Computer Vision Assignment 1

Omer Kamal Ali Ebead

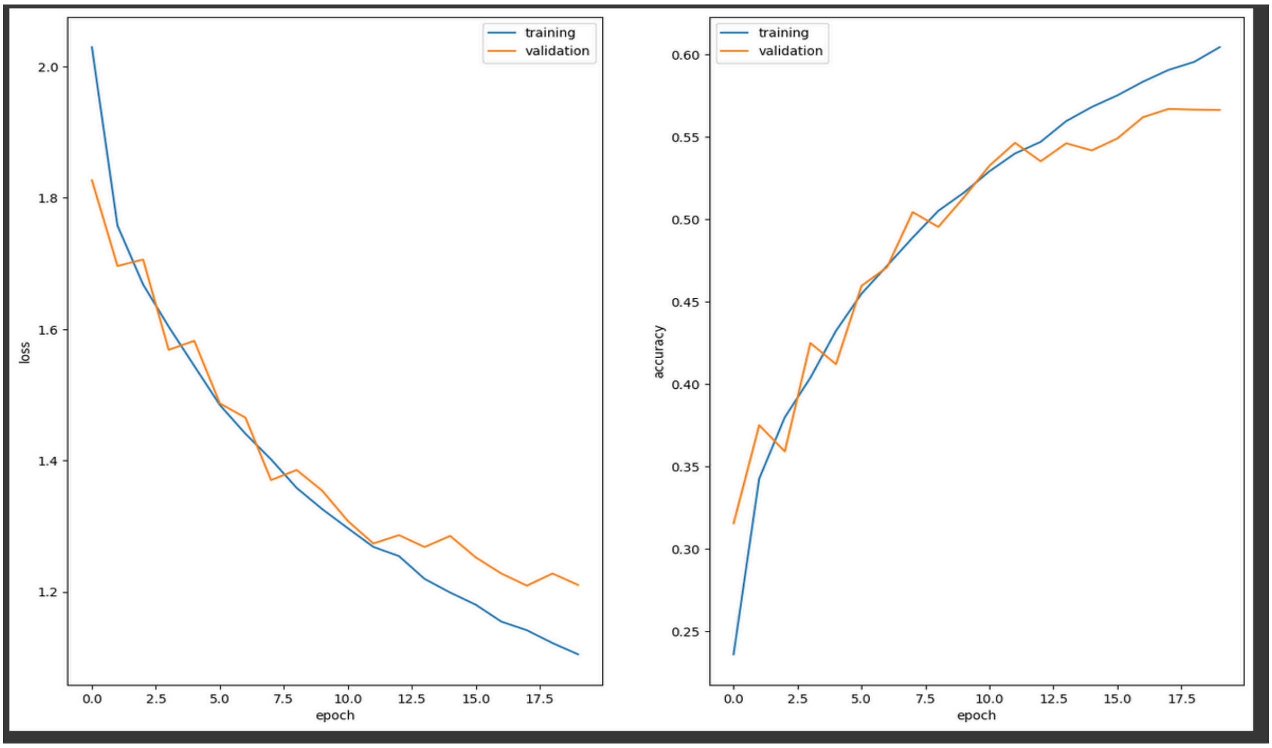
Problem 1:

Preprocessing steps:

- Convert cifar-10 y labels to one hot vector so it can match the softmax output
- Normalising cifar-10 x images by dividing them by 255.0
- Try different architectures

1- Simple architecture:

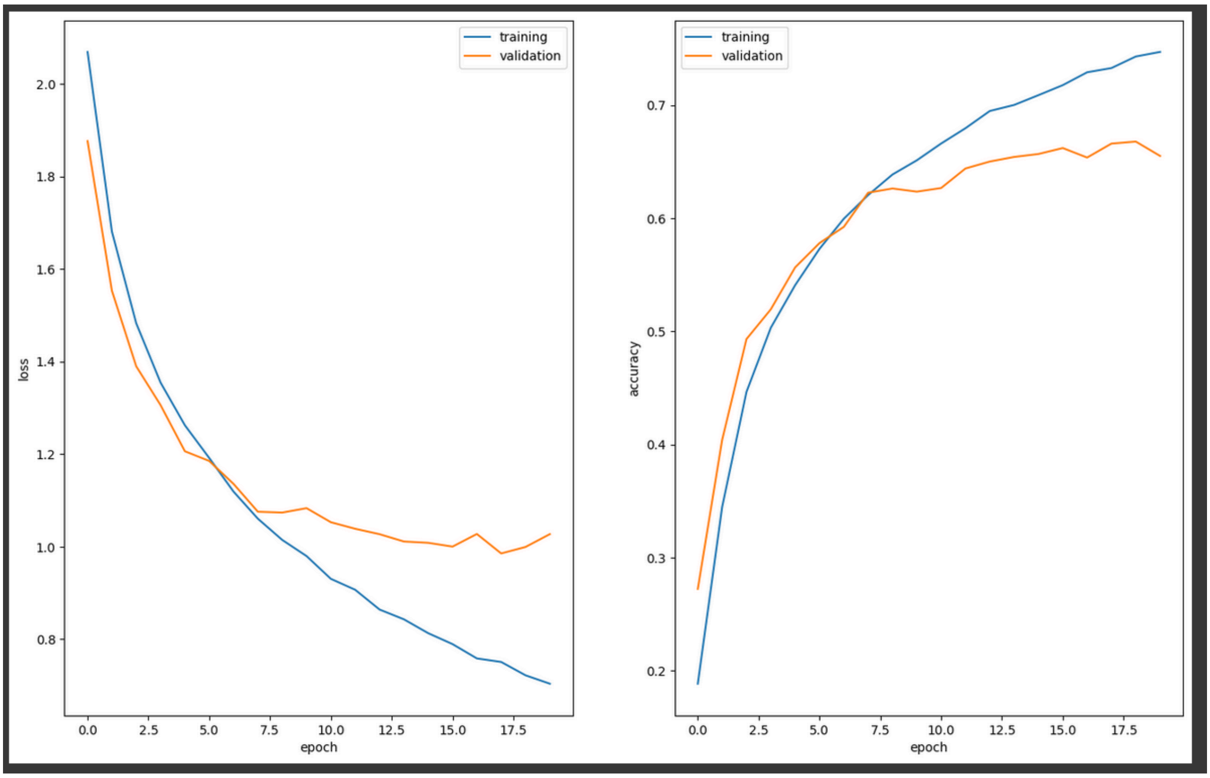
```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_22 (Conv2D)          (None, 32, 32, 4)         112

 average_pooling2d_20 (Aver  (None, 16, 16, 4)         0
 agePooling2D)

 conv2d_23 (Conv2D)          (None, 16, 16, 8)         520

 average_pooling2d_21 (Aver  (None, 8, 8, 8)           0
 agePooling2D)

 conv2d_24 (Conv2D)          (None, 8, 8, 16)          2064

 average_pooling2d_22 (Aver  (None, 4, 4, 16)          0
 agePooling2D)

 conv2d_25 (Conv2D)          (None, 4, 4, 32)          8224

 average_pooling2d_23 (Aver  (None, 2, 2, 32)          0
 agePooling2D)

 flatten_9 (Flatten)         (None, 128)               0

 dense_18 (Dense)            (None, 256)               33024

 dense_19 (Dense)            (None, 128)               32896

 dense_20 (Dense)            (None, 64)                8256

 dense_21 (Dense)            (None, 10)                650

=================================================================
Total params: 85746 (334.95 KB)
Trainable params: 85746 (334.95 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



- Around 55% validation accuracy.
- I needed to add more regularisation to the model and maybe make it more complex so the accuracy went higher.

2- Adding dropout and increase the number of layers:

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_26 (Conv2D)          (None, 32, 32, 8)         608

 max_pooling2d (MaxPooling2  (None, 16, 16, 8)         0
 D)

