AI-BHARATA EMERGING TECHNOLOGIES PVT LTD

INTERN STARTER PROJECT LEVEL-1 REPORT

TITLE: CAT AND DOG CLASSIFIER

By-

M.Surya

AI/ML Developer Intern

BE 8th Semester

CONTENTS

ABSTRACT

The ever-increasing growth of data as well as the need of gaining insights from those data, has prompted us to develop sophisticated tools and technologies which help in increasing the accuracy of insights gained from the data.

Image classification is the process of predicting the class of an image based on the features provided in the training of the model. It is a type of supervised learning model.

The purpose of this project is to develop a state of the art deep learning model which aims to classify an input image into one of the two classes either cat or dog with the sole aim of achieving high amount of accuracy.

A convolutional neural network has been used for initial training of the model, later I have used VGG16.

Key Words: Image classification, convolutional neural network, VGG16, Deep learning

INTRODUCTION

Image classifications forms an integral part of everyday life, wherein we need to distinguish between different things to take better decisions. Although it seems easy to human eyes to distinguish, but when the classification is automated, it becomes difficult to conclude when the process is automated.

Although there exist various machine learning methods for solving this problem, by using deep learning models, we can successfully increase the accuracy by many folds.

In this project, I have used convolutional neural network as my base model and VGG16 as a pre-trained model, to solve the problem and achieve greater performance as well as accuracy.

The dataset used for this project is kagglecatsanddogs5340.zip from Microsoft which consists of 25000 images in total.

METHOD

1)**Convolutional Neural Network**

A Convolutional Neural Network (CNN) is a type of Deep learning architecture commonly used for image classification and recognition tasks. Convolution is a special type of linear operation. CNN consists of many layers namely:

- Input layer
- Convolutional Layer
- Activation Function Layer
- Pool layer
- Fully Connected Layer

In convolutional networks, we represent an image using a cuboid which has a defined length, width, and height as dimensions. Images generally have red, green and blue channels.

Convolutional layers consist of a set of learnable filters, every filter has small width and height, and depth is approximately equal to input volume. In forward pass, we slide each filter across the whole input volume in strides and compute the dot product.

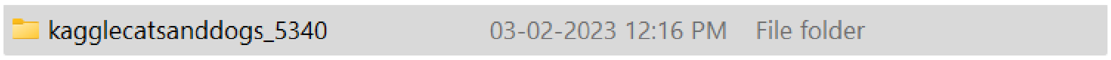
2) DATA COLLECTION

The data is collected and downloaded using the below given link.

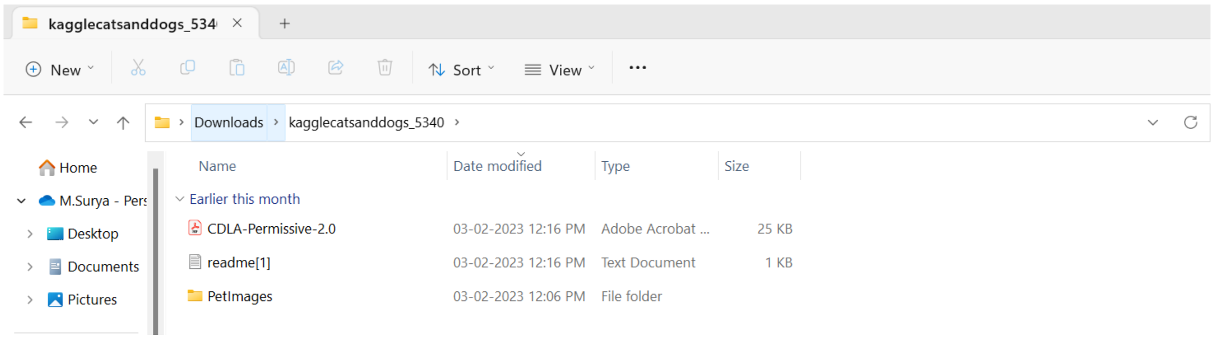
https://www.microsoft.com/en-us/download/details.aspx?id=54765

The images are stored in a tree like structure wherein the root directory is /content/PetImages and its subdirectories are Cat and Dog which consists of 12500 images each of cats and dogs respectively.

