run. But i believed that I can achieve better results if i train the model longer.

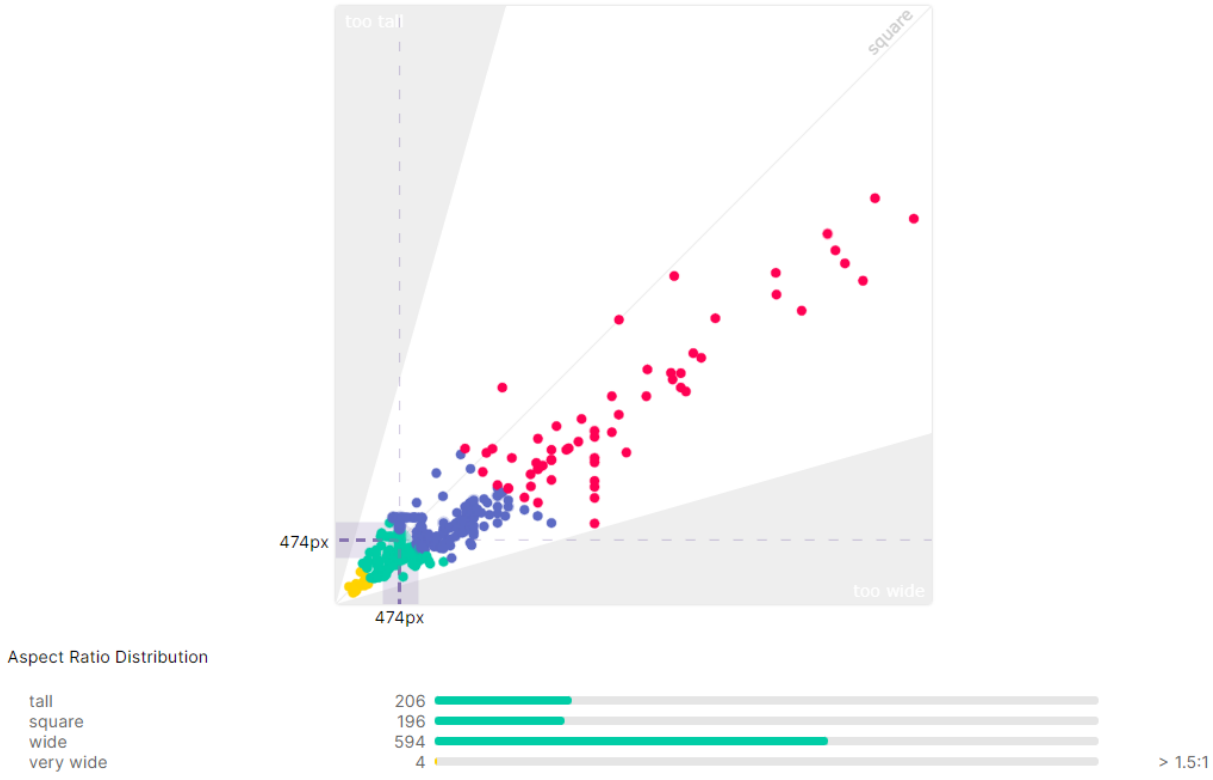
3. Ng Wei Xiang (3.5 pages)

3.1. Model Training

The model chosen is EfficientDet D0, the smallest of the EfficientDet family with a performance of 33.6 COCO mAP. Using EfficientNet B0 as the backbone and bi-directional feature pyramid network, it consists of 3.9M parameters.

3.1.1. Image Resizer

Two types of image resizers are explored. The first being the default resizer that keeps the original aspect ratio and pad the image to 512 by 512 with zeros on the right and bottom. While it preserves the aspect ratio, it might not be appropriate for images that are elongated as most of the resized image would simply be padding. This is as such in our case as seen in the distribution of the image aspect ratio as below.