 conv2d_27 (Conv2D)          (None, 16, 16, 16)        2064

 max_pooling2d_1 (MaxPoolin  (None, 8, 8, 16)          0
 g2D)

 conv2d_28 (Conv2D)          (None, 8, 8, 32)          8224

 dropout (Dropout)           (None, 8, 8, 32)          0

 conv2d_29 (Conv2D)          (None, 8, 8, 64)          32832

 max_pooling2d_2 (MaxPoolin  (None, 4, 4, 64)          0
 g2D)

 conv2d_30 (Conv2D)          (None, 4, 4, 128)         131200

 max_pooling2d_3 (MaxPoolin  (None, 2, 2, 128)         0
 g2D)

 dropout_1 (Dropout)         (None, 2, 2, 128)         0

 conv2d_31 (Conv2D)          (None, 2, 2, 256)         524544

 max_pooling2d_4 (MaxPoolin  (None, 1, 1, 256)         0
 g2D)

 flatten_10 (Flatten)        (None, 256)               0

 dense_22 (Dense)            (None, 5012)              1288084

 dense_23 (Dense)            (None, 256)               1283328

 dense_24 (Dense)            (None, 10)                2570

=================================================================
Total params: 3273454 (12.49 MB)
Trainable params: 3273454 (12.49 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



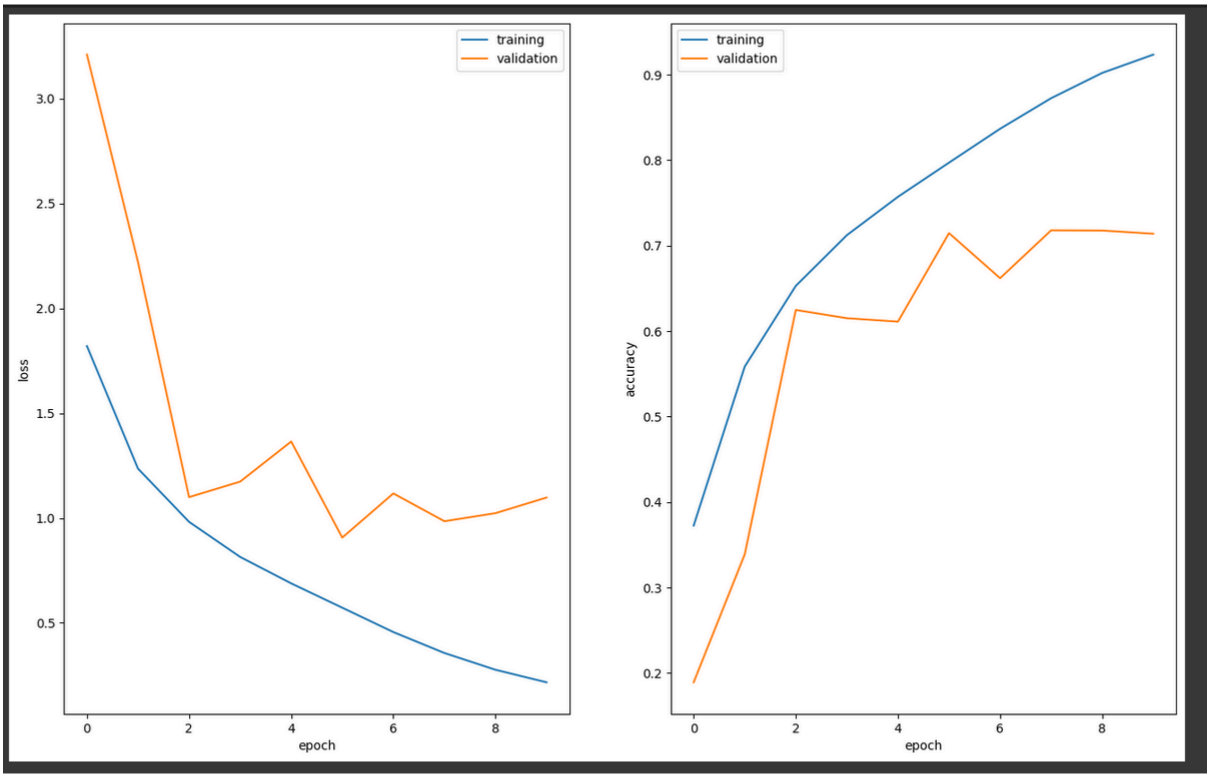
- Around 65% validation accuracy.

- Training and validation losses were decreasing until a point where it seems that the model is overfitting (7.5 epoch). I try early stopping next and Batch Normalisation.

3- Add Early stopping and Batch Normalisation:

Increasing the number of epochs to 100 and adding patience of 4 to the early stopping callback.

