Assignment 4 Report

Course Code: CS 512
Course Name: Computer Vision
Course Instructor: Dr. Gady Agam

AS4
Prepared by: Khalid Saifullah
Student ID: A20423546
Date: 18th April, 2023

1. Problem Statement

The programming task is divided into 2 parts, the first part is semantic segmentation and the second part is object detection. Oxford pet dataset is used for this assignment. The following are the outcomes:

- Train a simple convolutional neural network for supervised semantic segmentation without skip connections and evaluate its performance.
- Train a simple convolutional neural network for supervised semantic segmentation with skip connections and evaluate its performance.
- Use the pretrained TFSegformerForSemanticSegmentation model from the transformers module and continue to train it. Evaluate and visualize the results as before.
- Download and use a pre-trained YOLO model.
- Apply the model to predict results and show the detection results using bounding boxes drawn on the image.
- Evaluate the performance of the model on the cats/dogs dataset using the known labels.

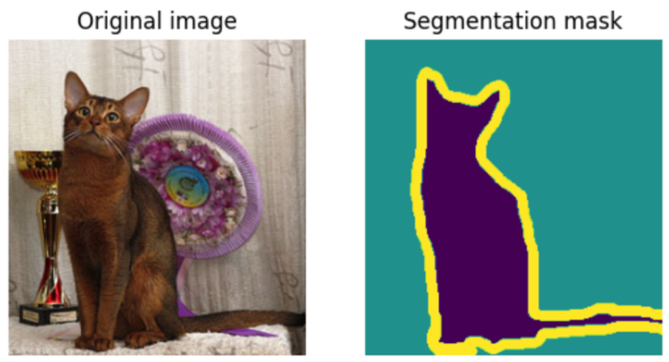
2. Proposed Solution

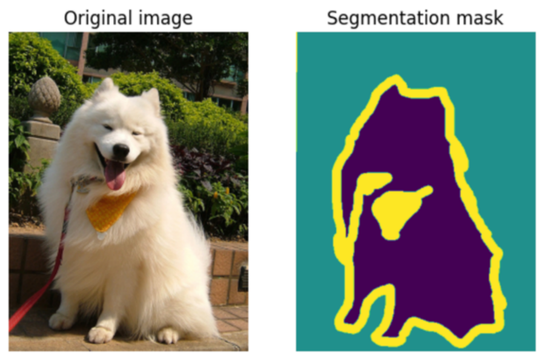
Keras and tensorflow packages were used along with python in google colab to preprocess the data, train the model and test it on the test dataset.

3. Implementation Details

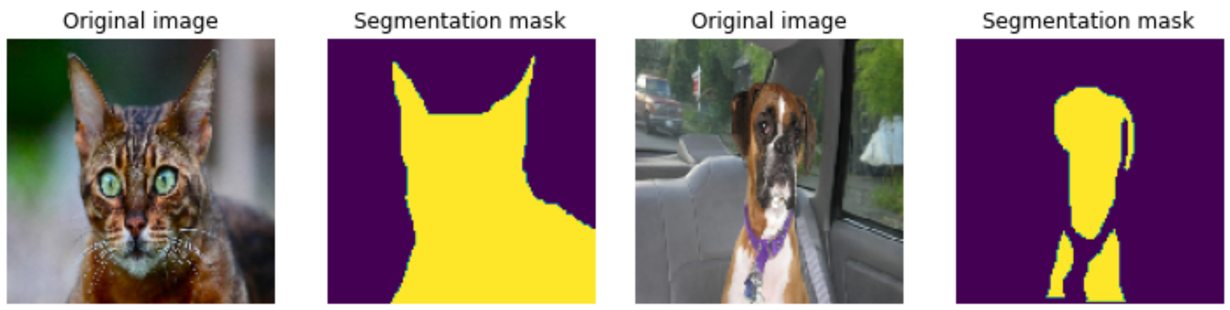
Programming Problem 1:

At first, the Oxford cats/dogs dataset is downloaded and then the cat and dog breed labels were converted into cats/dogs label categories. Afterwards, a user defined function is called to display sample images and their corresponding masks.

