Deep Learning

Assignment 4

Binary Classification

Abstract

To perform binary classification on the Kaggle cats and dogs dataset by convolution neural networks and to tune the model. Finally by transfer learning, the VGG16 convent model is loaded and evaluated. It is also hyper tuned by unfreezing the last layer and checked.

Problem Statement

The Kaggle dataset has 2 classes namely cats and dogs. Each class has 12500 images. For the purpose of creating a model that can give accuracy on lower datasets only 2000 images are chosen. These datasets are split into training, test and validation datasets. A convolution model is then built and ran on it. It is then hyper tuned. The filters and the convent layers are visualized. Finally the method of transfer learning is performed by loading the VGG16 convent.

Proposed Solution

Data preprocessing

The dataset is chosen and preprocessed. This process includes converting the images to numerical forms, splitting the dataset into test, train and validation parts.

Model layers

The convent layers are used in the model as specified in the question. The layers include:

Conv2D layer

A conv2D layer is a convolution layer that includes the number of filters and the dimensions of the filter and an activation function. In each convolution layer all the filters with the given dimensions are convoluted with the images to extract the features of the image. By this means more of features are extracted with less number of parameters.

MaxPool2D

The aim of this layer is to reduce the dimensionality of the input and extract features at the same time. For a given dimensions, it chooses the image and extracts features, combines them together. By this way it reduces the coefficients of the models.

Building the model

A convolution network is built with three layers of convolution and three layers of pooling. The number of filters in each layer and the dimensions are chosen randomly to give the best accuracy. Finally a fully connected model with one dense layer and an output layer with a softmax activation function is used.

Activations used in the model layer

Relu activation

Relu activation function is the most commonly used activation function. This is because it converges all positive values to one and all negative values to zero. Thus it is found more optimizing than the other activations.

Sigmoid activation

Sigmoid activation is used on the final dense layer to give maximum accuracy. It produces a large value for larger inputs and a smaller value for smaller inputs. It is the most commonly used activation for binary classification.

Optimizer used in the model

Rmsprop – It is a gradient based optimization technique to optimize the neural network model. It decreased the gradient that increases to stop from exploding and increases the gradient that decreases to prevent from vanishing.

Loss used in the model

Binary cross entropy – This loss is similar to a cross entropy or a softmax function. It predicts the difference between the actual and predicted labels across all the instances.

Regularization

Regularization includes certain methods followed to prevent the data from overfitting and under fitting once the correct loss functions and optimizers are determined.

The techniques used are:

Dropout – After each iteration the weight of the input or the parameter is decayed or decreased to prevent overfitting.

Batch Normalization – In this case the output of each layer of the neural network is normalized in batches to give a scalable input to the following layers.

Evaluation metrics

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 36, 36, 64)        0
```

```
_________________________________________________________________
dropout_2 (Dropout)          (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               3211776
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 513
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

Fitting the model

```
Epoch 15/15
100/100 [==============================] - ETA: 2:10 - loss: 0.0891 - accuracy
: 0.93 - ETA: 2:05 - loss: 0.1282 - accuracy: 0.92 - ETA: 2:03 - loss: 0.1128
- accuracy: 0.93 - ETA: 2:01 - loss: 0.1020 - accuracy: 0.94 - ETA: 2:00 - los
s: 0.0869 - accuracy: 0.95 - ETA: 2:00 - loss: 0.0960 - accuracy: 0.95 - ETA:
1:57 - loss: 0.0906 - accuracy: 0.95 - ETA: 1:55 - loss: 0.0828 - accuracy: 0.
96 - ETA: 1:54 - loss: 0.0841 - accuracy: 0.96 - ETA: 1:53 - loss: 0.0806 - ac
curacy: 0.96 - ETA: 1:52 - loss: 0.0838 - accuracy: 0.95 - ETA: 1:52 - loss: 0
.0826 - accuracy: 0.95
```

Testing against test case

```
0.7574999928474426
```

Hyper tuning

Hyper tuning is done by adding dropout and data augmentation.