```
model3.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 8)         392

 conv2d_1 (Conv2D)           (None, 32, 32, 16)        2064

 conv2d_2 (Conv2D)           (None, 32, 32, 32)        8224

 batch_normalization (Batch  (None, 32, 32, 32)        128
 Normalization)

 dropout (Dropout)           (None, 32, 32, 32)        0

 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0
 D)

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        32832

 conv2d_4 (Conv2D)           (None, 16, 16, 128)       131200

 dropout_1 (Dropout)         (None, 16, 16, 128)       0

 conv2d_5 (Conv2D)           (None, 16, 16, 256)       524544

 batch_normalization_1 (Bat  (None, 16, 16, 256)       1024
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 8, 8, 256)         0
 g2D)

 flatten (Flatten)           (None, 16384)             0

 dense (Dense)               (None, 256)               4194560

 dense_1 (Dense)             (None, 128)               32896

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 10)                650

=================================================================
Total params: 4936770 (18.83 MB)
Trainable params: 4936194 (18.83 MB)
Non-trainable params: 576 (2.25 KB)
_________________________________________________________________
```



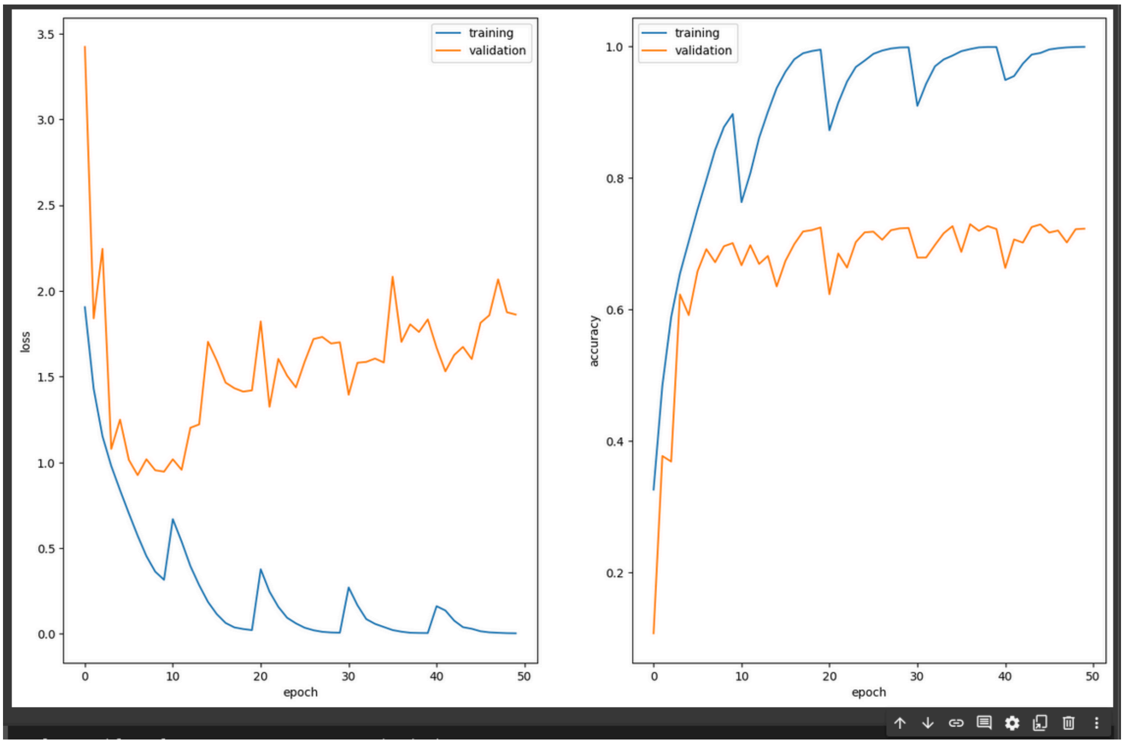
- Around 70% validation accuracy.
- The model stopped after the 10th epoch to prevent overfitting on the training data.

4- Add Snapshot Ensampling:

Paper: https://arxiv.org/pdf/1704.00109v1
Tutorial : https://www.kaggle.com/code/fkdplc/snapshot-ensemble-tutorial-with-keras

- I trained 5 models for 50 epochs (1 model per 10 epochs) and saved snapshots for each model (model parameters).

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 8)         392

 dropout (Dropout)           (None, 32, 32, 8)         0

 conv2d_1 (Conv2D)           (None, 32, 32, 16)        2064

 dropout_1 (Dropout)         (None, 32, 32, 16)        0

 conv2d_2 (Conv2D)           (None, 32, 32, 32)        8224

 batch_normalization (Batch  (None, 32, 32, 32)        128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 16, 16, 32)        0
 D)

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        32832

 batch_normalization_1 (Bat  (None, 16, 16, 64)        256
 chNormalization)

 dropout_2 (Dropout)         (None, 16, 16, 64)        0

 conv2d_4 (Conv2D)           (None, 16, 16, 128)       131200

 batch_normalization_2 (Bat  (None, 16, 16, 128)       512
 chNormalization)

 dropout_3 (Dropout)         (None, 16, 16, 128)       0

 conv2d_5 (Conv2D)           (None, 16, 16, 256)       524544

 batch_normalization_3 (Bat  (None, 16, 16, 256)       1024
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 8, 8, 256)         0
 g2D)

 flatten (Flatten)           (None, 16384)             0

 dense (Dense)               (None, 256)               4194560

 dense_1 (Dense)             (None, 128)               32896

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 10)                650

=================================================================
```