Fig: Representation in Local File System

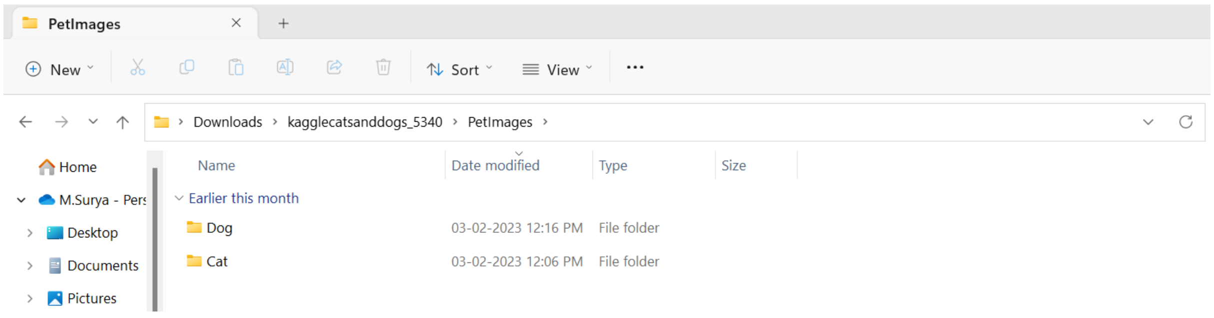


kagglecatsanddogs_5340 03-02-2023 12:16 PM File folder



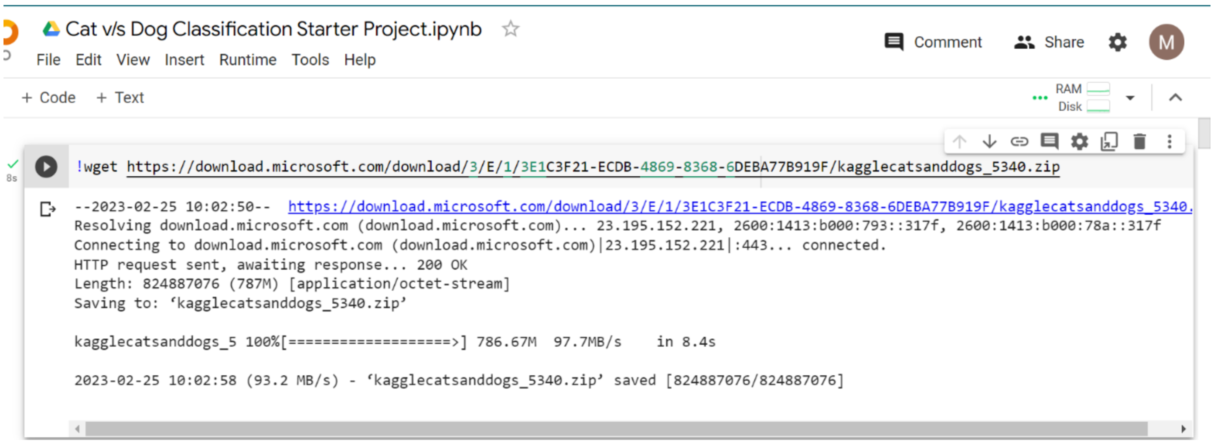
Downloads › kagglecatsanddogs_5340 ›

Name	Date modified	Type	Size
CDLA-Permissive-2.0	03-02-2023 12:16 PM	Adobe Acrobat ...	25 KB
readme[1]	03-02-2023 12:16 PM	Text Document	1 KB
PetImages	03-02-2023 12:06 PM	File folder	



