Cat

Vs

Dog

Classifier

By

AKASH SHARMA

Contact No. and Mail- Akashsharma8875@gmail.com, 8949122676

Contents

[Problem Statement 4](#_Toc80927733)

[Dataset Details-Understanding and Observations 4](#_Toc80927734)

[Observation-1-Basic CNN Model 5](#_Toc80927735)

[Observation-2-Transfer learning procedure 9](#_Toc80927736)

[Observations-3-Using Encoder output as features 18](#_Toc80927737)

[Detailed Comparison of All three concepts 24](#_Toc80927738)

[Appendix 25](#_Toc80927739)

[**Figure 1:Visualization of Dataset images of cats and dogs** 5](#_Toc80927547)

[**Figure 2:Data pre-processing for classification** 6](#_Toc80927548)

[**Figure 3:CNN Model Creation For feature selection** 7](#_Toc80927549)

[**Figure 4:CNN Training Procedure with EPOCHS** 8](#_Toc80927550)

[**Figure 5:Making A prediction By Observation-1 Model** 9](#_Toc80927551)

[**Figure 6:Dataset extraction with path allocation** 10](#_Toc80927552)

[**Figure 7:Single Plotted Image output** 11](#_Toc80927553)

[**Figure 8:Rescaled Picture of a randomly selected picture from Dataset** 12](#_Toc80927554)

[**Figure 9:Base Model Creation from the pre-trained MobileNet V2** 13](#_Toc80927555)

[**Figure 10:Feature Extraction and Creation by adding layers** 14](#_Toc80927556)

[**Figure 11:Model Compilation on a large dataset and accuracy-loss check at initial start** 15](#_Toc80927557)

[**Figure 12:Visialization of accuracy and loss comparison on both training and validation dataset** 16](#_Toc80927558)

[**Figure 13: Accuracy and Loss check after Re-Compilation** 17](#_Toc80927559)

[**Figure 14:Predicted Output with the Observation-2** 18](#_Toc80927560)

[**Figure 15:Classifier Creation for Image Dataset** 19](#_Toc80927561)

[**Figure 16:Image Dataset Extraction** 20](#_Toc80927562)

[**Figure 17:Checking a Random Image** 20](#_Toc80927563)

[**Figure 18:Misclassified Image of Cat as Dog** 21](#_Toc80927564)

[**Figure 19:Misclassified Image of Dog as Cat** 21](#_Toc80927565)

[**Figure 20:Layer output visualization for a single layer** 22](#_Toc80927566)

[**Figure 21:Feature Extraction by setting encoded feature in the filter in the second layer** 23](#_Toc80927567)

[**Figure 22:Cat or Dog prediction after output feature selection** 24](#_Toc80927568)

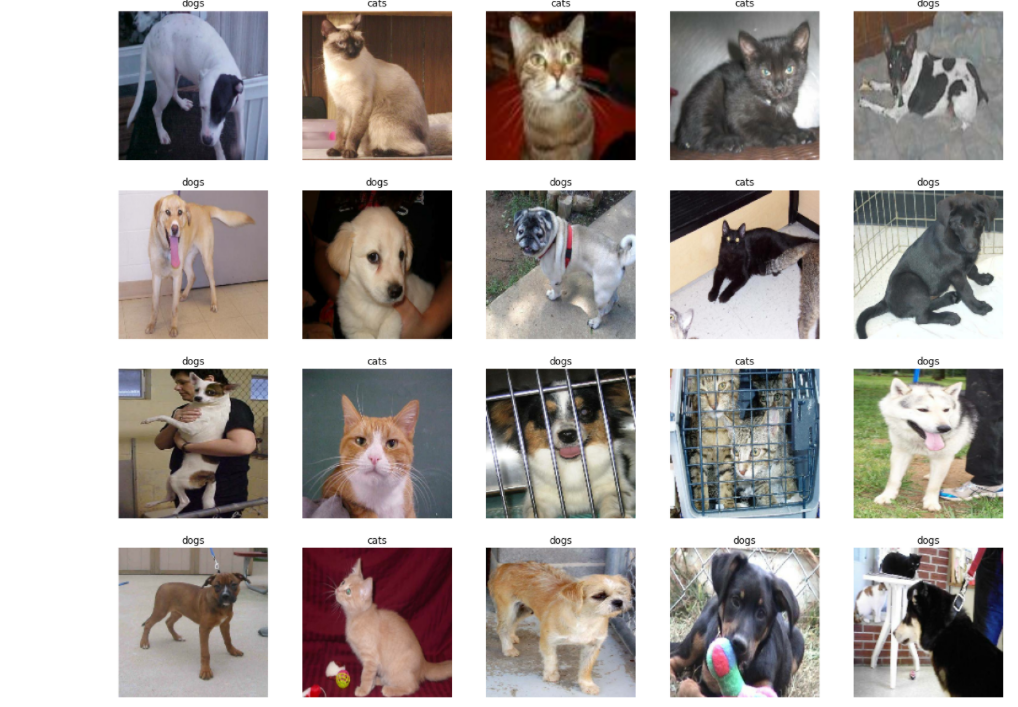
# Problem Statement

The Cat vs Dogs dataset is based on a computer vision dataset which consists of the classifying data of either dog and cat. The problem sounds simple as it is easily recognized and separated by CNN (Convolutional Neural Network) in Deep learning. The Practice has been solved by different data scientists but it can be used to practice, evaluate, and modify with the help of some newly introduced operations.

# Dataset Details-Understanding and Observations

The dataset is given here with the two categories of images. Training and test data, in which the training data consist of 20000 images that belong to 2 classes. The test data or it can be called validation data contains 5000 images belonging to two classes. The total number of validation batches is found at 157 of which the data test datasets are around 31.

|  |  |
| --- | --- |
| **Details** | **Size and Observations** |
| *Total number of images* | 25000 |
| *Image format* | .jpg |
| *Train Dataset* | 20000 |
| *Test Dataset* | 5000 |
| *Total number of Class* | 2 |
| *Name of Class* | Cat and Dog |



**Figure 1:Visualization of Dataset images of cats and dogs**

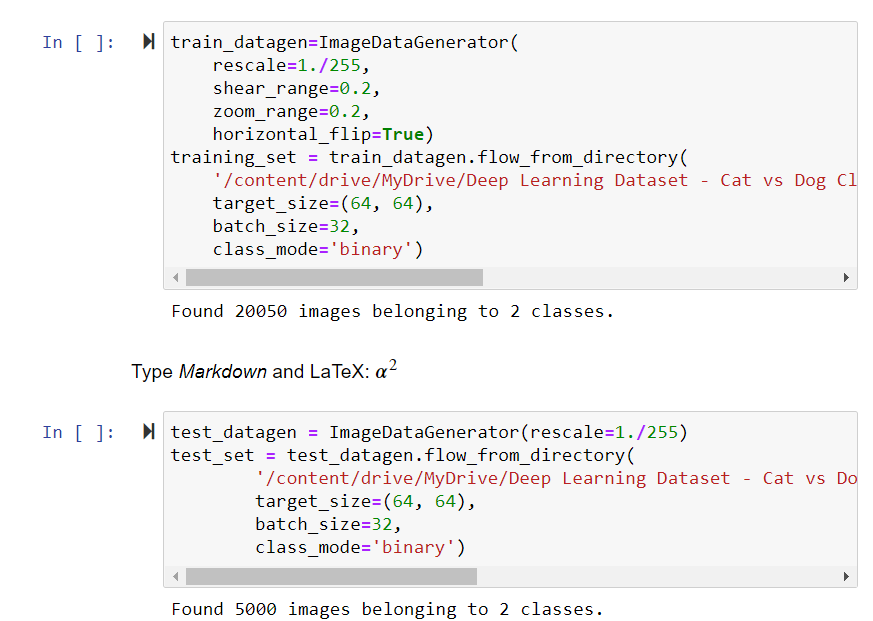
There are 3 kinds of observations that have been followed in this model preparation which is:

1. Basic CNN model
2. Transfer learning procedure
3. Using Encoder output as Features

# Observation-1-Basic CNN Model

This is a very exciting activity of a neural network where the neural connects one more extra layer of convolutional. It takes an image of a 3D network and adds this convolutional neural layer to this image to extract the feature of the image. The dataset has lots of images of cats and dogs which are going to be classified with a correct prediction of either a CAT and DOG. Here are the process steps are given below for creating a CNN model from scratch.

1. **Data pre-processing**- The training dataset was pre-processed with the help of multiple steps and it was done to avoid overfitting. The steps come under image augmentation. The image data generator class is used as a tool that processes and transforms according to the image dataset. the rescale was used around 1./255 which means each pixel comes between 0 to 255. The shear image range was set around 0.2 and also the zoom range.



**Figure 2:Data pre-processing for classification**

1. **Building the CNN**-This is the main architecture of the whole model. A CNN class was used to create this layer and a sequential class of Keras was called for CNN or fully connected layer.