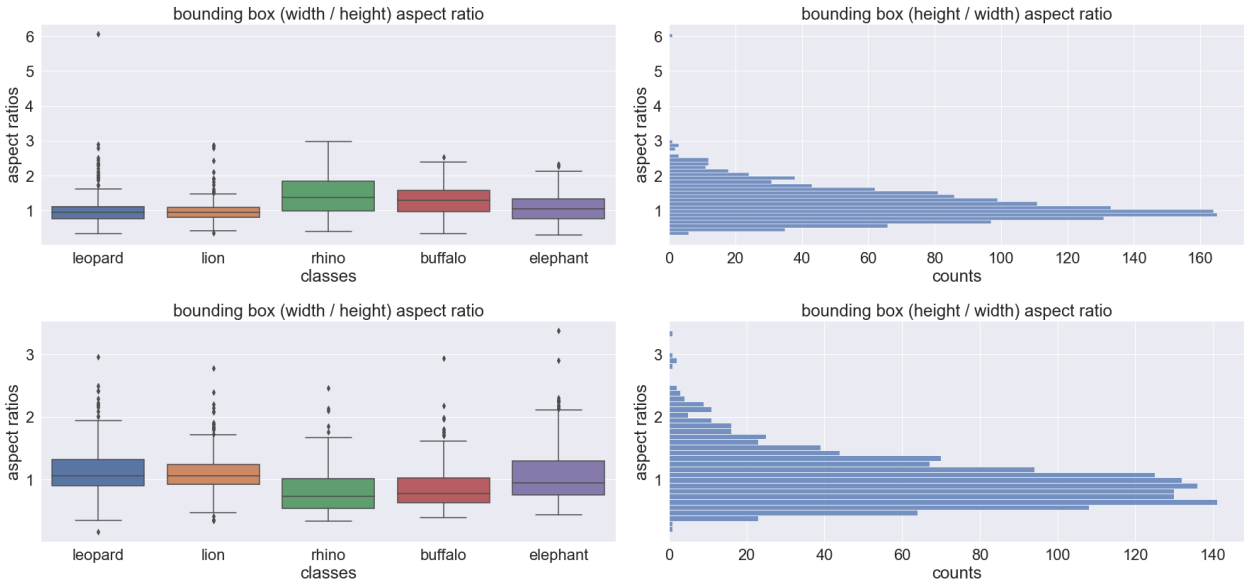
The second is the fixed shape resizer that stretches the image to 512 by 512. While this distorts the aspect ratio, it ensures that the entire resized image is filled with some useful information rather than padding.

3.1.2. Anchor Generation

The multiscale anchor box generator is used with various aspect ratios and scales per octave.

3.1.2.1. Anchor Box Aspect Ratio

Apart from the default anchor box aspect ratios of 1, 2 and 0.5, two other different sets of aspect ratios are used based on the bounding box aspect ratios of the data as follows.



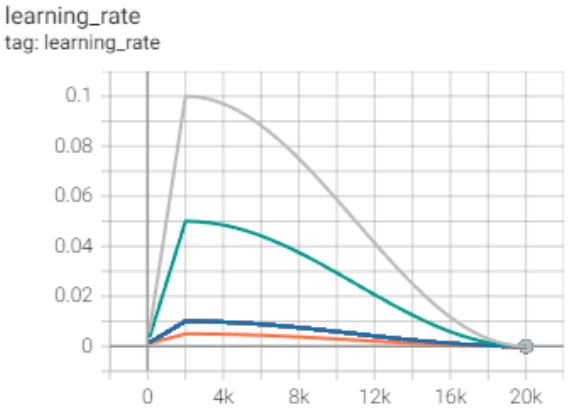
As seen from above, the majority of the bounding boxes have an aspect ratio up to 2.5. The width / height aspect ratio for the rhino class is especially higher with the upper median at about 3. As such, one set of anchor boxes was given aspect ratios of 1, 2, 0.5, 3 and 0.3333333333, while the other set of anchor boxes was given aspect ratios of 1, 1.6, 0.625, 2.2 and 0.4545454545, uniformly distributed up to 2.5 coverage.

3.1.2.2. Anchor Box Scales per Octave

Besides the aspect ratio of anchor boxes, scales per octave of 3 and 5 were also tried out, in hope of improving the performance on smaller objects.

3.1.3. Learning Rate

The initial learning rate is set at 0.001 and warms up linearly across 2000 steps to the base learning rate, of which various values of 0.1, 0.05, 0.01 and 0.005 were tried to find the optimal learning rate that converges the fastest. Afterwhich, the cosine learning rate decay kicks in through the next 18000 steps to a learning rate value of zero.