Downloads › kagglecatsanddogs_5340 › PetImages ›

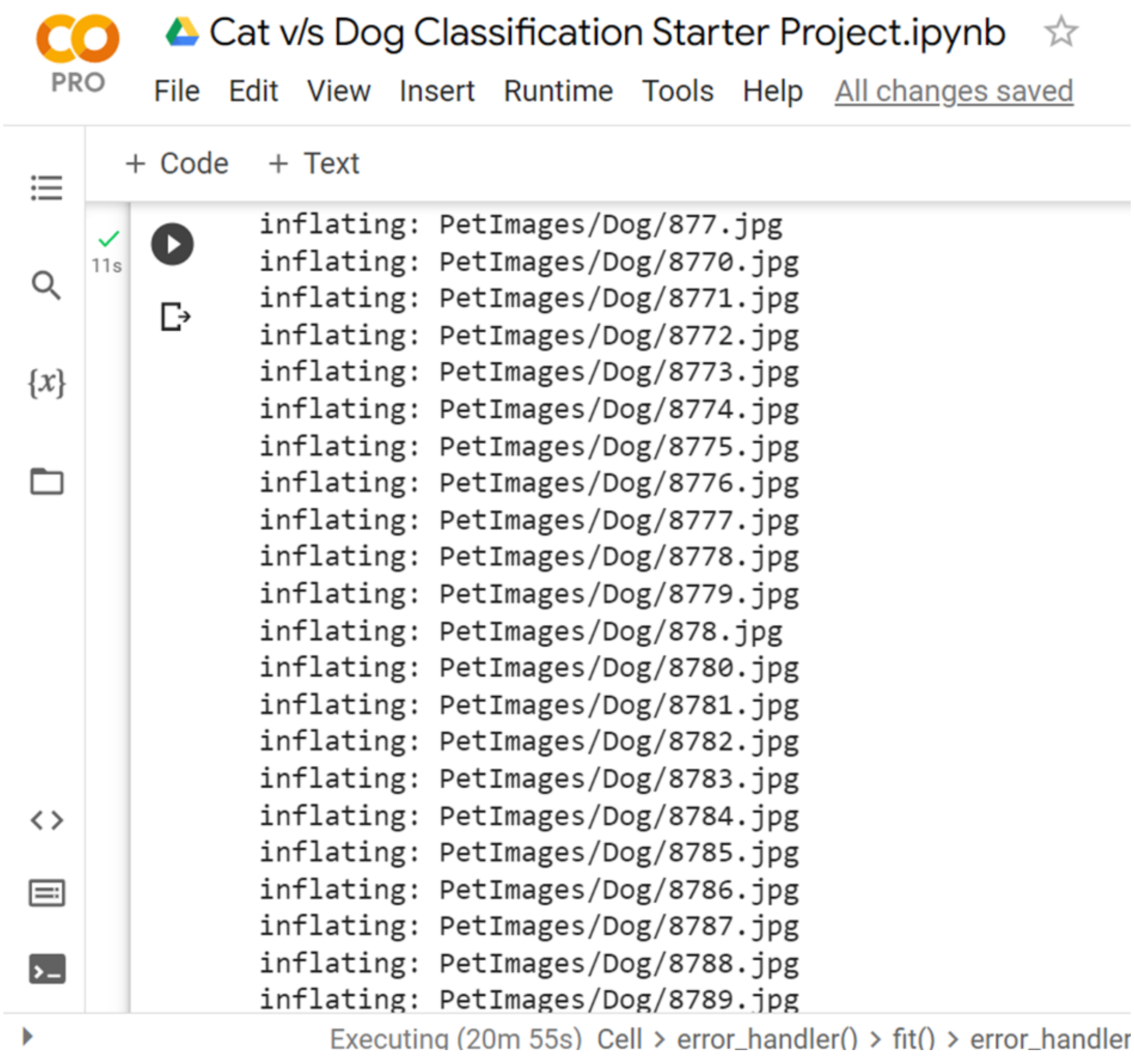
Name	Date modified	Type	Size
Dog	03-02-2023 12:16 PM	File folder	
Cat	03-02-2023 12:06 PM	File folder	