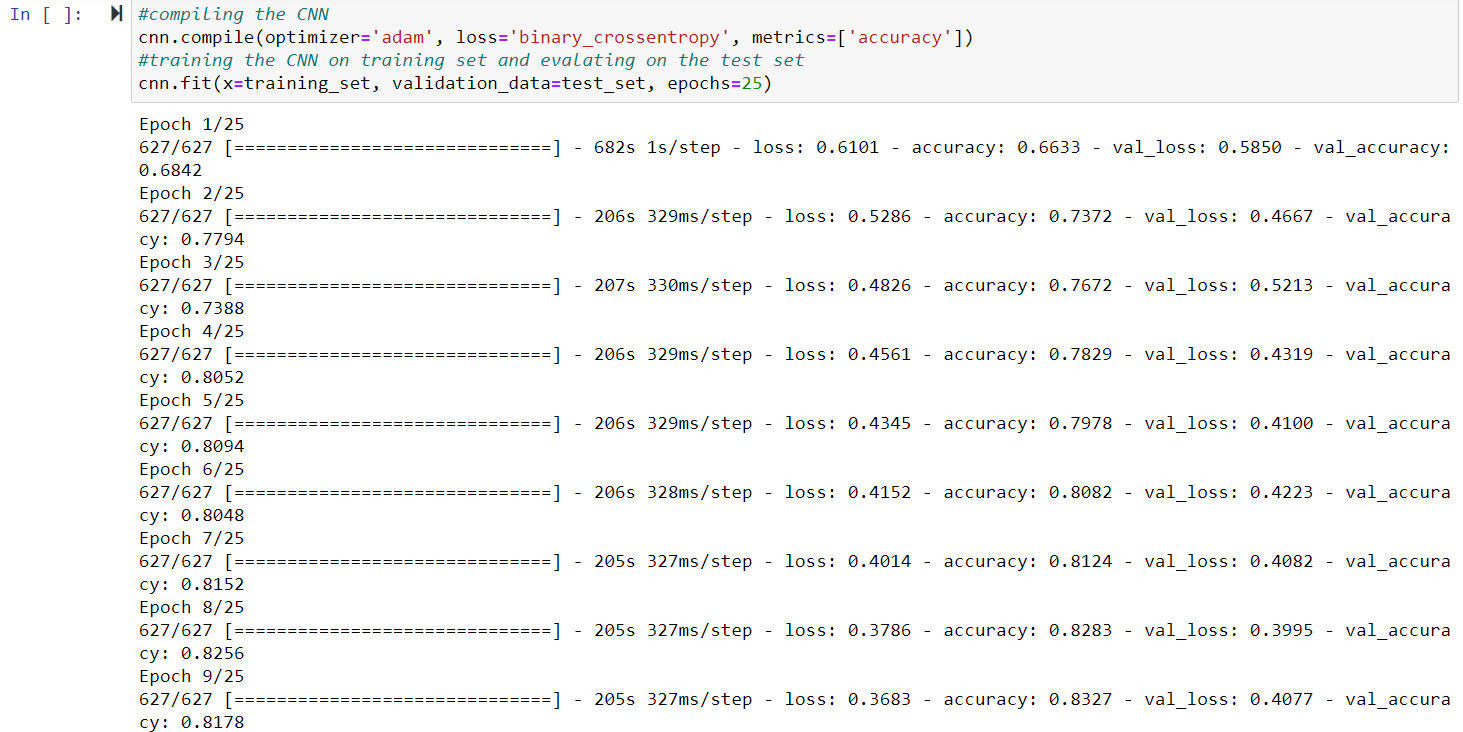
* **The CONVOLUTION**-The convolutional layer was generated by calling the CNN class and add it into the access of layers of the CONV2D class.  This layer generates a convolution kernel that is airstream through layers input which supports yield a tensor of outputs. Some arguments were set in this layer where the filter or feature detectors are chosen upon the classic process. It was set to 32. The kernel size was set to 3 and the activation function was taken relu (rectifier linear unit). Input shape was set around (64,64,3) which depends on the data pre-processing steps.
* **POOLING**-In the second step, a max-pooling layer was created and added. The maxpool2D class was used in this process. The maxpool2D class helps to select the maximum elements from the region in a feature map. These maps are covered by the filter. The pooling is used to pool feature maps. Pool size is set after the class implication on the image pixels. It gets shifted from one frame of an image pixel to another automatically. For this class, the argument was set which is pool size=2, strides=2.
* **FLATTENING**-A flatten class was added in CNN to reshape the tensor and to have the shape which is equal to the number of elements including the batch dimension.
* **FULL CONNECTION**-Dense class was added in the elements with the units of 128, and activation function of relu (rectifier linear unit). It is used to connect all the layers that were created.
* **OUTPUT LAYER**-A single neuron was added in the unit of each layer but the activation function was changed to sigmoid to optimize the result of image classification.

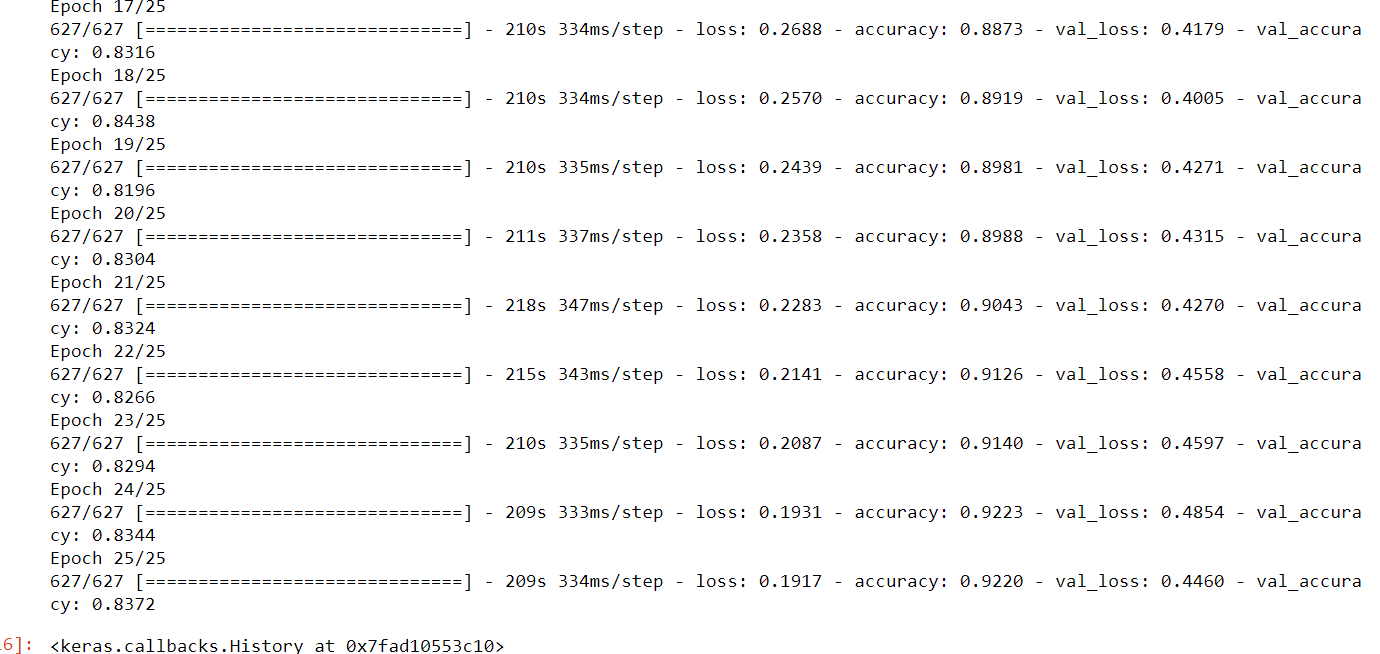


**Figure 3:CNN Model Creation For feature selection**

The same steps were followed to create a second convolutional layer but the input shape was not given to further layer as it was a renewal process.

1. **Training The** **CNN**-This process was divided into two steps in which the first steps were based on the compilation of CNN and the next is to train the CNN on the training set with an evaluation of the test set. The compile method was called in the CNN class. Some arguments that were given during this process and are, optimizer=Adam, loss=binary cross-entropy, and metrics=accuracy. The Adam optimizer was used to exponentially decaying the average of the past gradients. The binary cross-entropy was used to make the loss independent from the class which means the CNN output vector will not be affected for every CNN output and component value. Now the Second steps were to train the CNN by fitting the class. It is clearly shown in this figure. A total of 25 epochs were executed for perfect accuracy.





**Figure 4:CNN Training Procedure with EPOCHS**

1. **Making a Single Prediction**-It was found out with the previous step that the accuracy was around 0.92 and loss was 0.19 which is a far better result. Now the single image was processed to find out if it belongs to a category of cat or dog. It can be seen in the figure the randomly selected image was classified perfectly between cats and dogs.



**Figure 5:Making A prediction By Observation-1 Model**

# Observation-2-Transfer learning procedure

Transfer learning works upon a small premise where the data model is trained upon a large dataset and the extracted knowledge is transferred to a smaller dataset. the reason behind this process is that it freezes the layers for the networks and trains a few layers to make a prediction.

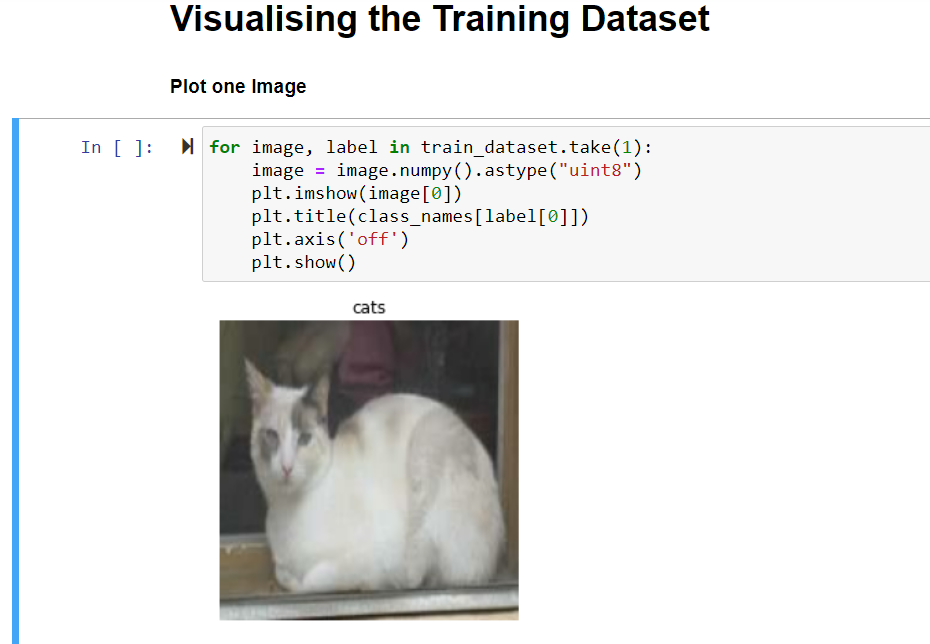
Here the second process for prediction was done upon the transfer learning. The dataset was extracted from the directory with the help of a module named image\_dataset\_from\_directory. The batch size was given the same as the above basic model around 32. Image size was set on the default (160,160) of both the train and test datasets.