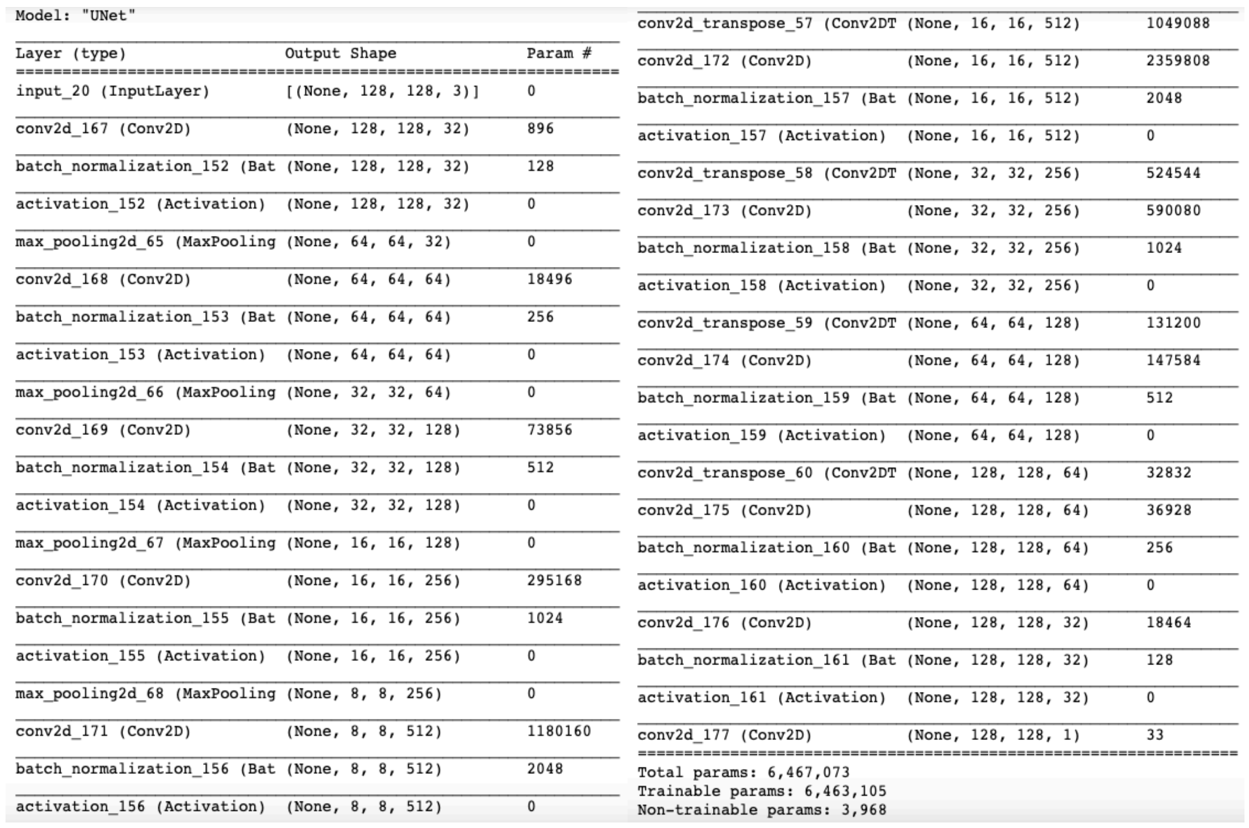




The dataset consists of 37 category pets with roughly 200 images for each class. It has 7349 images; 2371 cats and 4978 dogs. The original segmentation masks were labelled as 1, 2 and 3. In order to binarize the masks, the mask was thresholded at 2, i.e. any values equal or above 2 is considered to be 1 and others are all 0. This resulted in the following image segmentation.