Cat v/s Dog Classification Starter Project.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
8s

```
!wget https://download.microsoft.com/download/3/E/1/3E1C3F21-ECDB-4869-8368-6DEBA77B919F/kagglecatsanddogs_5340.zip
```

```
--2023-02-25 10:02:50--  https://download.microsoft.com/download/3/E/1/3E1C3F21-ECDB-4869-8368-6DEBA77B919F/kagglecatsanddogs_5340.
Resolving download.microsoft.com (download.microsoft.com)... 23.195.152.221, 2600:1413:b000:793::317f, 2600:1413:b000:78a::317f
Connecting to download.microsoft.com (download.microsoft.com)|23.195.152.221|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 824887076 (787M) [application/octet-stream]
Saving to: ‘kagglecatsanddogs_5340.zip’

kagglecatsanddogs_5 100%[===================>] 786.67M  97.7MB/s    in 8.4s

2023-02-25 10:02:58 (93.2 MB/s) - ‘kagglecatsanddogs_5340.zip’ saved [824887076/824887076]
```

<u>Fig: Downloading of Dataset</u>


PRO
Cat v/s Dog Classification Starter Project.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
11s

```
inflating: PetImages/Dog/877.jpg
inflating: PetImages/Dog/8770.jpg
inflating: PetImages/Dog/8771.jpg
inflating: PetImages/Dog/8772.jpg
inflating: PetImages/Dog/8773.jpg
inflating: PetImages/Dog/8774.jpg
inflating: PetImages/Dog/8775.jpg
inflating: PetImages/Dog/8776.jpg
inflating: PetImages/Dog/8777.jpg
inflating: PetImages/Dog/8778.jpg
inflating: PetImages/Dog/8779.jpg
inflating: PetImages/Dog/878.jpg
inflating: PetImages/Dog/8780.jpg
inflating: PetImages/Dog/8781.jpg
inflating: PetImages/Dog/8782.jpg
inflating: PetImages/Dog/8783.jpg
inflating: PetImages/Dog/8784.jpg
inflating: PetImages/Dog/8785.jpg
inflating: PetImages/Dog/8786.jpg
inflating: PetImages/Dog/8787.jpg
inflating: PetImages/Dog/8788.jpg
inflating: PetImages/Dog/8789.jpg
```