1. **Creating the test dataset**-The validated dataset was created with the most count from this dataset. it showed that the validated dataset was separated according to the batches in which the 6 batches are used for the test and the rest of 26 are used for the validations.



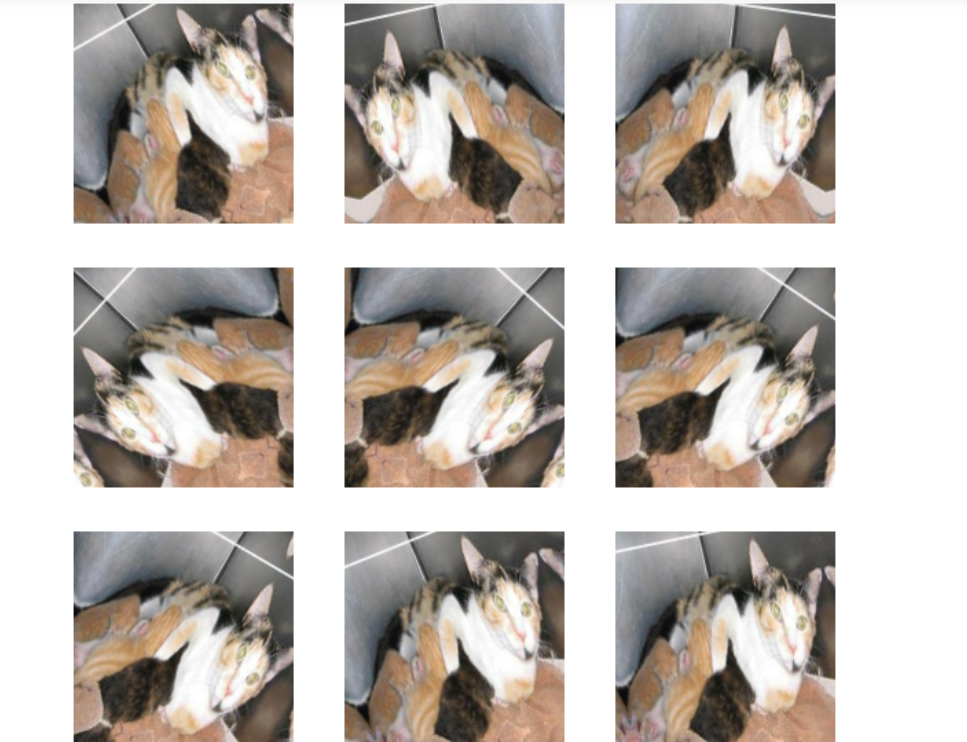
**Figure 6:Dataset extraction with path allocation**

1. **Visualization of the dataset**-The Dataset was visualized by plotting the single image shown in the image below. With the completion of visualization, the multiple training datasets were plotted in which some auto-classified images of cats and dogs are shown.
2. **Configuring the data**-Buffer Prefetching was used to load the images from the dataset with facing the input and output blockage. The Images were AUTOTUNED in this process and it is set upon the buffer size.
3. **Data Augmentation on Training Set**-It is used for some realistic transformations such as flipping, image rotation, and cropping. It helped in exposing the model and reduced the overfitting. These layers are found active during the training. When the model. the fit process starts then the model starts evaluating the inactive modes.



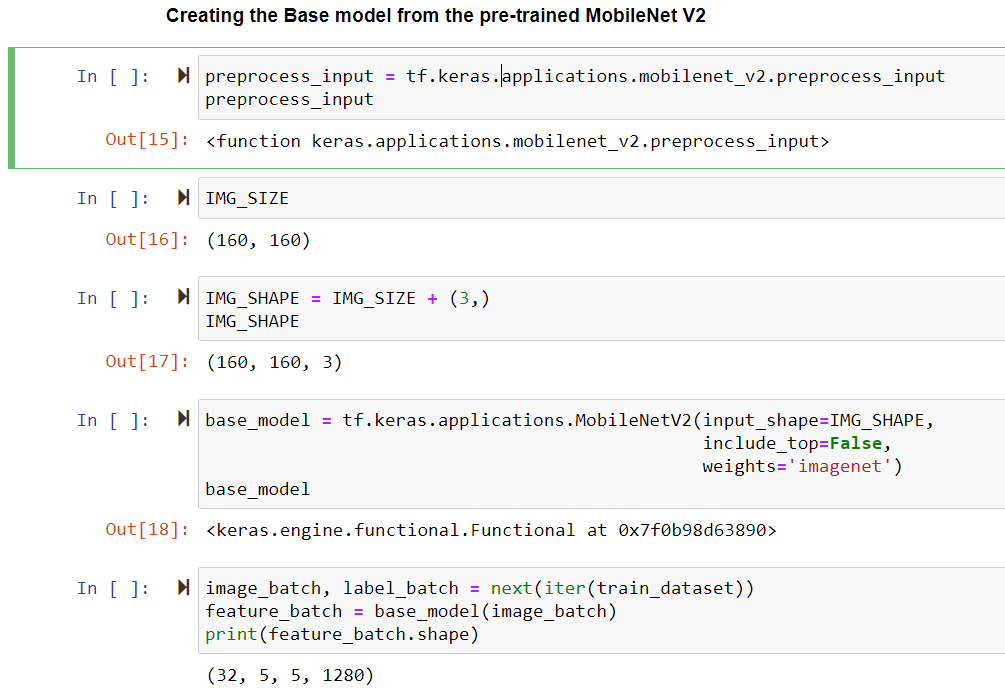
**Figure 7:Single Plotted Image output**

1. Plot one augmented image-here is the code flow with the output shown below in which the features are extracted.
2. Rescale the pixel values-Mobile netV2 model which was originally developed by Google was used. It pre-processes the NumPy or tensor array by encoding a batch of images. It returns a pixel value that is scaled between -1 to +1 (Sample-wise).



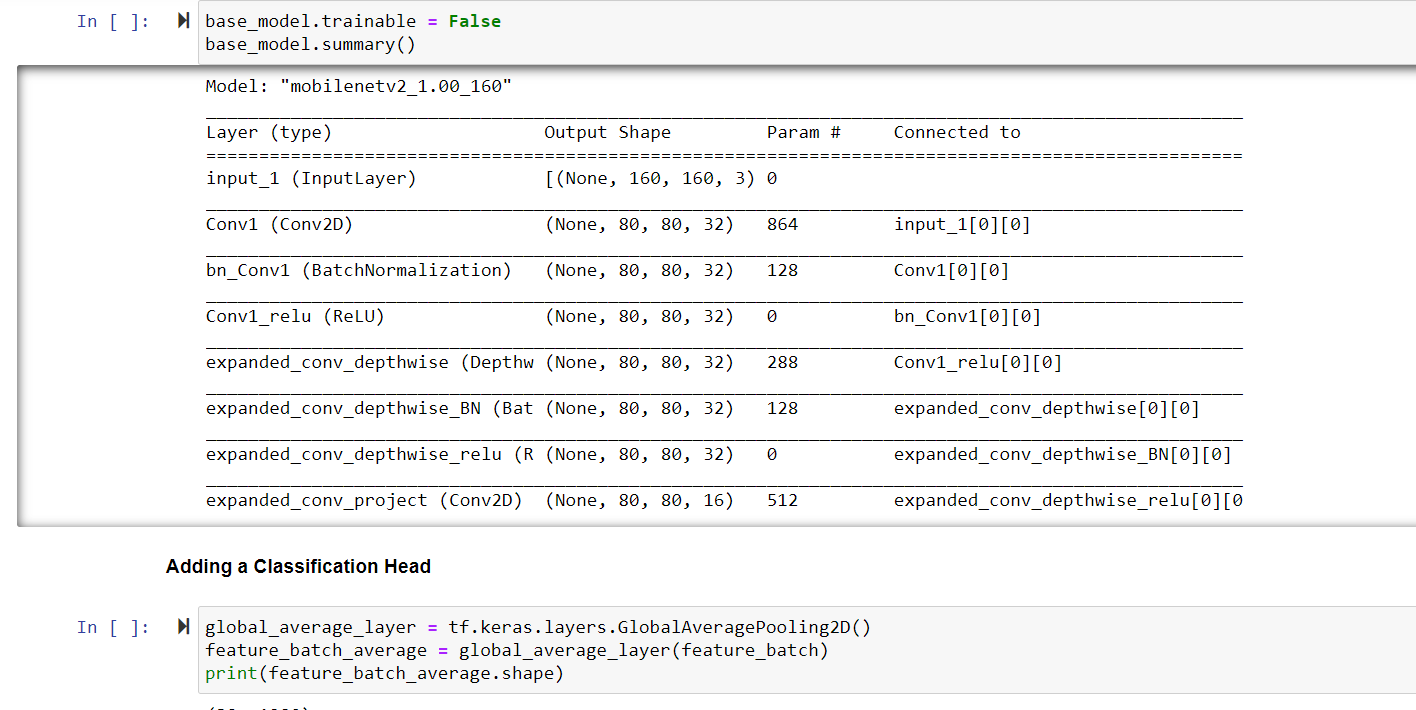
**Figure 8:Rescaled Picture of a randomly selected picture from Dataset**

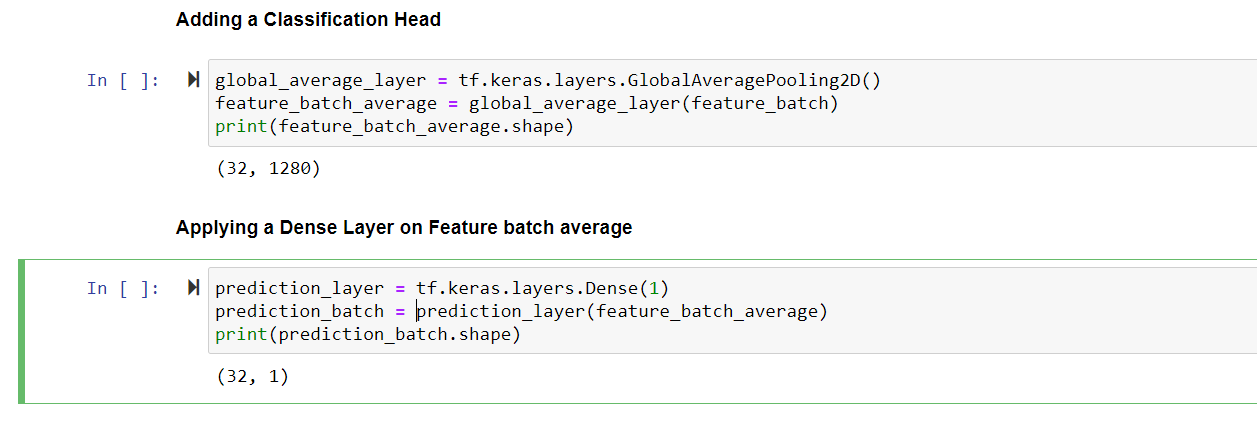
1. **Creating the base model from the pre-Trained model**-The processes are done upon the small datasets and feature generation advantage was taken on a larger dataset within the same domain. The process was done by adding a fully-connected classifier on top. In this process, the pre-Trained model was frozen and Classifier weights were updated during the training.