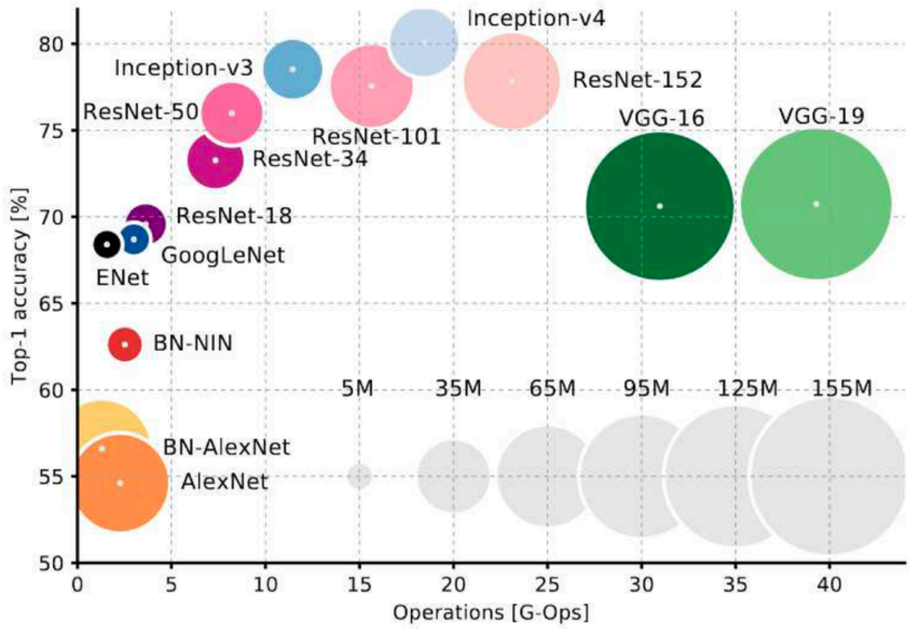



```
[10] result4 = model4.evaluate(norm_x_test,norm_y_test,batch_size=128)
79/79 [==============================] - 3s 40ms/step - loss: 1.8794 - accuracy: 0.7207 - categorical_accuracy: 0.7207
```

- Validation accuracy differs from each model to the other but on average 70% accuracy.
- On testset it evaluates to 72%
- Maybe regularisation and increasing the number of epochs per model for can help optimise the results but I got caught by time.