Executing (20m 55s) Cell > error_handler() > fit() > error_handler

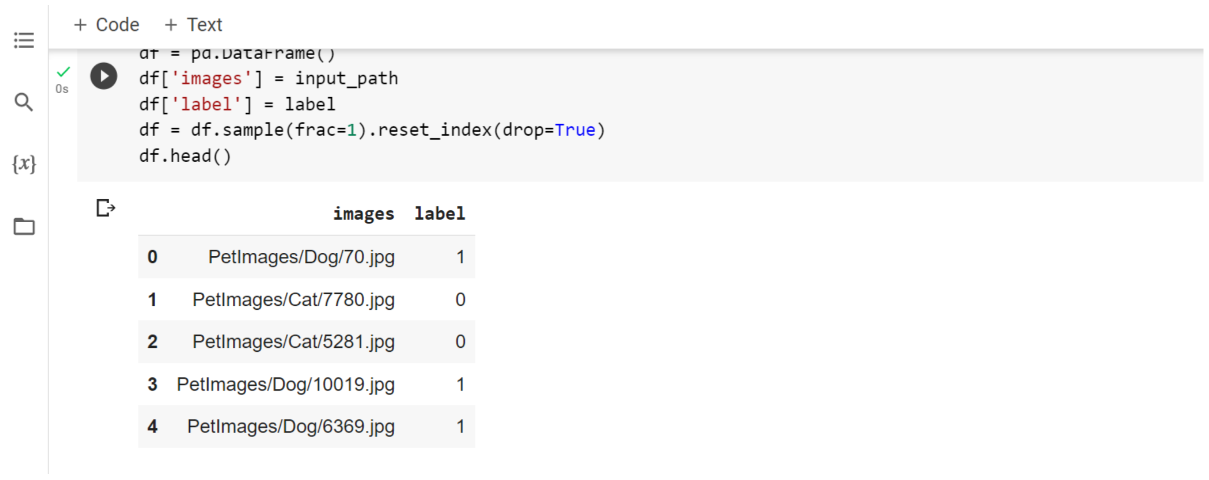
<u>Fig: Unzip Operation</u>

3) ORGANIZATION OF DATA:

After the data collection stage is complete, we need to store and organize the data into an appropriate form so that it can be easier to process the data.

Here, we have used a data-frame to convert the input and output paths of an image for easier processing. Two empty lists are created namely “input_path” and “label”. Here “input_path” stores the path of the image stored in the folders and “label” enables us to distinguish between images of cats and dogs by assigning number 0 to cats and 1 to dogs.

We later split the data-frame into rows and columns, where images column consists of “path_name” of the image, and “label” consists of the numeric value assigned to each image.



```
df = pd.DataFrame()
df['images'] = input_path
df['label'] = label
df = df.sample(frac=1).reset_index(drop=True)
df.head()
```

```
                 images  label
0     PetImages/Dog/70.jpg      1
1   PetImages/Cat/7780.jpg      0
2   PetImages/Cat/5281.jpg      0
3  PetImages/Dog/10019.jpg      1
4   PetImages/Dog/6369.jpg      1
```

4)DATA PREPROCESSING:

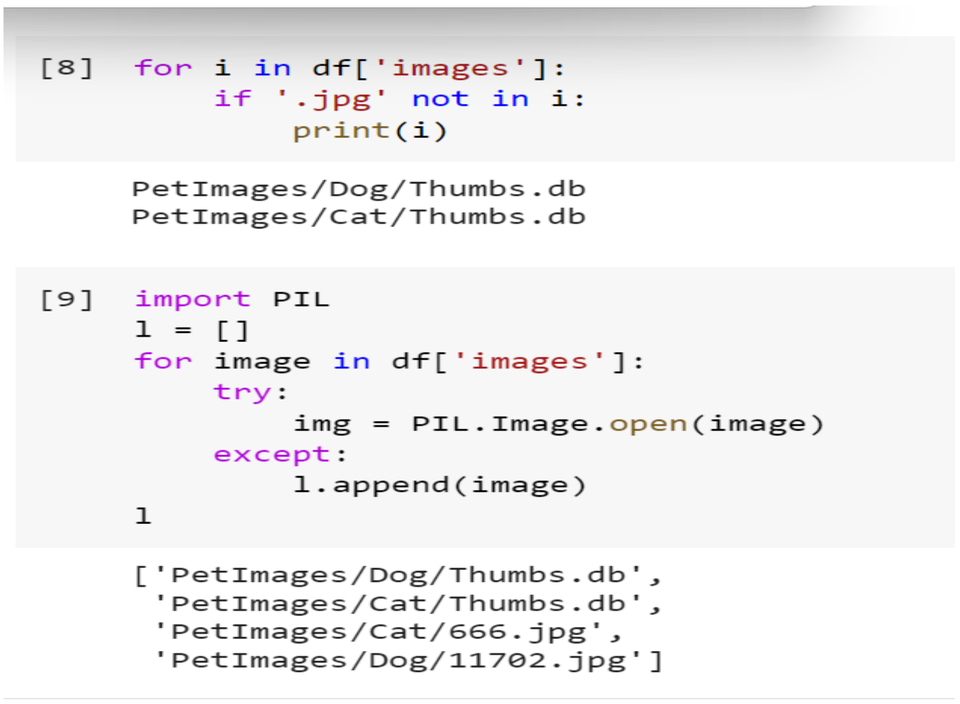
Once the dataset is built, there arises a need to pre-process the data to extract useful features from our deep learning models.