**Figure 9:Base Model Creation from the pre-trained MobileNet V2**

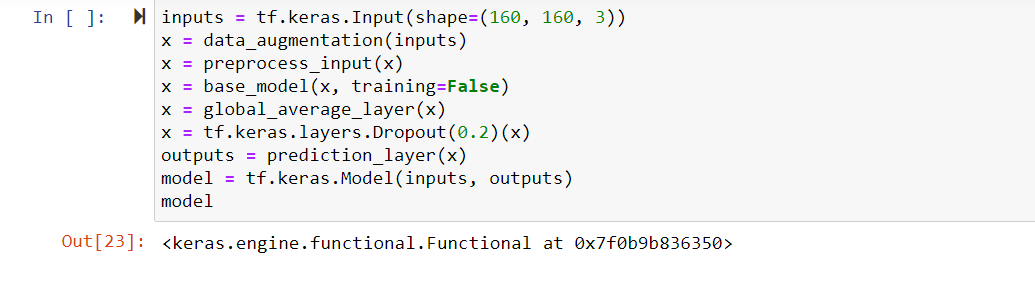
The feature extractor converted each image into a smaller size of the block of features (5\*5\*1280). The convolutional layer base gets frozen before the compilation of the model. As it was observed that the MobileNet V2 has many layers so setting up the entire list was flagged as FALSE. A classification Head was added which was used to generate from the block of features. As it has been that the average spatial location of features block is underuse so with the help of GlobalAveragePooling2D was used to convert these blocks of features in a single 1280-element vector per image. In the last step of creating a base model, a dense layer on the feature batch average was applied. It was done without using the activation function here. *Keras. layers. Dense* Layer was used to convert these features into a single prediction.

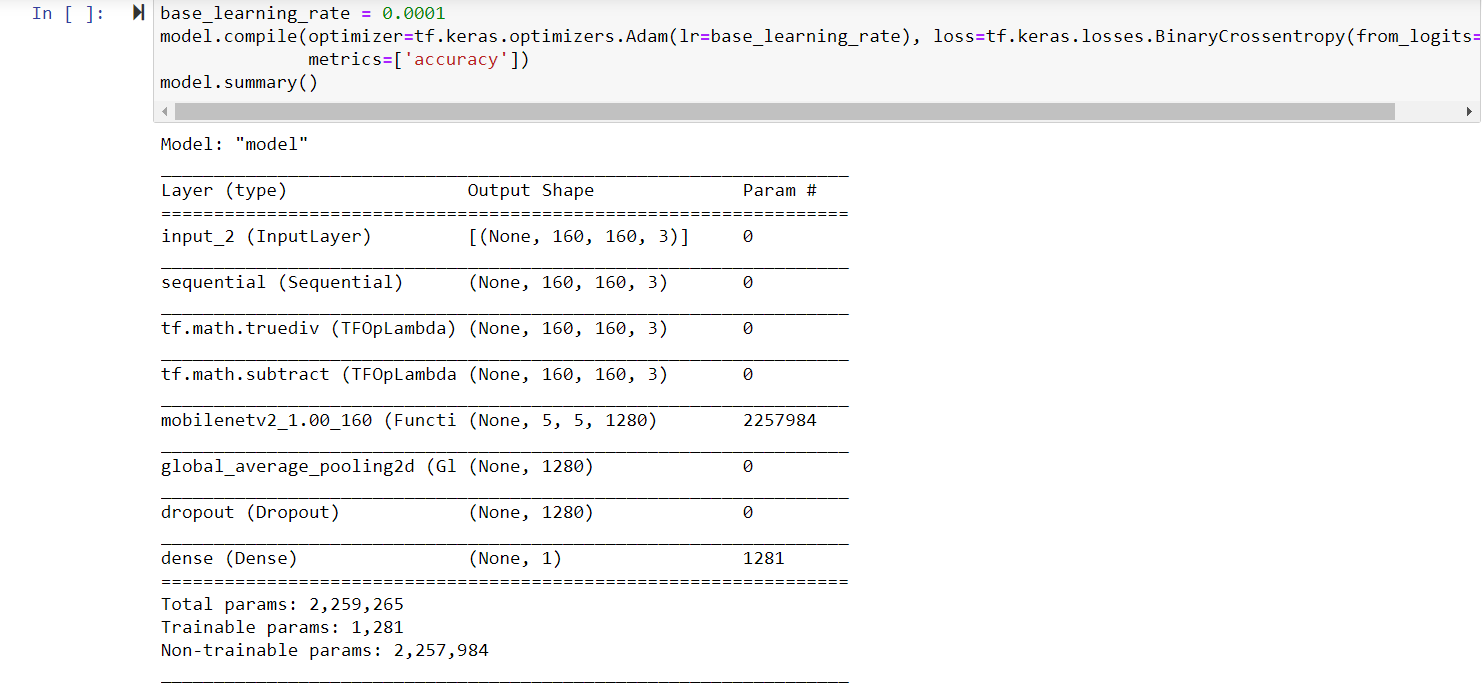


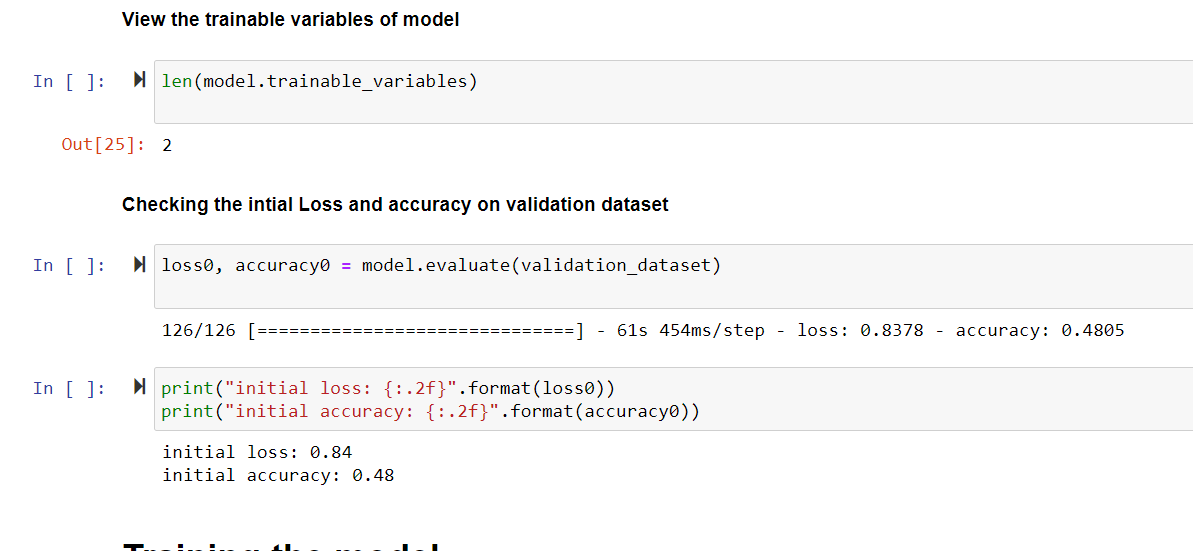


**Figure 10:Feature Extraction and Creation by adding layers**

1. **Building and compiling the model**-The model was built by inputting the created base model features and a dense layer was also added to get the functional engine working. The model was compiled over the binary cross-entropy. The loss with form\_logits was set to TRUE as the model offers a Linear Output. The model summary shows that there are 2.5M parameters are frozen but around 1.2K trainable parameters are still available in the Dense Layer. Initial accuracy and loss on the test datasets were found 0.48 and 0.83 respectively.