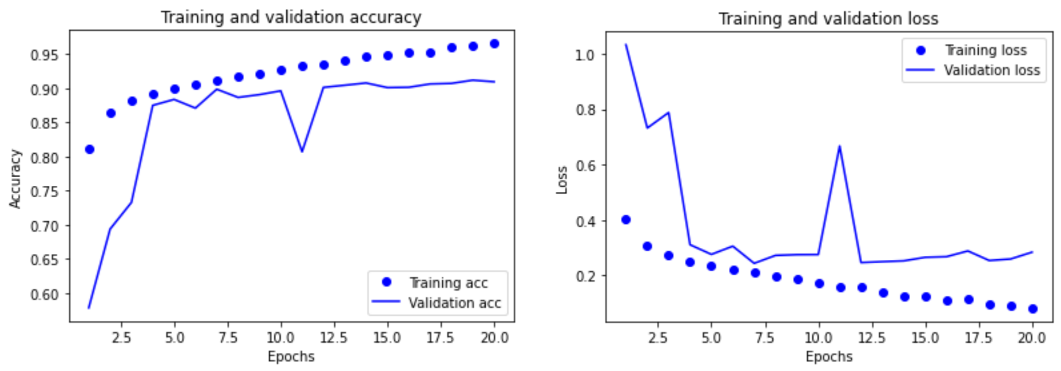


All the images and their corresponding labels went through a series of preprocessing. They were renamed and copied into new directories. The images were then separated into training, validation and testing datasets. A UNet model was defined as follows. Note that the input image shape of each image was 128x128x3 as they were rgb images. The mask shapes were, however, 128x128x1.



```
Model: "UNet"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_20 (InputLayer)        [(None, 128, 128, 3)]     0
_________________________________________________________________
conv2d_167 (Conv2D)          (None, 128, 128, 32)      896
_________________________________________________________________
batch_normalization_152 (Bat (None, 128, 128, 32)      128
_________________________________________________________________
activation_152 (Activation)  (None, 128, 128, 32)      0
_________________________________________________________________
max_pooling2d_65 (MaxPooling (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_168 (Conv2D)          (None, 64, 64, 64)        18496
_________________________________________________________________
batch_normalization_153 (Bat (None, 64, 64, 64)        256
_________________________________________________________________
activation_153 (Activation)  (None, 64, 64, 64)        0
_________________________________________________________________
max_pooling2d_66 (MaxPooling (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_169 (Conv2D)          (None, 32, 32, 128)       73856
_________________________________________________________________
batch_normalization_154 (Bat (None, 32, 32, 128)       512
_________________________________________________________________
activation_154 (Activation)  (None, 32, 32, 128)       0
_________________________________________________________________
max_pooling2d_67 (MaxPooling (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_170 (Conv2D)          (None, 16, 16, 256)       295168
_________________________________________________________________
batch_normalization_155 (Bat (None, 16, 16, 256)       1024
_________________________________________________________________
activation_155 (Activation)  (None, 16, 16, 256)       0
_________________________________________________________________
max_pooling2d_68 (MaxPooling (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_171 (Conv2D)          (None, 8, 8, 512)         1180160
_________________________________________________________________
batch_normalization_156 (Bat (None, 8, 8, 512)         2048
_________________________________________________________________
activation_156 (Activation)  (None, 8, 8, 512)         0
_________________________________________________________________
conv2d_transpose_57 (Conv2DT (None, 16, 16, 512)       1049088
_________________________________________________________________
conv2d_172 (Conv2D)          (None, 16, 16, 512)       2359808
_________________________________________________________________
batch_normalization_157 (Bat (None, 16, 16, 512)       2048
_________________________________________________________________
activation_157 (Activation)  (None, 16, 16, 512)       0
_________________________________________________________________
conv2d_transpose_58 (Conv2DT (None, 32, 32, 256)       524544
_________________________________________________________________
conv2d_173 (Conv2D)          (None, 32, 32, 256)       590080
_________________________________________________________________
batch_normalization_158 (Bat (None, 32, 32, 256)       1024
_________________________________________________________________
activation_158 (Activation)  (None, 32, 32, 256)       0
_________________________________________________________________
conv2d_transpose_59 (Conv2DT (None, 64, 64, 128)       131200
_________________________________________________________________
conv2d_174 (Conv2D)          (None, 64, 64, 128)       147584
_________________________________________________________________
batch_normalization_159 (Bat (None, 64, 64, 128)       512
_________________________________________________________________
activation_159 (Activation)  (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_transpose_60 (Conv2DT (None, 128, 128, 64)      32832
_________________________________________________________________
conv2d_175 (Conv2D)          (None, 128, 128, 64)      36928
_________________________________________________________________
batch_normalization_160 (Bat (None, 128, 128, 64)      256
_________________________________________________________________
activation_160 (Activation)  (None, 128, 128, 64)      0
_________________________________________________________________
conv2d_176 (Conv2D)          (None, 128, 128, 32)      18464
_________________________________________________________________
batch_normalization_161 (Bat (None, 128, 128, 32)      128
_________________________________________________________________
activation_161 (Activation)  (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_177 (Conv2D)          (None, 128, 128, 1)       33
=================================================================
Total params: 6,467,073
Trainable params: 6,463,105
Non-trainable params: 3,968
```

The model was trained for 20 epochs with a learning rate of 0.0001 and 'adam' as the optimizer, 'binary_crossentropy' as the loss function and 'accuracy' as the metric. The resulting curves indicating loss and accuracy are given below.



It can be seen that the model's accuracy increases with each epoch and the loss is being minimized as well.