Data pre-processing is quintessential to gain an understanding regarding how it is organized and to be familiar with the structure of data.

Data pre-processing helps in increasing the accuracy of the model by removing unnecessary images which are not helpful to our analysis.

Three steps of pre-processing have been applied in this project:

- Identifying files which are not images.
- Creating a list of non-image type files and corrupted images.
- Deleting the corrupted files and non-image files from the dataset.



```
[8]  for i in df['images']:
         if '.jpg' not in i:
             print(i)

     PetImages/Dog/Thumbs.db
     PetImages/Cat/Thumbs.db

[9]  import PIL
     l = []
     for image in df['images']:
         try:
             img = PIL.Image.open(image)
         except:
             l.append(image)
     l

     ['PetImages/Dog/Thumbs.db',
      'PetImages/Cat/Thumbs.db',
      'PetImages/Cat/666.jpg',
      'PetImages/Dog/11702.jpg']
```

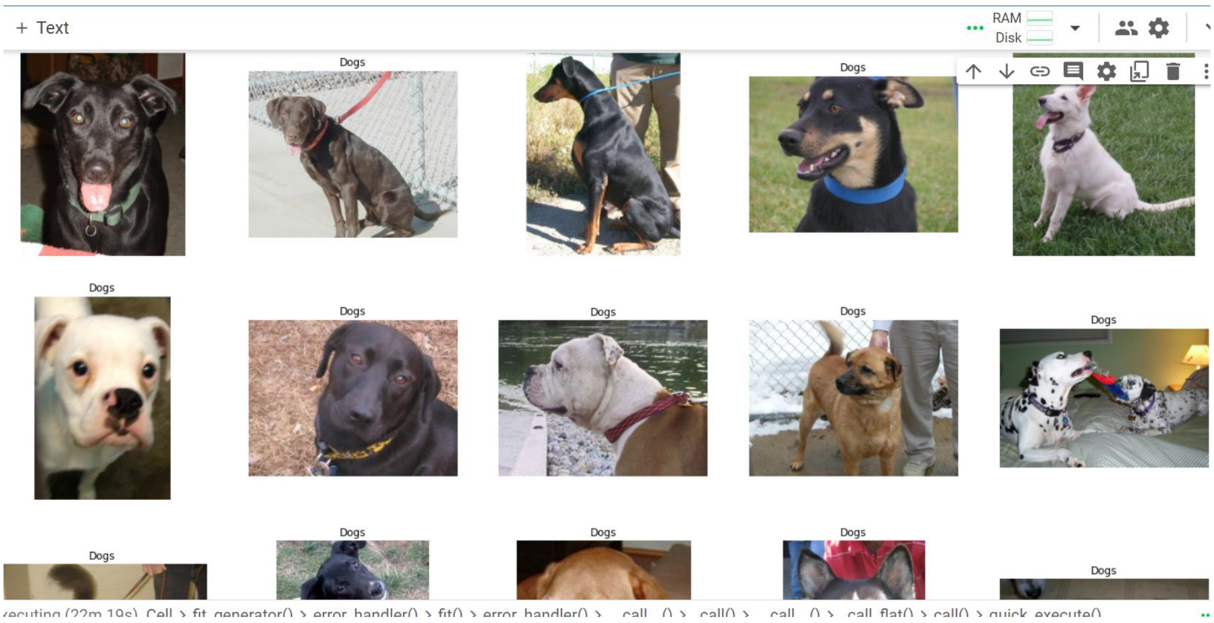
Fig: Deleting redundant images

5)EXPLORATORY DATA ANALYSIS (EDA):

The pre-processing step is helpful to extract images which are not having any sort of defect from the dataset.