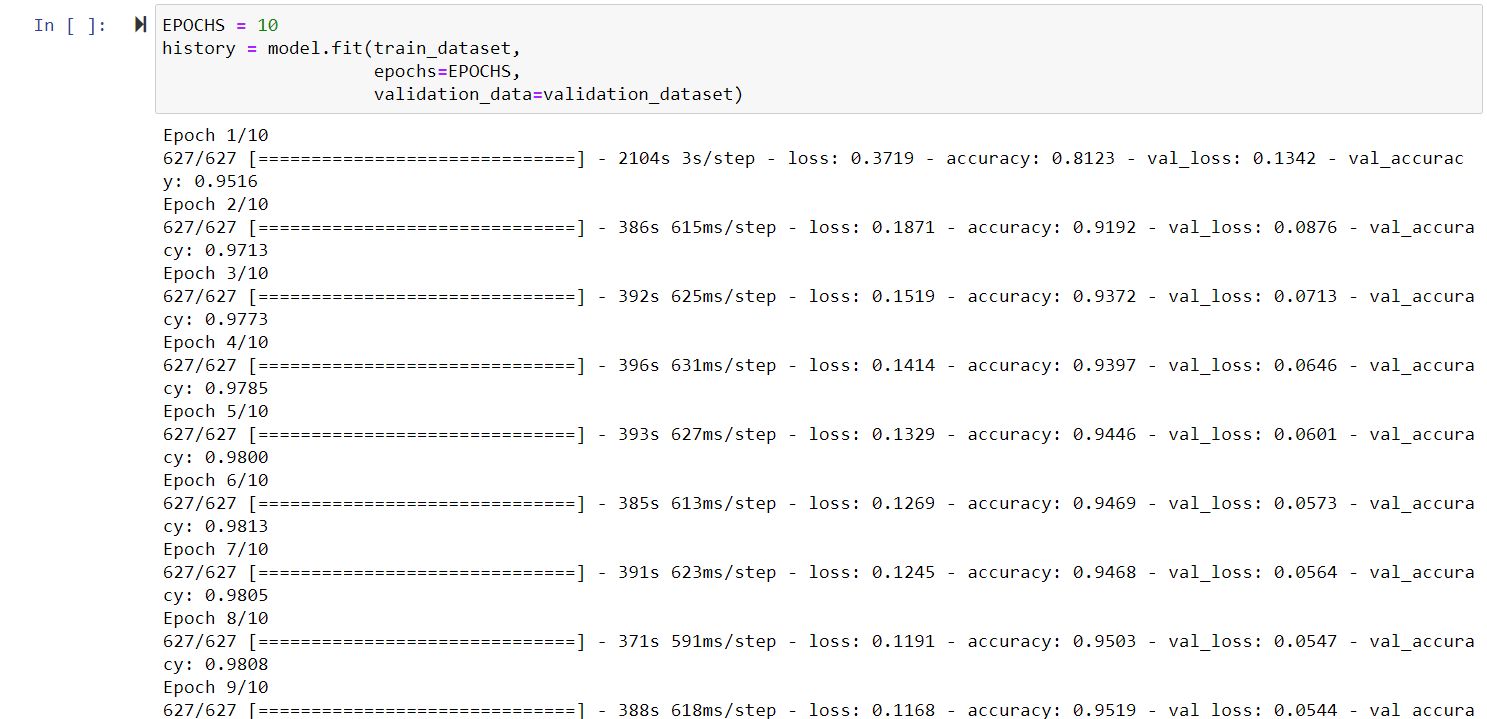


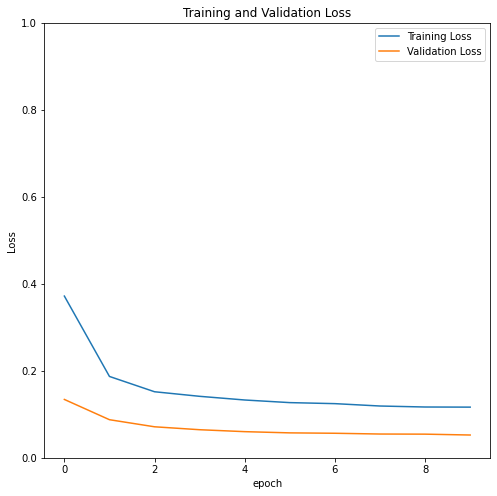
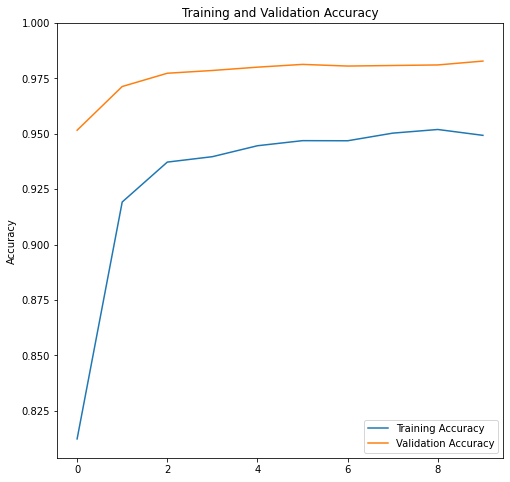




**Figure 11:Model Compilation on a large dataset and accuracy-loss check at initial start**

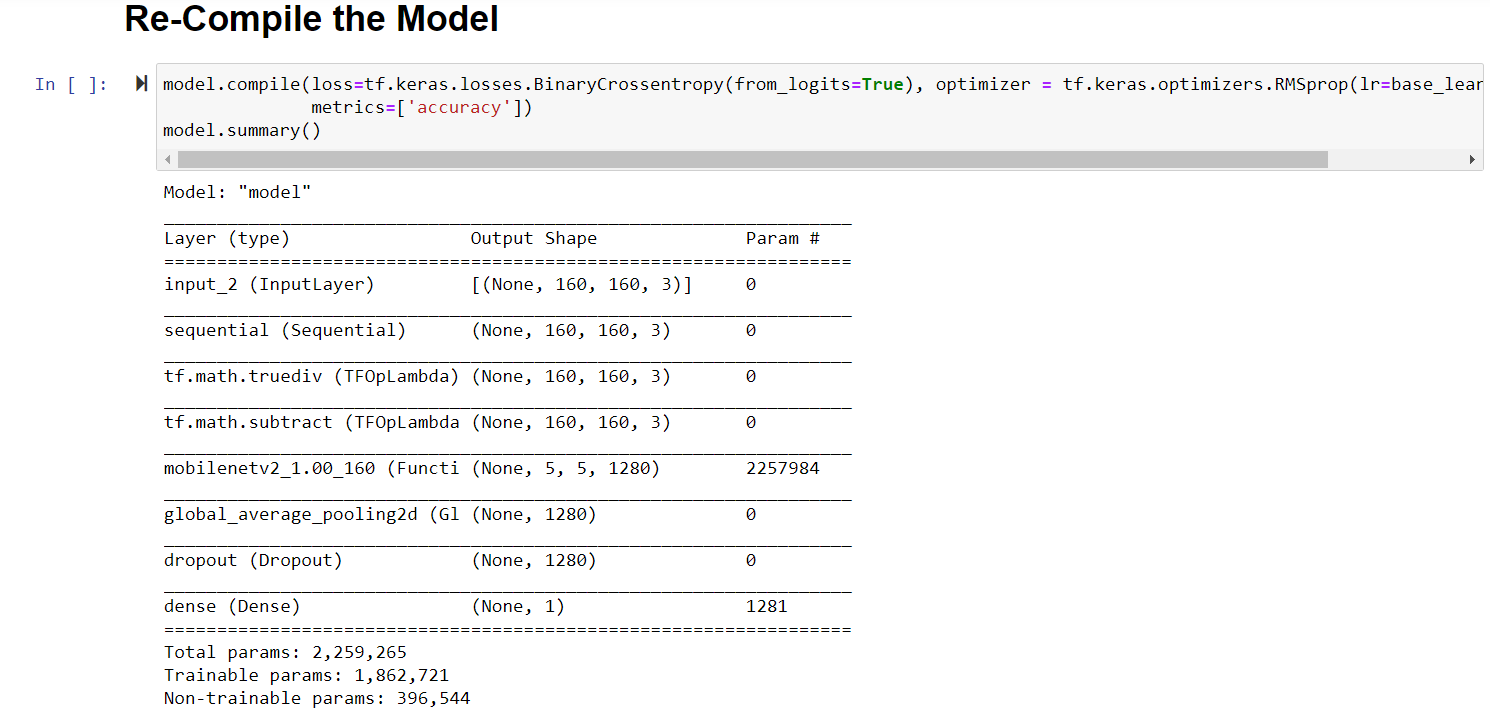
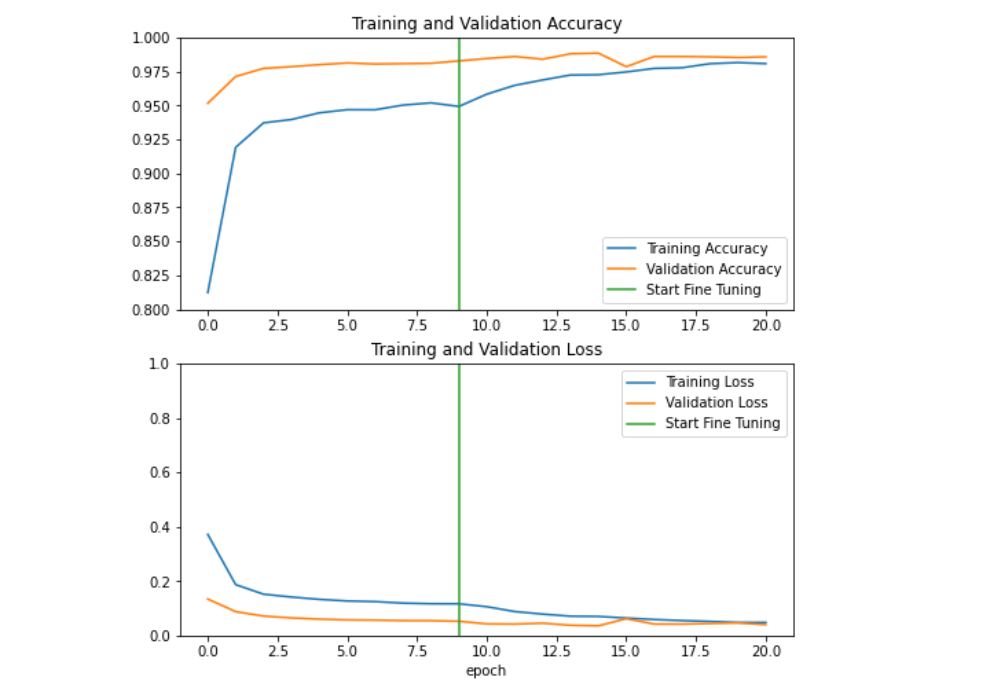
1. **Training the model**-The complete model was fit with the base model and total EPOCHS=10 was set. It is done upon validation or test dataset. here are some images are given below of the training process. The accuracy was achieved here around 0.94 and loss was 0.0524. some plots are shown for the validation and training accuracy and plots below.