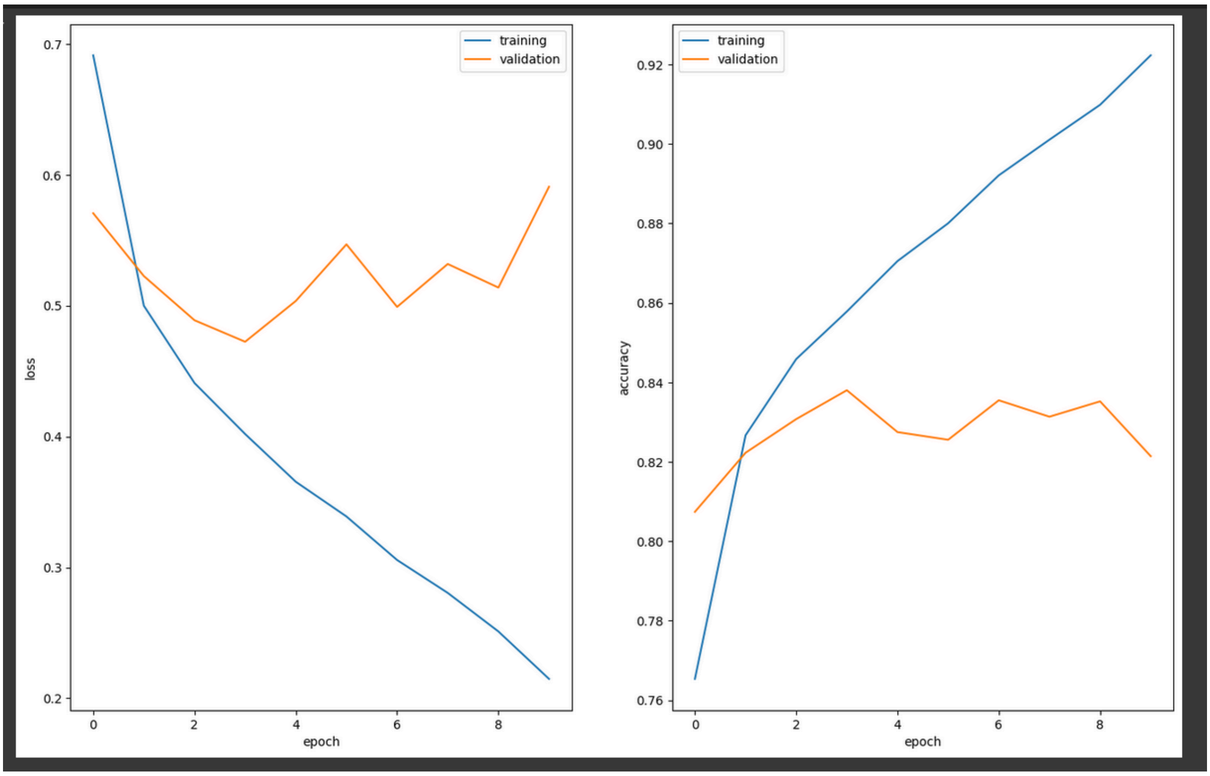
Problem 2:

- I focused on the main three factors, operations, total accuracy and number of parameters. Therefore, I chose inception V3.
- I also considered choosing Inception V4 but I was not convinced that to achieve just a little higher accuracy, we need more operations and more parameters.



- Inception V3 received 256*256 pixels so i had to resize my dataset (32*32 pixels) using bilinear interpolation (take the average of the closest 4 pixels)
- I removed the Inception V3 model classifier and froze its feature extraction weights and added my classifier on top.
- The resulting architecture became :

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 256, 256, 3)]     0

 inception_v3 (Functional)   (None, 6, 6, 2048)        21802784

 global_average_pooling2d_1  (None, 2048)              0
  (GlobalAveragePooling2D)

 dense_4 (Dense)             (None, 256)               524544

 dense_5 (Dense)             (None, 128)               32896

 dense_6 (Dense)             (None, 64)                8256

 dense_7 (Dense)             (None, 10)                650

=================================================================
Total params: 22369130 (85.33 MB)
Trainable params: 566346 (2.16 MB)
Non-trainable params: 21802784 (83.17 MB)
_________________________________________________________________
```

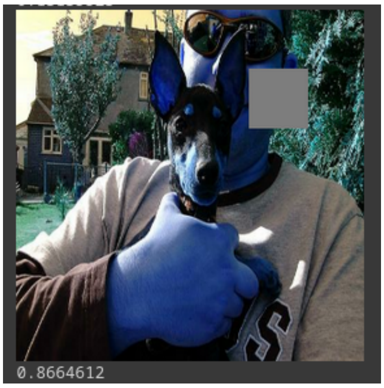


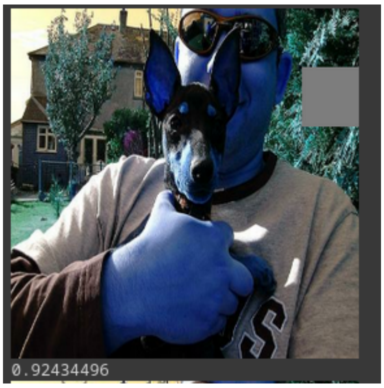