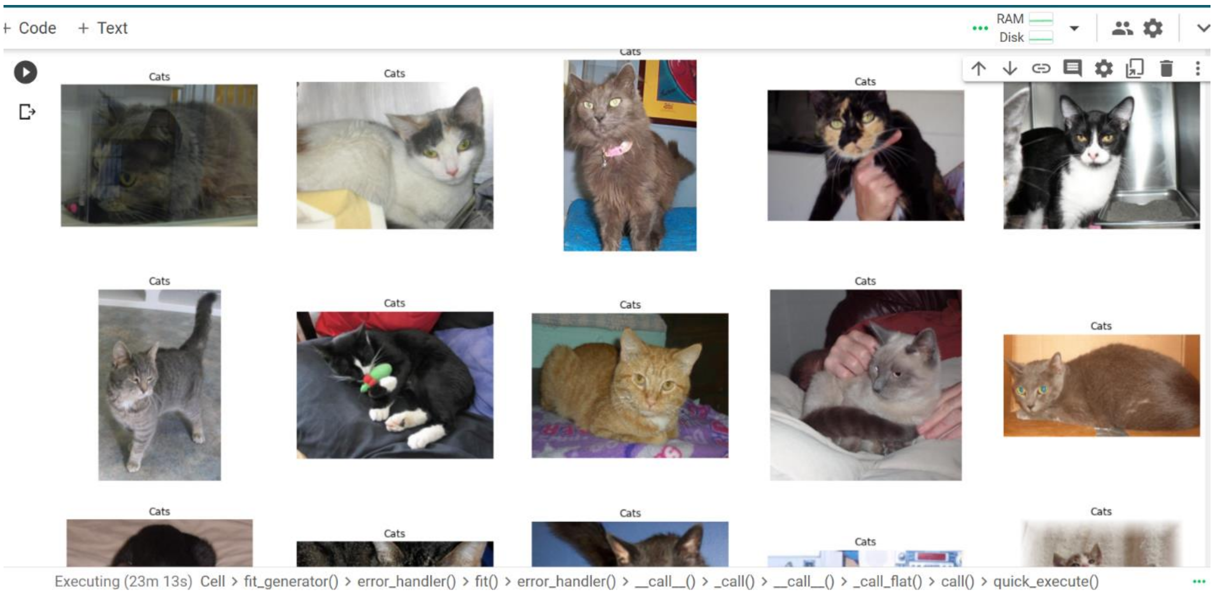
Exploratory data analysis (EDA) is an approach to analyse the data using visual techniques. This approach is applied to discover the trends, underlying patterns, or to check assumptions with the help of statistical summary and graphical representations.

Here, we display the images of cats and dogs using a 5 x 5 grid by using the label assigned to the images. Each image represented in the grid can have different saturation and qualities.



ecuting (22m 19s) Cell › fit_generator() › error_handler() › fit() › error_handler() › __call__() › _call() › __call__() › _call_flat() › call() › quick_execute()