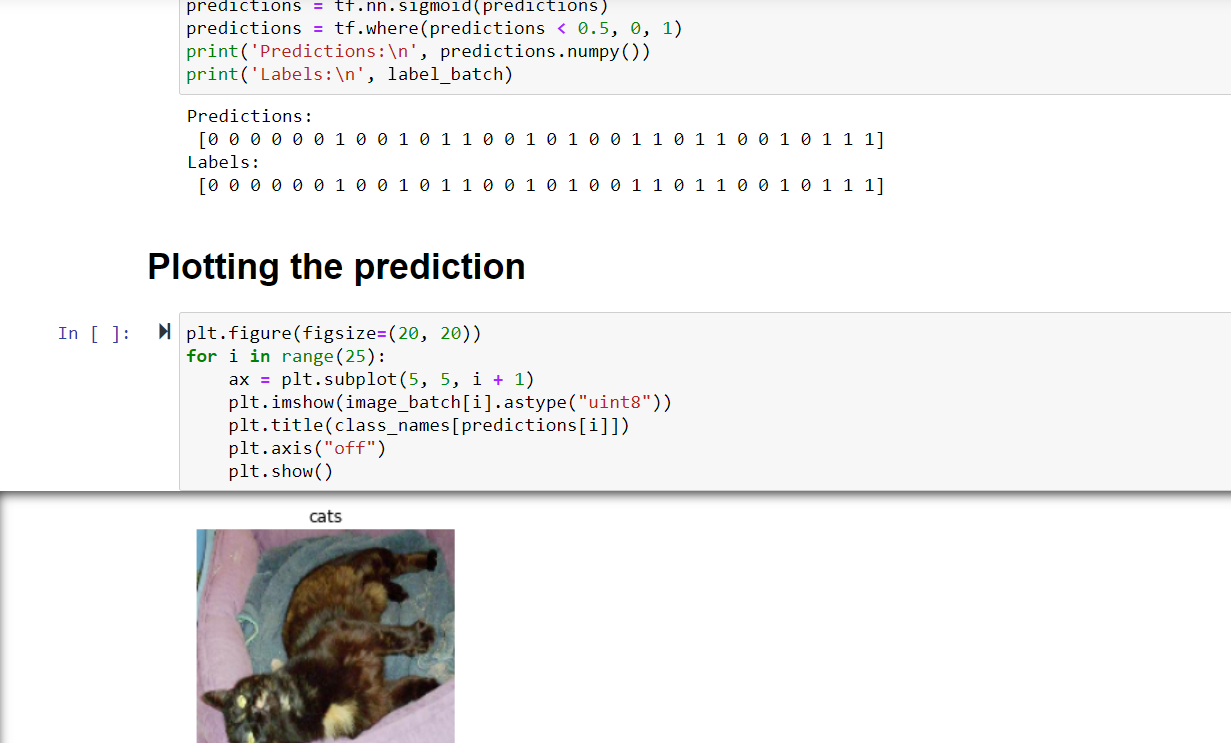
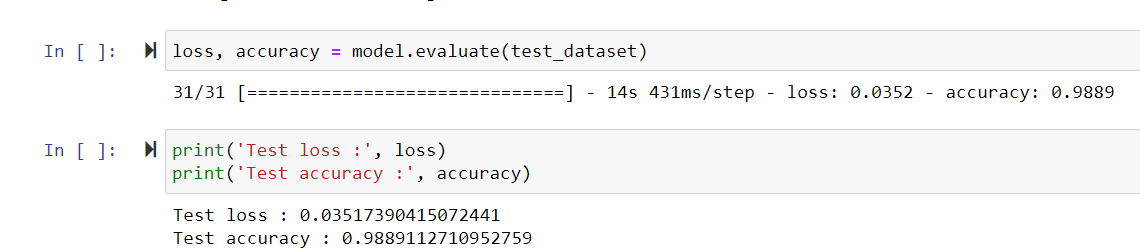
**Figure 12:Visialization of accuracy and loss comparison on both training and validation dataset**

1. **Fine-tuning of the pre-trained model**- The above process of feature extraction experiment was done upon MobileNet V2 mode but the weights were not updated during the training. Here the model is finely tuned upon the small number of top layers rather than the MobileNet model. For this process, the frozen layer was unfrozen and the model was compiled again to tune the model with the top layers. Now the model has trained again with added some fine-tune EPCOHS. The accuracy and loss were achieved around 0.98 and 0.047 respectively. Here are the comparison images of accuracy and loss invalidation and training datasets shown below. The Final test loss was achieved around 0.0351 and test accuracy was around 0.988.



**Figure 13: Accuracy and Loss check after Re-Compilation**

1. **Predict the test dataset and plotting the prediction**-The model gives two scores for each image and it can be observed that logits are unformalized final scores for the model so the SoftMax function was applied to achieve a probability distribution over classes. Finally, the image of cats and dogs are classified perfectly which is shown below.

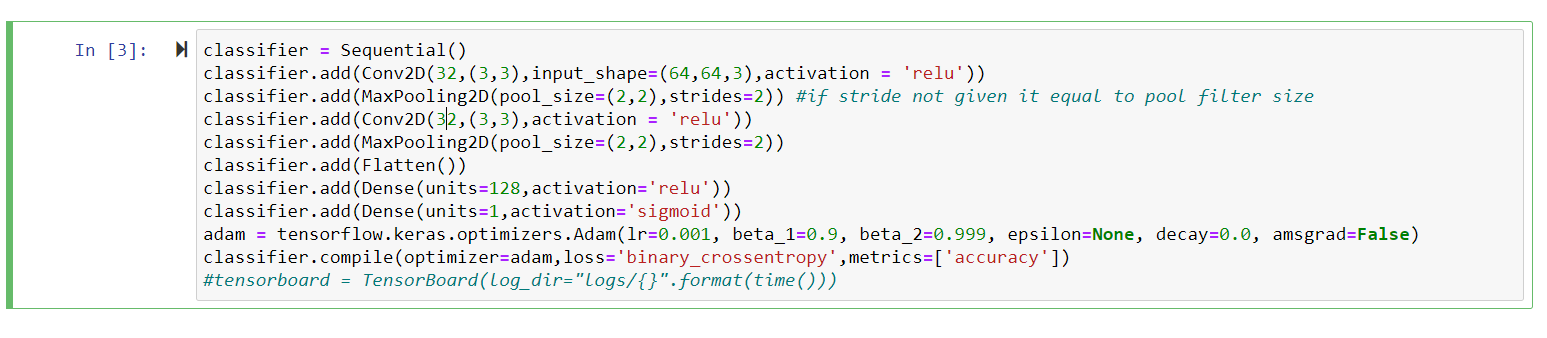


**Figure 14:Predicted Output with the Observation-2**

# Observations-3-Using Encoder output as features

CNN is a kind of neural network that allows to abstract of a higher illustration of the image content. CNN is advanced than the classical image recognition techniques because it takes the raw pixel data. It trains the data upon these raw pixels and extracts the features automatically for effective classification.

In Observation-3, the output encoded feature was generated and it was applied to the image dataset which is needed to be classified. The main operation was done upon the coloring of the image to make the feature more understandable and effective.



**Figure 15:Classifier Creation for Image Dataset**

**Architecture**

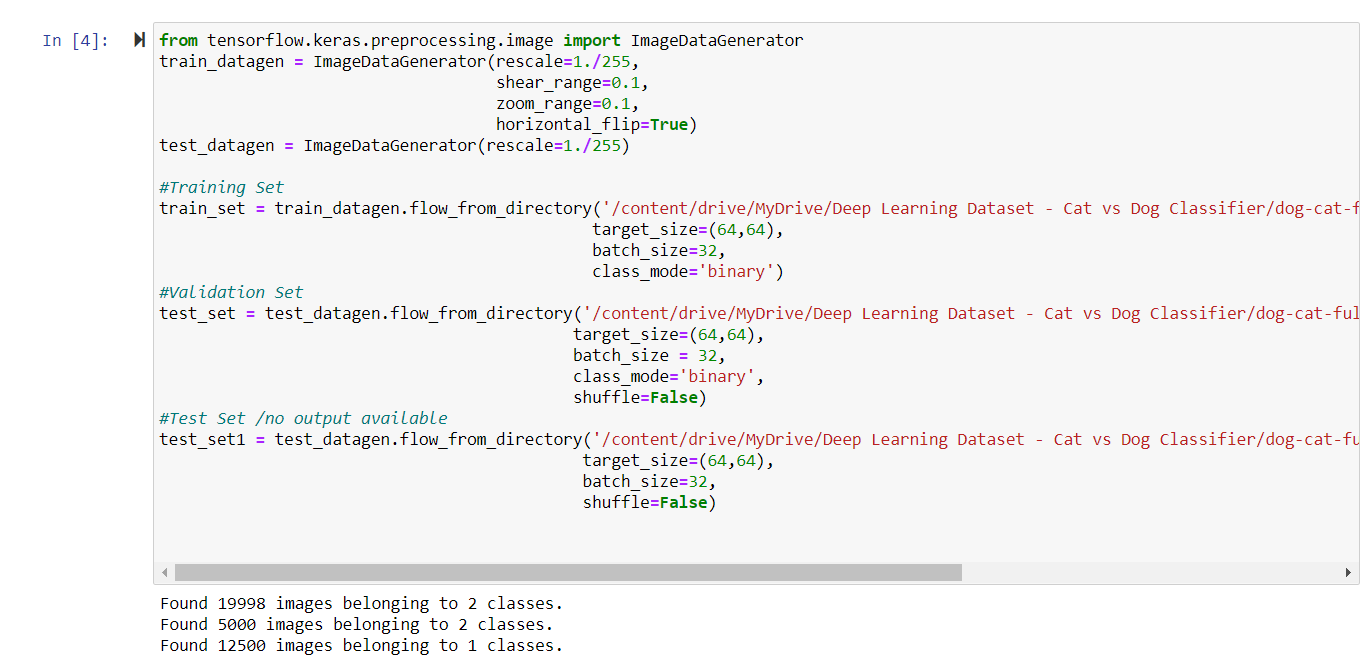
The Basic model architecture is done upon some network parameters such as:

1. Rectifier Linear Unit
2. Adam Optimizer
3. Sigmoid on final Output
4. Binary Cross-Entropy

These parameters are used with an input size of (64,64,3) and the ReLu (Rectifier Linear Unit) function to find out the straight positive and negative output. Here it is used to get the advantage because it doesn’t active all the neurons at the same time. The Adam Optimizer used to train the basic model of DL here is based on descent Methodologies because it has given momentum here by taking the exponentially weighted average to converge towards the minima. The binary cross-entropy is used to predict the compared output to the actual class and here the dog image is given 1 as binary output and the cat is given 0.