Fitting the model

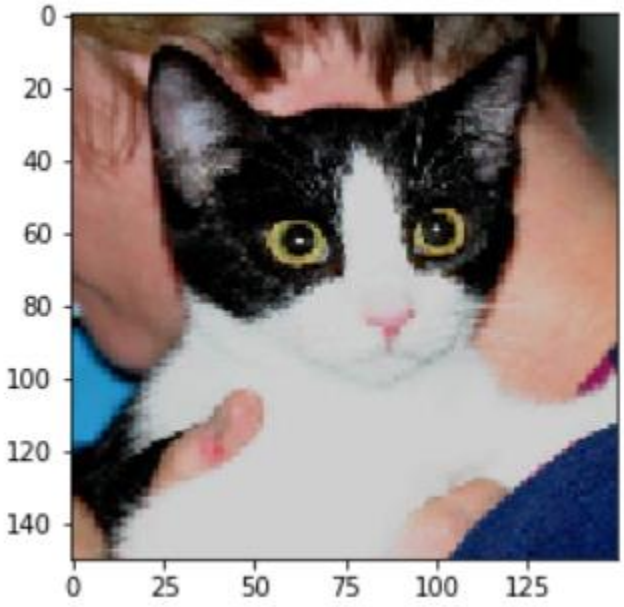
```
100/100 [==============================] - ETA: 1:04 - loss: 0.4829 - accu
racy: 0.71 - ETA: 1:04 - loss: 0.4380 - accuracy: 0.78 - ETA: 1:04 - loss:
0.4290 - accuracy: 0.78 - ETA: 1:03 - loss: 0.4193 - accuracy: 0.79 - ETA:
1:02 - loss: 0.4185 - accuracy: 0.78 - ETA: 1:02 - loss: 0.4358 - accuracy
: 0.78 - ETA: 1:01 - loss: 0.4792 - accuracy: 0.75 - ETA: 1:01 - loss: 0.4
801 - accuracy: 0.75 - ETA: 1:00 - loss: 0.4714 - accuracy: 0.76 - ETA: 59
s - loss: 0.4637 - accuracy: 0.7750 - ETA: 59s - loss: 0.4587 - accuracy:
0.778 - ETA: 58s - loss: 0.4489 - accuracy: 0.778 - ETA: 57s - loss: 0.455
8 - accuracy: 0.771 - ETA: 57s - loss: 0.4493 - accuracy: 0.776 - ETA: 56s
- loss: 0.4630 - accuracy: 0.770 - ETA: 55s - loss: 0.4736 - accuracy: 0.7
67 - ETA: 55s - loss: 0.4749 - accuracy: 0.773 - ETA: 55s - loss: 0.4734 -
accuracy: 0.770 - ETA: 54s - loss: 0.4733 - accuracy: 0.773 - ETA: 54s - l
oss: 0.4730 - accuracy: 0.768 - ETA: 53s - loss: 0.4718 - accuracy: 0.769
- ETA: 53s - loss: 0.4652 - accuracy: 0.775
```