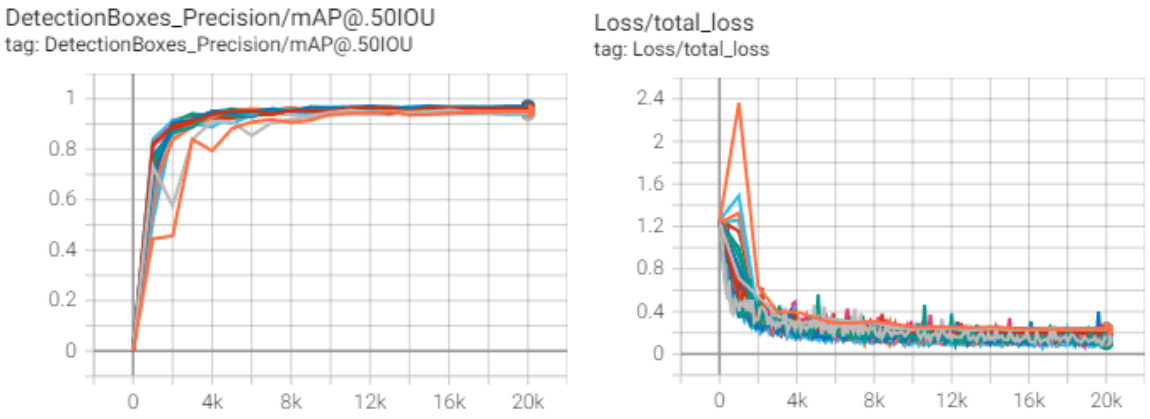
3.1.4. Data Augmentation

The default random cropping and scaling were kept as these simulates images taken from different distances and angles. However, the padding to square augmentation was removed in some experimental runs to ensure that all pixels of the augmented image contain some information rather than padding as mentioned in the previous section.

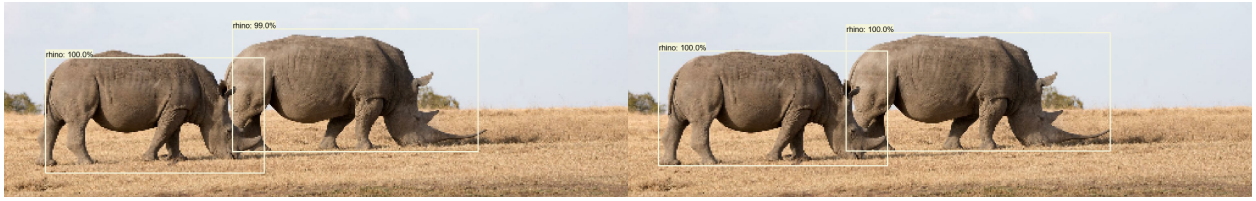
In hopes of improving the performance on noisier images, the brightness and contrast were randomly adjusted to simulate images that are taken in different lighting conditions, and gaussian patches were also randomly added to simulate how the objects of interest could be hidden behind other objects such as trees and bushes. As the performance of small objects were disappointing, random self concatenation and mosaic were also added as these would simulate more smaller objects.

3.2. Model Evaluation

The mAP@0.5IOU and loss of the 20 experimental runs are shown below.



One of the evaluation images from experimental run 3, which is deemed as the best model, is as follows, with the predicted on the left and the actual on the right.



There are no drastic improvements across the experimental runs. Although the finetuning did improve the performance on small objects for certain runs, it unfortunately came with a cost of poorer performance on the large and medium objects.

The mAP for each class were also calculated and it can be observed that the elephant and leopard classes did not perform as well as the other classes. While it is understandable that the leopard class performs worse, the poorer performance of the elephant class came as a surprise considering the class balance of the training set as shown below.

3.3. Model Testing

The model seems to perform well on the test image, which could partly be due to its high resolution and squarish aspect ratio. But when fed with another smaller test image with elongated aspect ratio, it did not manage to perform as well.

However, for the test video, it can be seen that the model does have some difficulties when there are some noise involved, for instance when there are texts covering the objects. Another noteworthy point is the poor detection of the lion cubs and the leopard in night setting at about 21 and 45 seconds into the video respectively, which is probably due to the lack of such data in our dataset.

4. Conclusion (0.5 pages)

In conclusion, while the EfficientDet managed to perform decently, the ResNet achieved an impressive performance of over 0.9 mAP.

Model	mAP	mAP (0.5 IOU)	mAP (0.75 IOU)	mAP (small)	mAP (medium)	mAP (large)
ResNet50 640x640	0.904	0.998	0.994	0.95	0.865	0.907
EfficientDet D0	0.7139	0.9648	0.8276	0.4000	0.5851	0.7202
EfficientDet D1	0.667	0.933	0.775	0.302	0.468	0.675

Further work that could potentially further improve the performance might be scaling of both the model and dataset.

Annex A. References

1) African Wildlife Kaggle Dataset
   https://www.kaggle.com/biancaferreira/african-wildlife
2) jmd_imagescraper Documentation
   https://github.com/joedockrill/jmd_imagescraper
3) Roboflow Documentation
   https://docs.roboflow.com/
4) Tensorflow Object Detection Documentation
   https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/
5) NYP ITI107 Session 4
   https://github.com/nyp-sit/iti107/tree/main/session-4
6) Tensorflow 2 Detection Model Zoo
   https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md