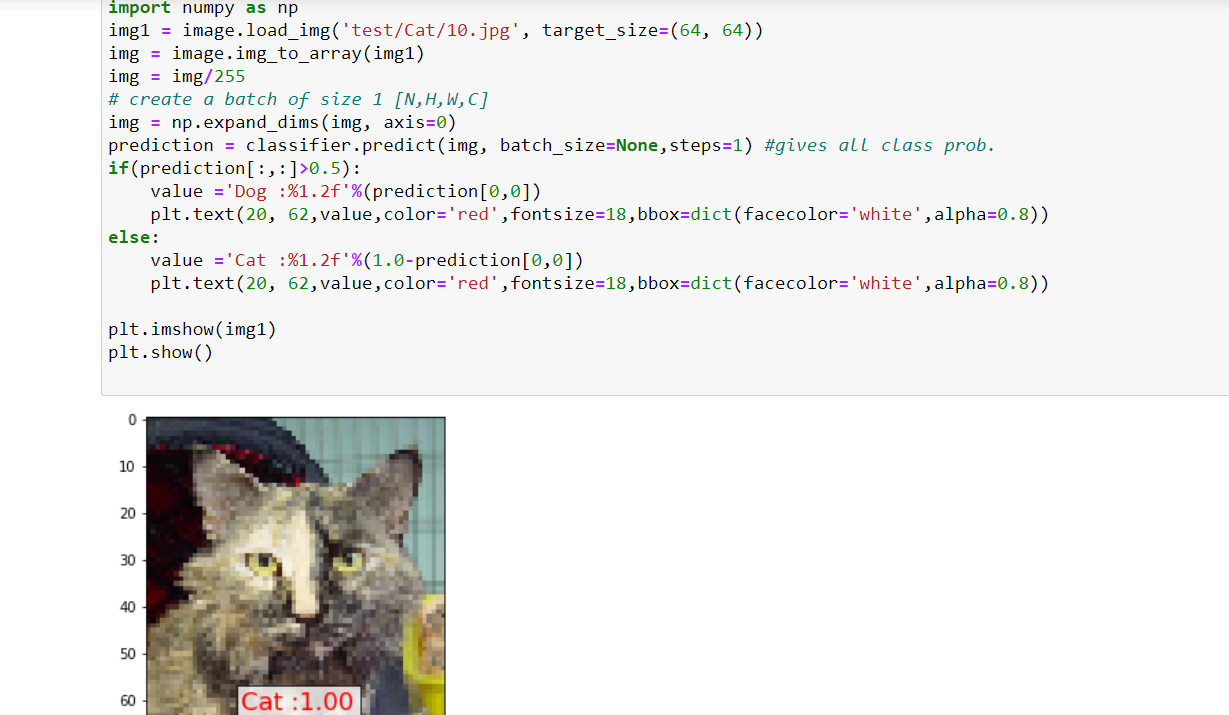
**Data Augmentation**

Data augmentation is used for effective tuning of the datasets and better results. Here some steps and techniques are followed for this model which is-Shearing of images, Random Zoom, And Horizontal Flips. The rescaled values are taken 1./255 and zoom range=0.1 with the same range of shear range. The target batch size was taken around 32 for both training and testing of the dataset. The 2D dimension was set around (64,64).



**Figure 16:Image Dataset Extraction**

With the help of a single image prediction on the first instance, it can be seen that the prediction was done right. It was found that there is a total of 122 images are misclassified among entire datasets.



**Figure 17:Checking a Random Image**

The heat map shows that there are 93 images of cats were considered as a dog which means it was misclassified and vice versa happened with some images of dogs which were classified as cats.



**Figure 18:Misclassified Image of Cat as Dog**

Here is the summary of the classified image is given below which shows that around 813,217 parameters are used and all these parameters are trainable and none.

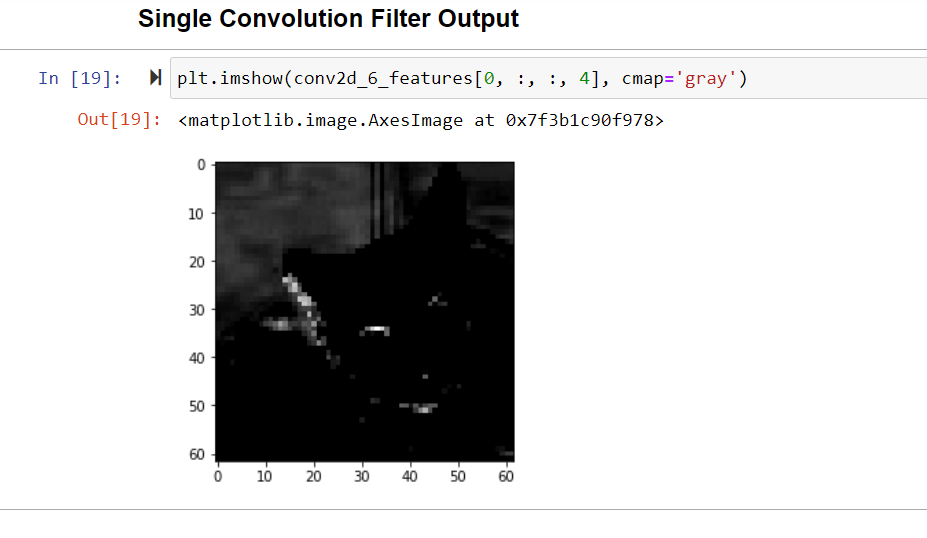
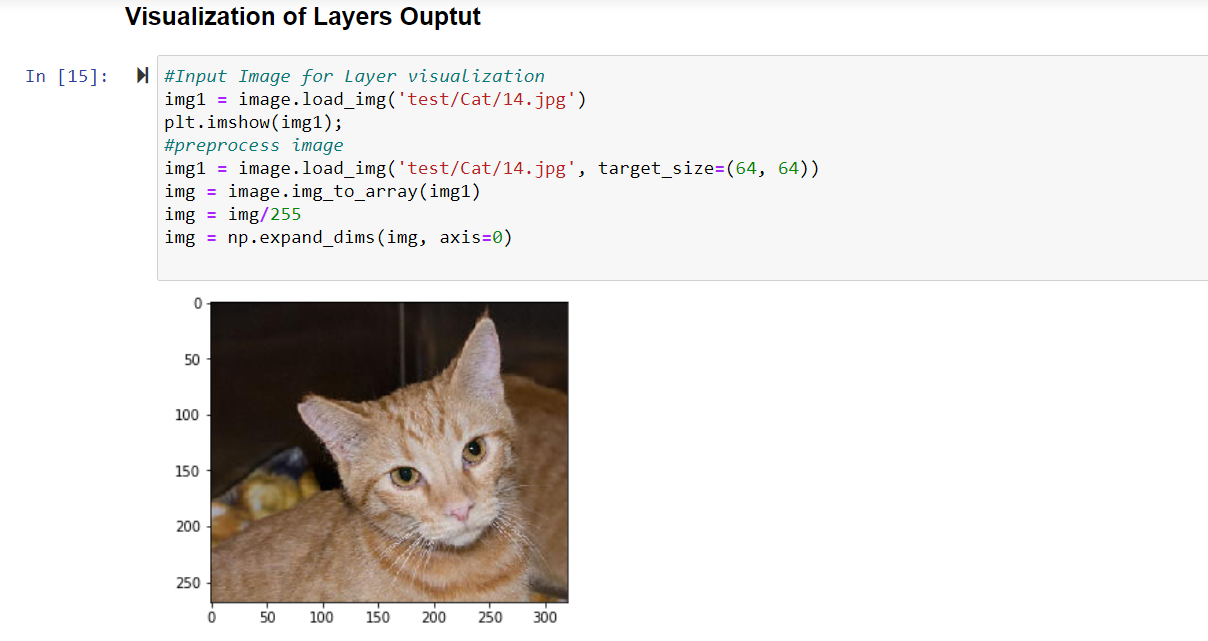