Testing against test case

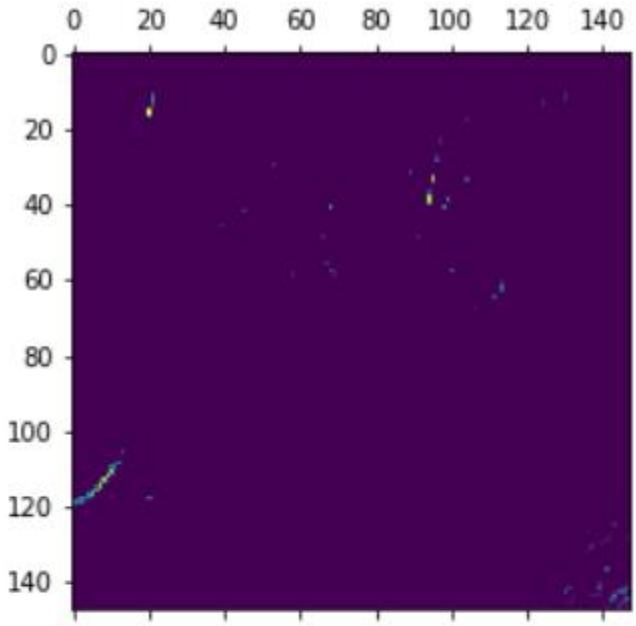
```
0.7649999856948853
```

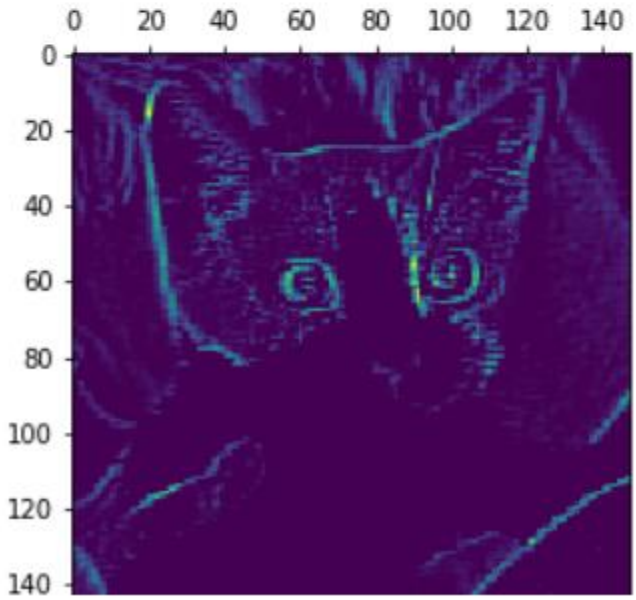
Visualization of the activations

Image

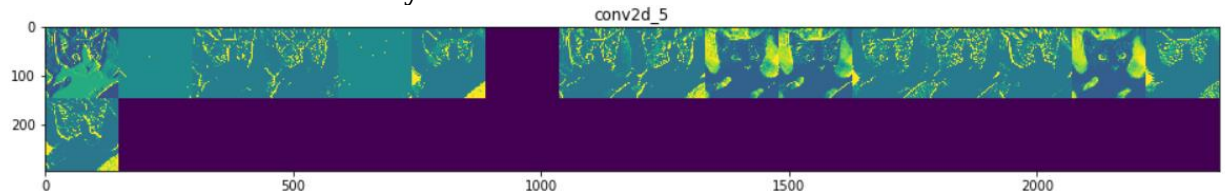


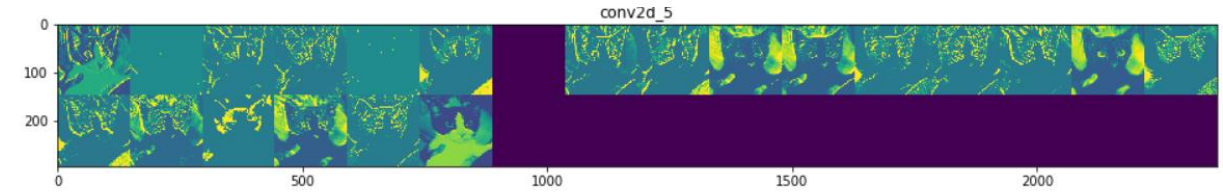
Visualizations

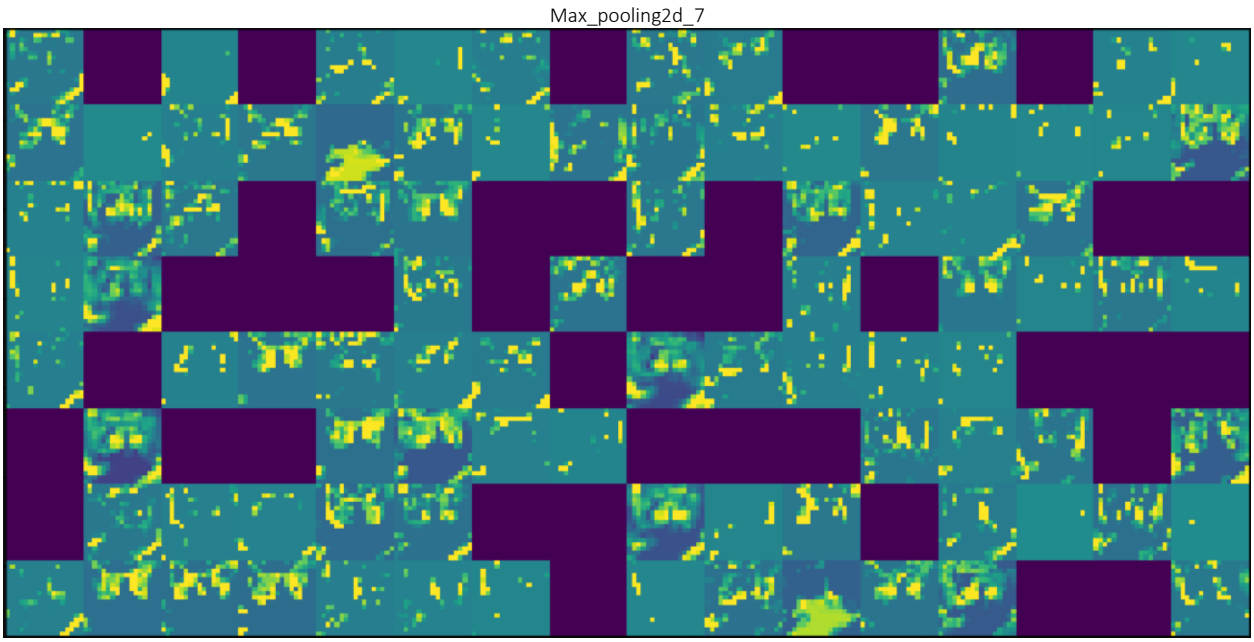




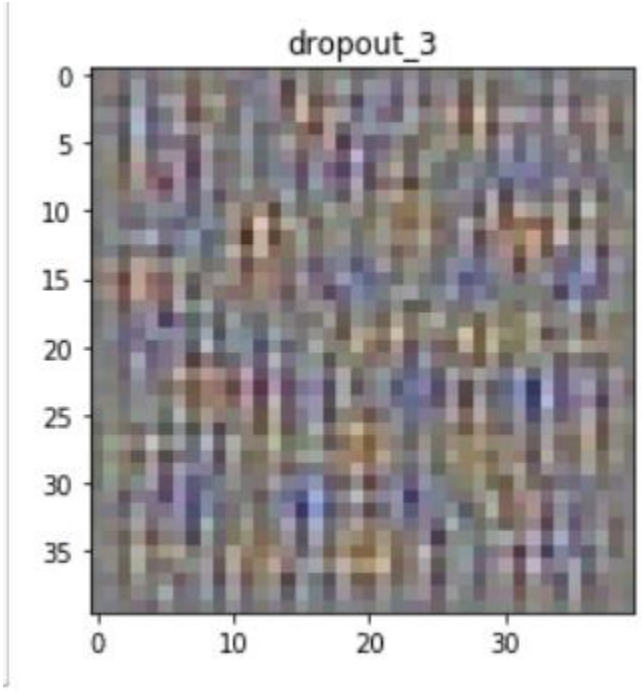
Visualizations for the entire layers

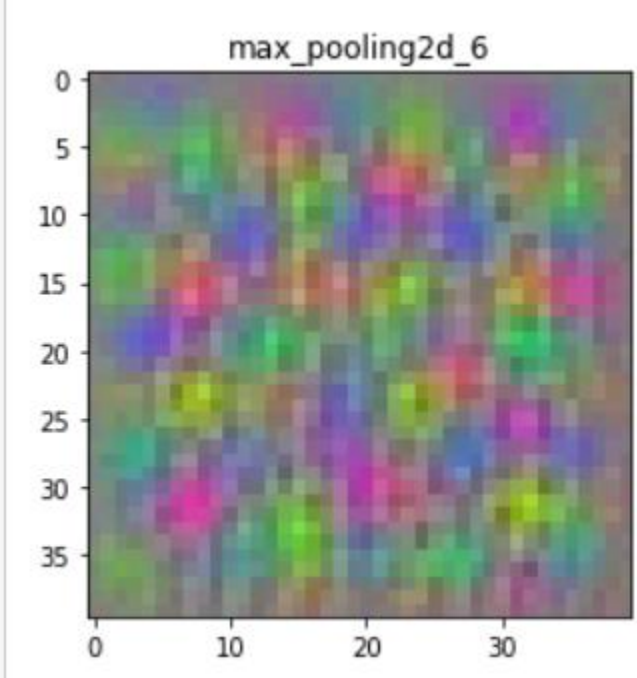






