Task 1. Multiple Pooling in Convolutional Neural Networks

Application used: Google Colab

Dataset : Subset of Kaggle Cat-Dog Data set of 10000 images.

Pooling used: Maximum and range Pooling

Model:

The data set is made into test and train set with cat and dog directories for both.
Then the input images are passed through image generator for prepossessing and making then to uniform shape of 64X64 pixels.
The processed images are passed into the model.
Model Architecture consist of a convolution layer followed by a Max Pooling layer. The convolution layer is used to create minimum pooling layer, which in turn is subtracted from a max pool layer to obtain a range pooling layer. The max pooling layer and range pooling layers are concatenated to get the final pooling layer. The output of the concatenated layer is used for the input of next convolution layer and the rest. 3 such layers are formed. Then its flattened and passed through a relu activation and 2 dense functions.
The Architecture of the model is shown below.

```
Model: "model"
__________________________________________________________________________________
_____
Layer (type)                    Output Shape         Param #
Connected to
==================================================================================
==========================
input_1 (InputLayer)            [(None, 64, 64, 3)]  0
__________________________________________________________________________________
_____
conv2d (Conv2D)                 (None, 64, 64, 32)   896
input_1[0][0]
__________________________________________________________________________________
_____
tf.math.negative (TFOpLambda)   (None, 64, 64, 32)   0
conv2d[0][0]
__________________________________________________________________________________
_____
max_pooling2d_1 (MaxPooling2D)  (None, 32, 32, 32)   0
tf.math.negative[0][0]
__________________________________________________________________________________
_____
max_pooling2d (MaxPooling2D)    (None, 32, 32, 32)   0
conv2d[0][0]
__________________________________________________________________________________
_____
tf.math.negative_1 (TFOpLambda) (None, 32, 32, 32)   0
max_pooling2d_1[0][0]
__________________________________________________________________________________
_____
```

```
tf.math.subtract (TFOpLambda)   (None, 32, 32, 32)   0
max_pooling2d[0][0]

tf.math.negative_1[0][0]
__________________________________________________________________________________________
concatenate (Concatenate)       (None, 32, 32, 64)   0
max_pooling2d[0][0]

tf.math.subtract[0][0]
__________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 32, 32, 32)   18464
concatenate[0][0]
__________________________________________________________________________________________
tf.math.negative_2 (TFOpLambda) (None, 32, 32, 32)   0
conv2d_1[0][0]
__________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 16, 16, 32)   0
tf.math.negative_2[0][0]
__________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 16, 16, 32)   0
conv2d_1[0][0]
__________________________________________________________________________________________
tf.math.negative_3 (TFOpLambda) (None, 16, 16, 32)   0
max_pooling2d_3[0][0]
__________________________________________________________________________________________
tf.math.subtract_1 (TFOpLambda) (None, 16, 16, 32)   0
max_pooling2d_2[0][0]

tf.math.negative_3[0][0]
__________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 16, 16, 64)   0
max_pooling2d_2[0][0]

tf.math.subtract_1[0][0]
__________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 16, 16, 32)   18464
concatenate_1[0][0]
__________________________________________________________________________________________
tf.math.negative_4 (TFOpLambda) (None, 16, 16, 32)   0
conv2d_2[0][0]
__________________________________________________________________________________________
max_pooling2d_5 (MaxPooling2D)  (None, 8, 8, 32)     0
tf.math.negative_4[0][0]
__________________________________________________________________________________________
max_pooling2d_4 (MaxPooling2D)  (None, 8, 8, 32)     0
conv2d_2[0][0]
__________________________________________________________________________________________
tf.math.negative_5 (TFOpLambda) (None, 8, 8, 32)     0
max_pooling2d_5[0][0]
__________________________________________________________________________________________
tf.math.subtract_2 (TFOpLambda) (None, 8, 8, 32)     0
max_pooling2d_4[0][0]

tf.math.negative_5[0][0]
__________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 8, 8, 64)     0
max_pooling2d_4[0][0]
```