**Figure 19:Misclassified Image of Dog as Cat**

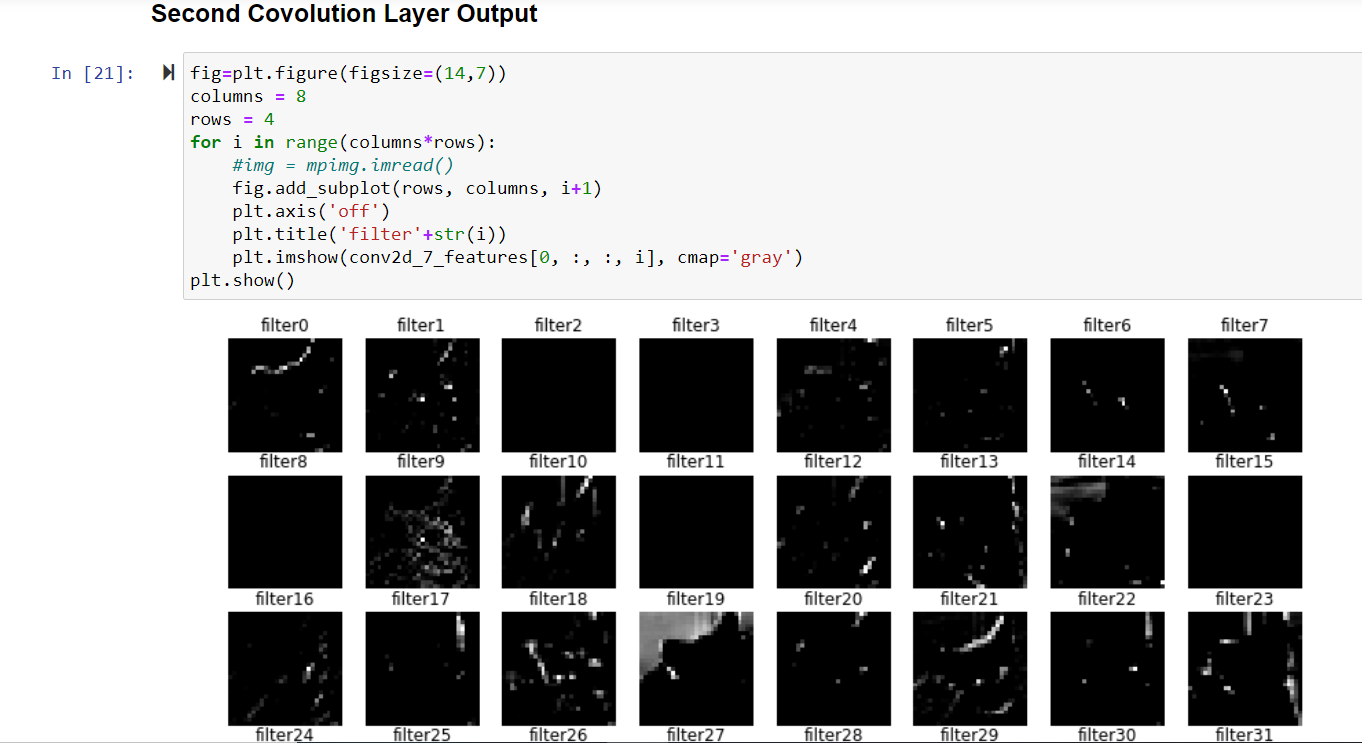
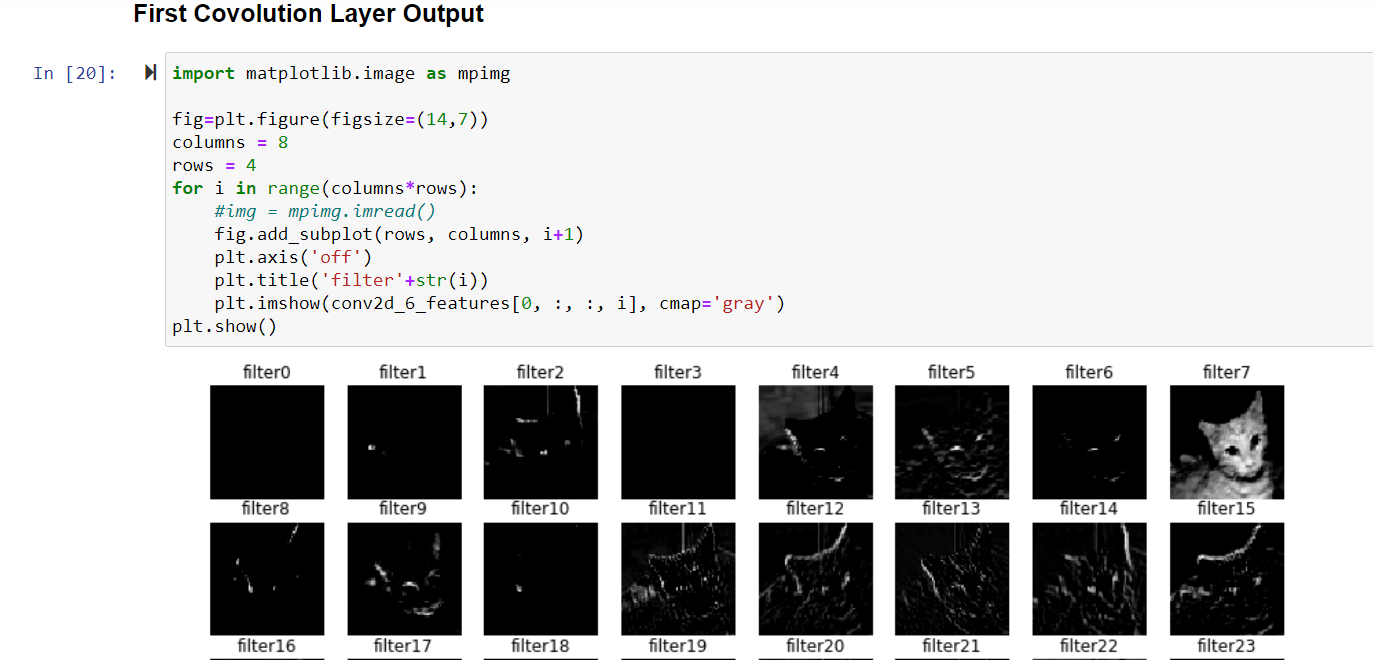
**Visualizations of layer Output**

Here some images with the code scripts are attached which show the layer output with the layer names. These process steps were done to extract the feature of cats and dogs for effective results. Two convolution layer output was extracted where the shape of first convolutional output was (1,29,29,32).

On the second convolutional layer output, the features were extracted effectively where each filter was applied. A total of 31 features were applied to the second layer output.



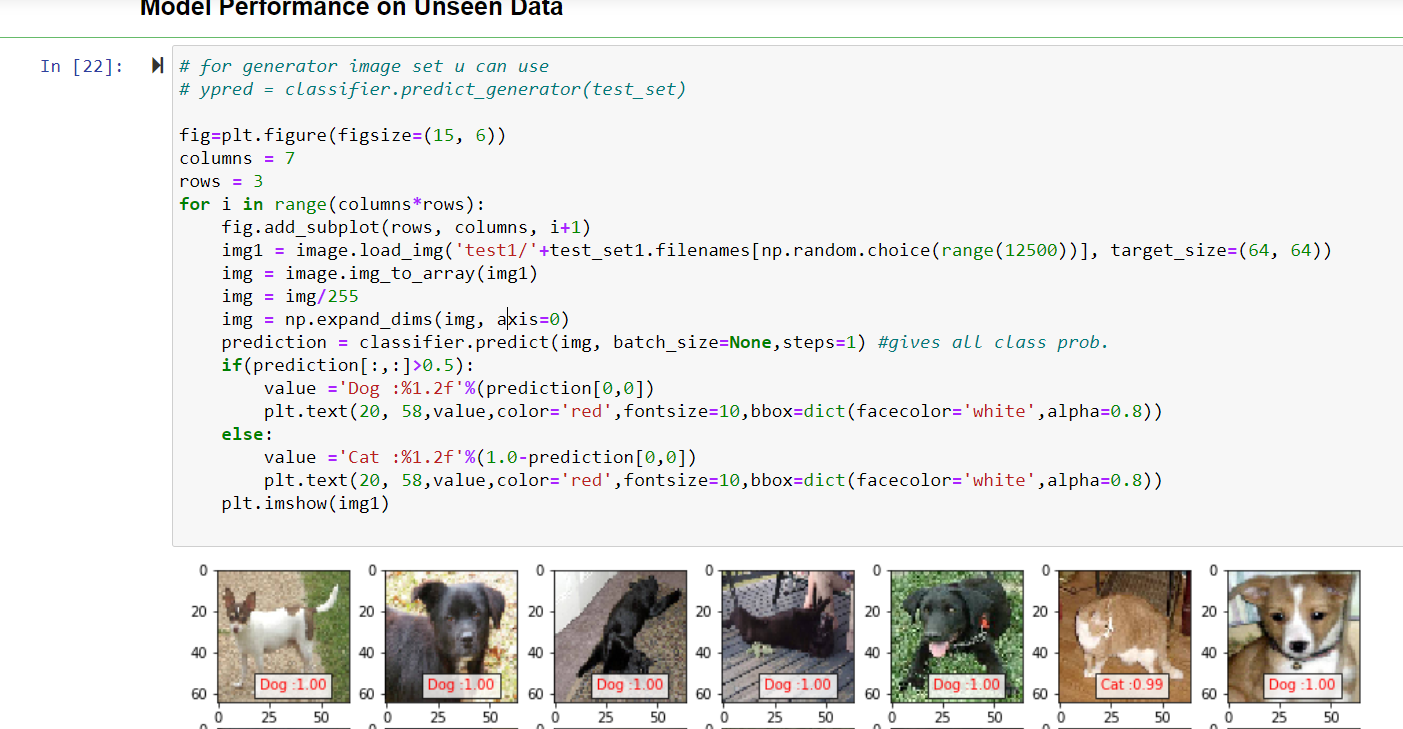
**Figure 20:Layer output visualization for a single layer**



**Figure 21:Feature Extraction by setting encoded feature in the filter in the second layer**

**Model Performance on Unseen Data**

This Step was done to find the generator image set that can be used for image prediction in which the binary output was set none for each prediction. The process was done among 7 columns and 3 rows. Here the set of predicted images of Cat and Dog with the algorithms are attached below:



**Figure 22:Cat or Dog prediction after output feature selection**

The training accuracy was found around 99.6% with the training loss of 0.002454 and validation accuracy was gained around 97.56% and validation loss was around 0.102678. It can be observed that the basic CNN model for image classification has shown a good accuracy but it may be possible to achieve the higher accuracy and lower possible loss by tuning the training parameters and by digging it deeper.

# Detailed Comparison of All three concepts

All three observations have been studied and now it has concluded this model. All three observations have shown a great output with effective accuracy and minimum loss. The first observation, which was based on a basic creation of the CNN model from scratch was done on a complete set of training and test data. On the other hand, the third observation based on the encoded output feature took a small class of data for effective feature selection. Transfer learning has shown the same kind of dataset processing as basic model creation. Talking about the feature selection, the feature was effectively selected and compiled in the third observation because the encoded output feature was easy to compile as it was applied on each image dataset.

The transfer learning and encoded output feature process for image classification of cats and dogs have shown great accuracy. Although, the transfer learning was best in terms of larger to small datasets compilations. The basic model has shown a good accuracy but it was not good in comparison to the rest of the observation models. The loss was measured lowest in the first observation which means that the feature extraction was way better in the basic model. Here is the table given for a comparison of accuracy and loss in all three observations given below.

So the conclusion can be made that the transfer learning and encoded output feature selection process were very effective and both can be used according to the known condition. On the one side, transfer learning takes less time on training data. It allows us to use the knowledge gained from other tasks to tackle new but similar problems quickly and effectively. Another side, the encoded feature selection process takes a long time but gives a perfect output as the feature selection becomes easier in it. The noise reduction or coloring effect can be applied easily to classify any kind of image which gives a huge advantage to this model.

THANK YOU

# Appendix

ALL THE OBSERVATION DONE ON BOTH GOOGLE COLLAB AND JUPYTER NOTEBOOK HAS BEEN ATTACHED IN BOTH .ipynb AND .py FORMAT WITH THIS FILE. EACH OBSERVATION HAS BEEN ATTACHED SEPARATELY FOR BETTER UNDERSTANDING.