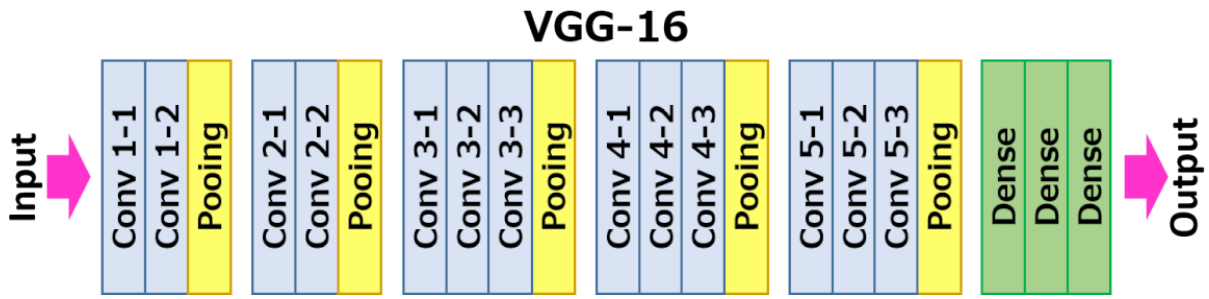
Visualizing the convent filters





VGG16 model:

VGG16 dataset is a pre trained model with its own convolution and dense layers. It was trained on millions of datasets and thus could give better results than a custom model that was tra ined only on the specific dataset.



From the model only the convolution model is taken and a custom dense layer is added.