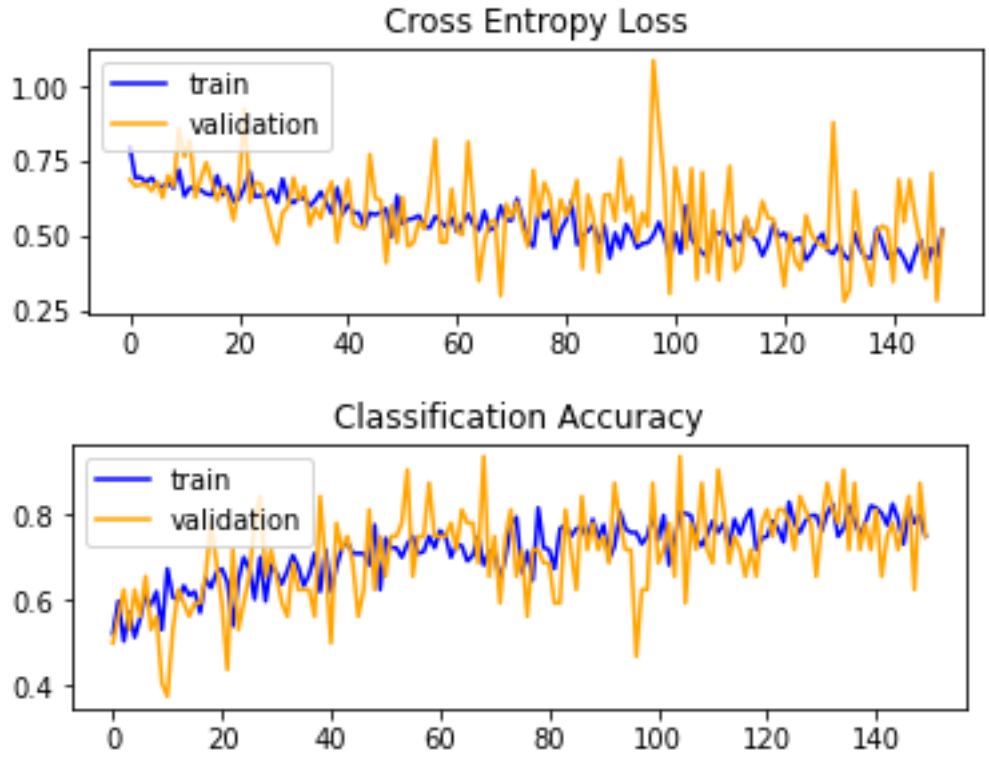
```
tf.math.subtract_2[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 4096)         0
concatenate_2[0][0]
__________________________________________________________________________________________________
re_lu (ReLU)                    (None, 4096)         0
flatten[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 256)          1048832
re_lu[0][0]
__________________________________________________________________________________________________
output (Dense)                  (None, 1)            257
dense[0][0]
==================================================================================================
Total params: 1,086,913
Trainable params: 1,086,913
Non-trainable params: 0
```

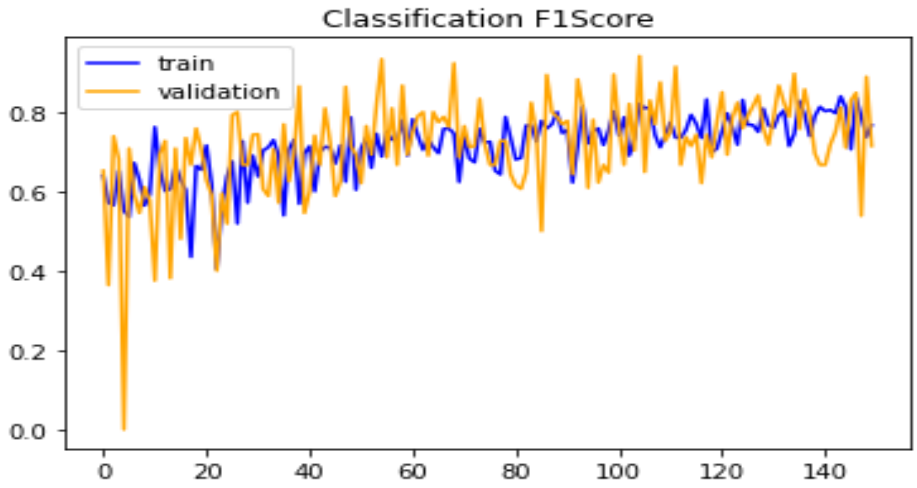
Optimizer: ADAM

Loss Function: Binary Crossentropy

No of Epoch: 150

Results:





The training plots are more stable compared to the test plots. This can be observed as higher variation for validation when compared to training data.

On observation the average of both training losses and validation losses are coming down. Similarly the average accuracy and f1 scores for both the sets are going up.