```
norm_x_test2 = resizing(norm_x_test)
result5 = model5.evaluate(norm_x_test2,norm_y_test,batch_size=128)

/79 [==============================] - 81s 1s/step - loss: 0.5073 - accuracy: 0.8354 - categorical_accuracy: 0.8354
```

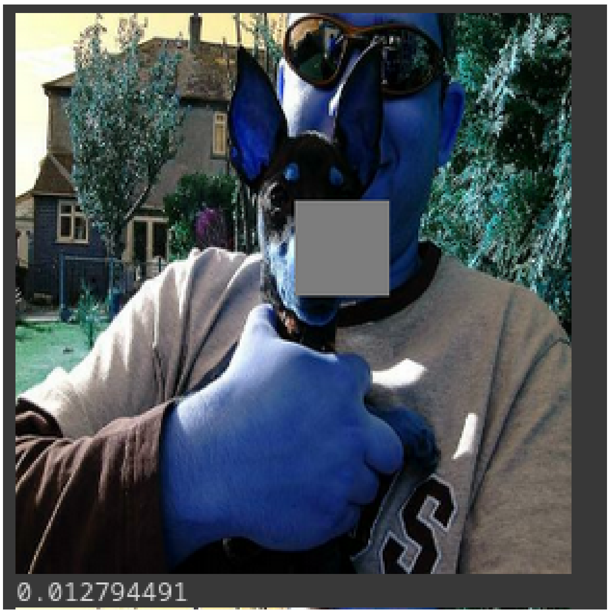
- Around 83% validation accuracy.
- On testset it evaluates to 83% also which is 10% better than my best model.

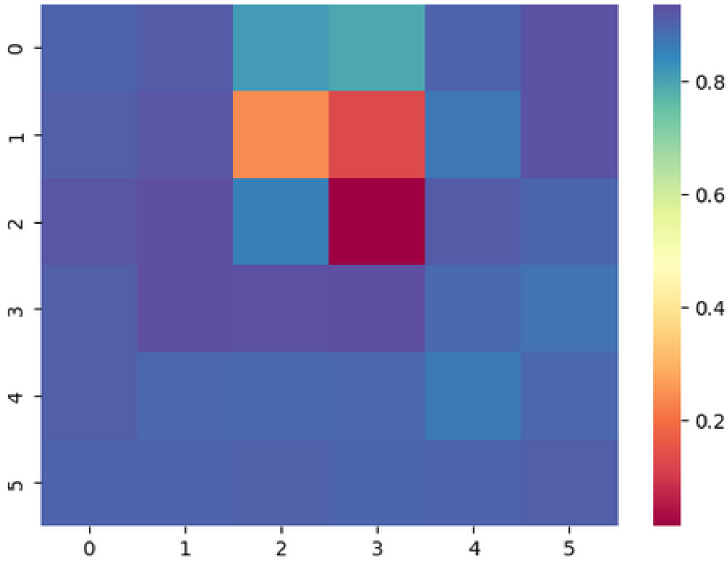
Problem 3:

- I made a function which receives the test image along with the dimensions of the occluder and slides it over the image.
- The function returns a list of images which are predicted by a pre-trained ResNet V2 model.
- I resized the test images so they can fit the model input layer with the same concept as in problem 2.
- Then I looped through the images and got predictions for each class.
- After that, I got only the probabilities of the correct class (toy terrier) from the predictions list.
- Below is a sample of the occluder sliding through the image with the prediction probability of the correct class “toy terrier”.

- 





- It is obvious that when we occlude the dog's face, the probability for the correct class reduces significantly.
- I created the below map to show the positions of the occluder with the probability of the model to classify the image correctly at that position.
- 

Colab notebook: omer_CV_Assignment_1.ipynb