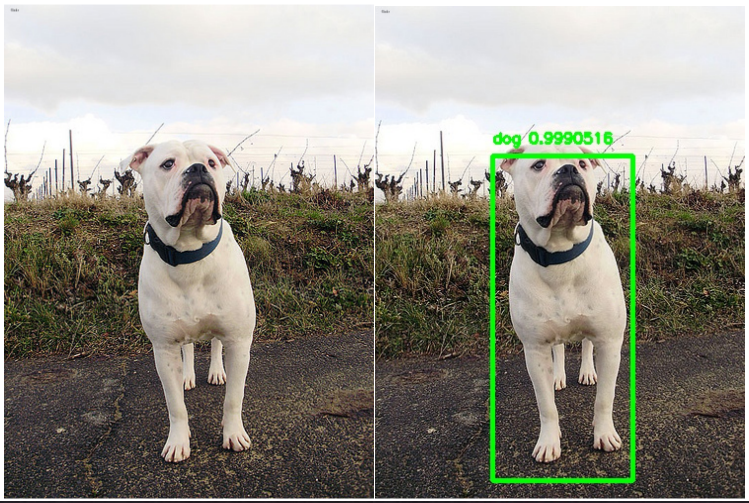
Programming Problem 2:

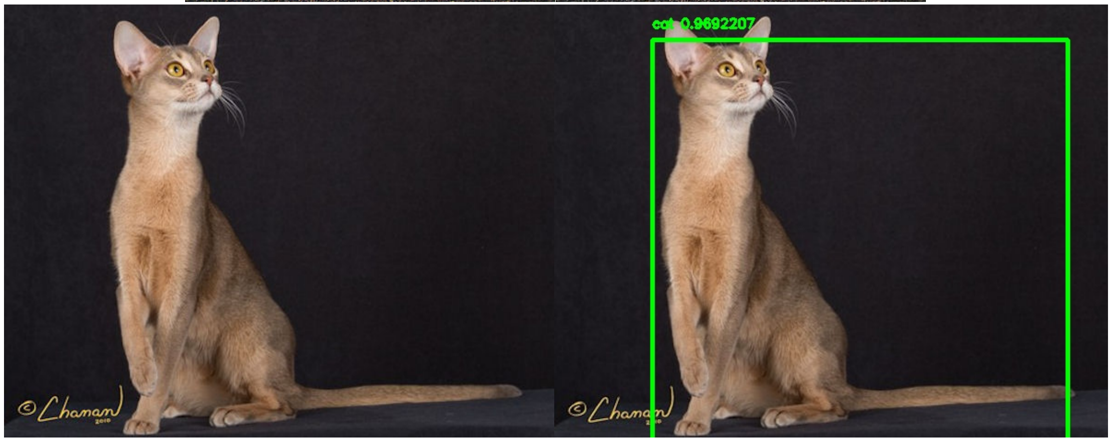
YOLO (You Only Look Once) version 3 is a real-time object detection system developed by Joseph Redmon and Ali Farhadi in 2018. It builds upon the previous versions of YOLO with a number of improvements to its architecture and training process. Here are some key features of YOLOv3:

- Improved accuracy: YOLOv3 is able to achieve state-of-the-art object detection accuracy with a relatively small number of parameters and training time compared to other methods.
- Multi-scale feature extraction: YOLOv3 uses a feature pyramid network to extract features at different scales and resolutions from the input image, allowing it to detect objects of various sizes more effectively.
- Backbone network: YOLOv3 uses a variant of the DarkNet neural network as its backbone, which provides a good trade-off between accuracy and computational efficiency.
- Non-max suppression: To remove duplicate detections of the same object, YOLOv3 uses a non-maximum suppression algorithm that compares the confidence scores of overlapping bounding boxes and removes the ones with lower scores.
- Object classification: YOLOv3 performs object classification by dividing the image into a grid of cells and predicting the probability of each cell containing an object of a certain class. This allows it to detect multiple objects in a single image.
- Training: YOLOv3 is trained using a variant of the DarkNet network with a focus on maximizing both object detection accuracy and computational efficiency.

YOLOv3 is widely used for real-time object detection tasks, such as traffic monitoring, surveillance, and robotics.

In this part, a pre-trained YOLO model was downloaded from [2] and made keras compatible after loading it. A sample image was loaded and normalized the model is then used to predict the input image. Some sample results are given below with the accuracy of classification and their corresponding bounding boxes.





4. Results and Discussion

It should be mentioned that a few issues were encountered while solving the programming questions. The pets dataset had 7 images which couldn't be opened or were empty. Because of these images, the training and testing were showing errors. They were eventually filtered out by deleting manually.

5. References

1. https://www.robots.ox.ac.uk/~vgg/data/pets/
2. https://modelzoo.co/model/keras-yolov3