Average Training set Accuracy:0.717

Average Test set Accuracy:0.706

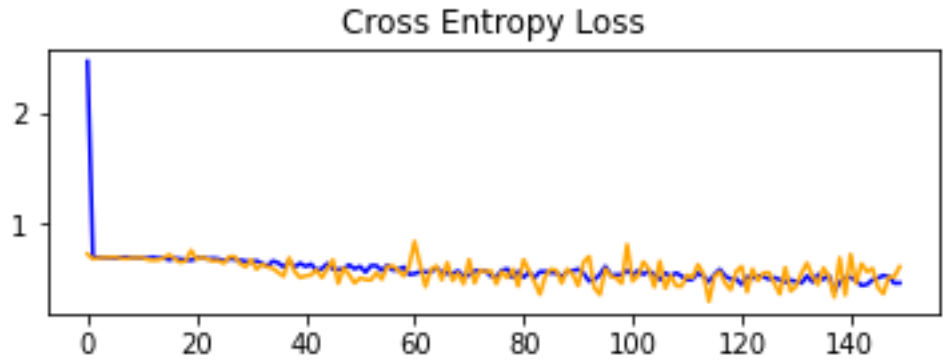
Average Training set F1 Score:0.71

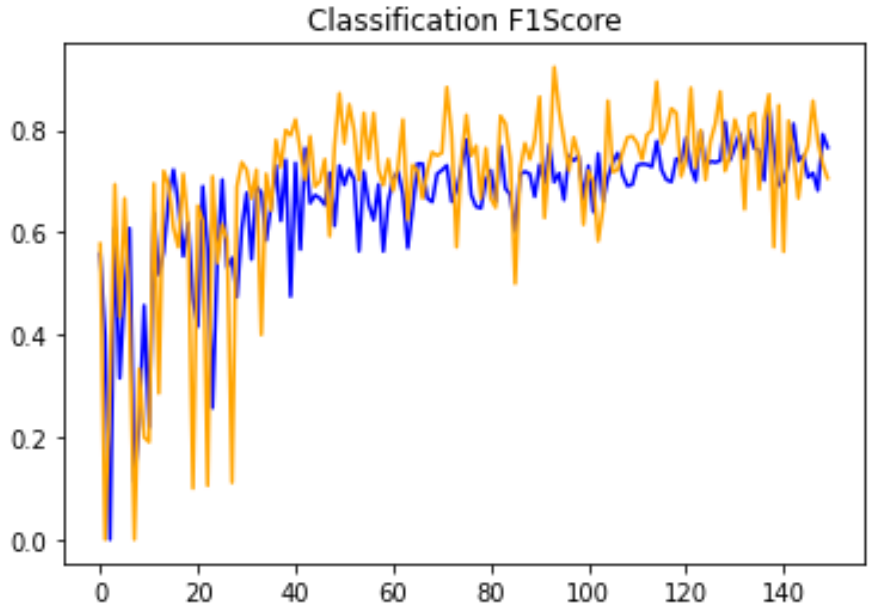
Average Test set F1 Score:0.71

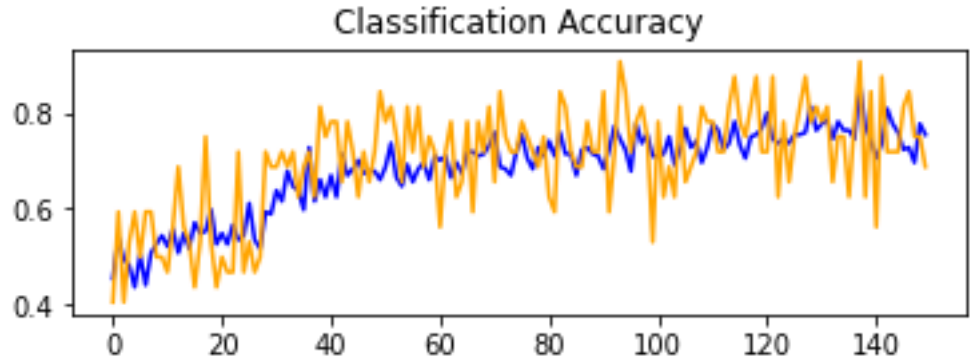
Comparison:

The model is compared with a regular 3 layer convolution model using max pooling.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 64, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 128)               1048704
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 1,142,081
Trainable params: 1,142,081
Non-trainable params: 0
_________________________________________________________________
```







On observation both the models looks similar. Except the multiple pooling model has picked a slight advantage in accuracy and f1 score.

Average Training set Accuracy:0.696

Average Test set Accuracy:0.697

Average Training set F1 Score:0.695

Average Test set F1 Score:0.701