Evaluation

```
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 3s 0us/step
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 150, 150, 3)       0
```

```
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Fitting the model

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_3 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_5 (Dense)              (None, 256)               2097408
```

```
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 16,812,353
Non-trainable params: 0
_________________________________________________________________
```

Freezing the conv_base

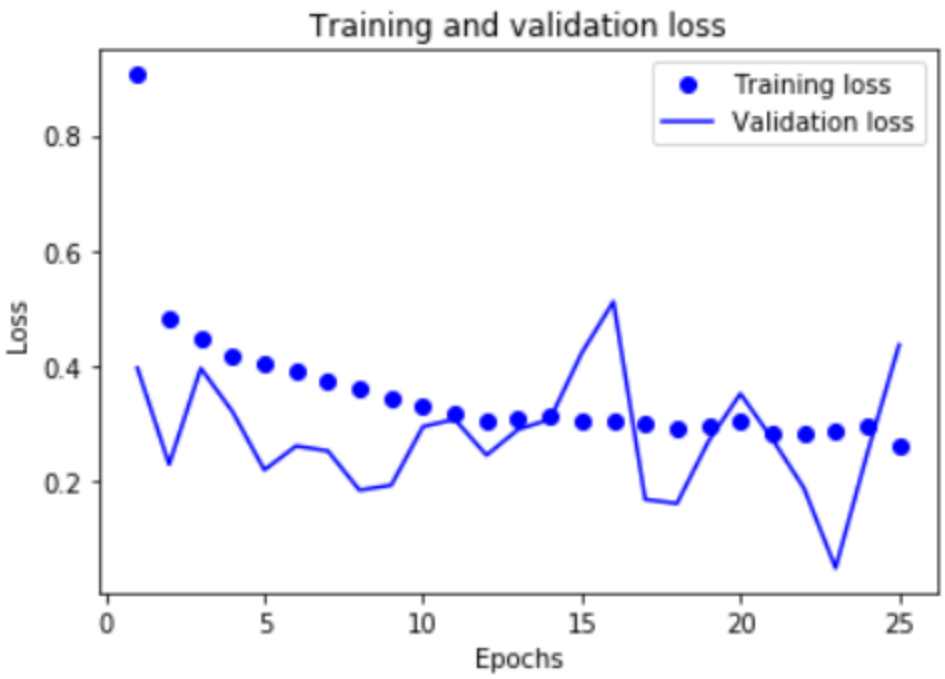
Conv base layers are frozen to protect the gradients of the model.

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_4 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_7 (Dense)              (None, 256)               2097408
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 2,097,665
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
Epoch 20/25
100/100 [==============================] - 207s 2s/step - loss: 0.3019 - a
ccuracy: 0.8684 - val_loss: 0.3514 - val_accuracy: 0.8905
Epoch 21/25
100/100 [==============================] - 205s 2s/step - loss: 0.2815 - a
ccuracy: 0.8806 - val_loss: 0.2730 - val_accuracy: 0.8913
Epoch 22/25
100/100 [==============================] - 206s 2s/step - loss: 0.2805 - a
ccuracy: 0.8763 - val_loss: 0.1874 - val_accuracy: 0.8639
Epoch 23/25
100/100 [==============================] - 205s 2s/step - loss: 0.2869 - a
ccuracy: 0.8788 - val_loss: 0.0486 - val_accuracy: 0.8730
Epoch 24/25
100/100 [==============================] - 206s 2s/step - loss: 0.2961 - a
ccuracy: 0.8716 - val_loss: 0.2479 - val_accuracy: 0.8848
Epoch 25/25
100/100 [==============================] - 205s 2s/step - loss: 0.2612 - a
ccuracy: 0.8819 - val_loss: 0.4360 - val_accuracy: 0.8854
```

Testing against test module

```
0.9225000143051147
```



Hyper tuning

The model can be tuned by unfreezing only the last layer of the conv_base. This is because the last layer extracts more features and is more specific.

Evaluation

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_5 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_9 (Dense)              (None, 256)               2097408
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 9,177,089
Non-trainable params: 7,635,264
_________________________________________________________________
```

Testing against test model

```
0.9525000143051147
```

Conclusion

The convolution model gave a less accuracy. This could be because of smaller dataset. Thus the dataset was augmented and ran to give more accuracy. Finally when the VGG16 model

was added, the accuracy raised more and gave better results. Finally when the model was hyper tuned, it gave even better results.

Implementation details

Implemented using anaconda and jupyter.

The issues included

1. Large datasets took a longer time.
2. Time consumption due to evaluation against all models.
3. The code was ran on a CPU that consumed more time.
4. Not every model was saved because the models were misinterpreted by the jupyter kernel. Thus only the best model code is provided.
5. The kaggle code was ran in two different systems and thus different code is attached.

References:

1. Code given by the professor