Fig: Grid representation of dogs

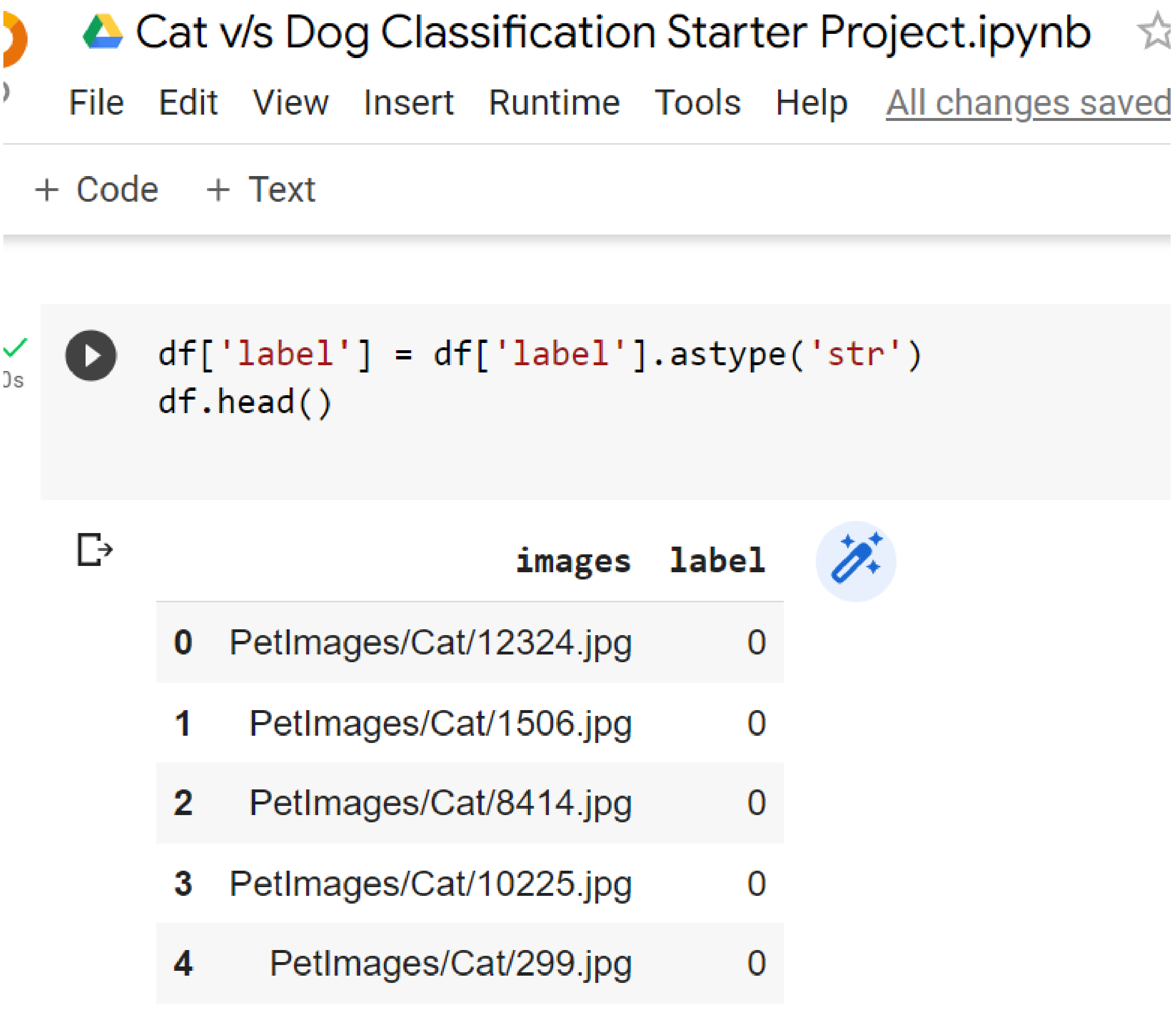


Executing (23m 13s) Cell › fit_generator() › error_handler() › fit() › error_handler() › __call__() › _call() › __call__() › _call_flat() › call() › quick_execute()

Fig: Grid representation of Cats

6)CREATING DATA GENERATOR FOR IMAGES:

Here we convert the data type of labels assigned to images as strings. Data Generators loads the data from the disk for reading and training data directly thereby saving the RAM space.


Cat v/s Dog Classification Starter Project.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
0s

```
df['label'] = df['label'].astype('str')
df.head()
```

```
                  images  label
0  PetImages/Cat/12324.jpg      0
1   PetImages/Cat/1506.jpg      0
2   PetImages/Cat/8414.jpg      0
3  PetImages/Cat/10225.jpg      0
4    PetImages/Cat/299.jpg      0
```

Fig: Converting labels into strings

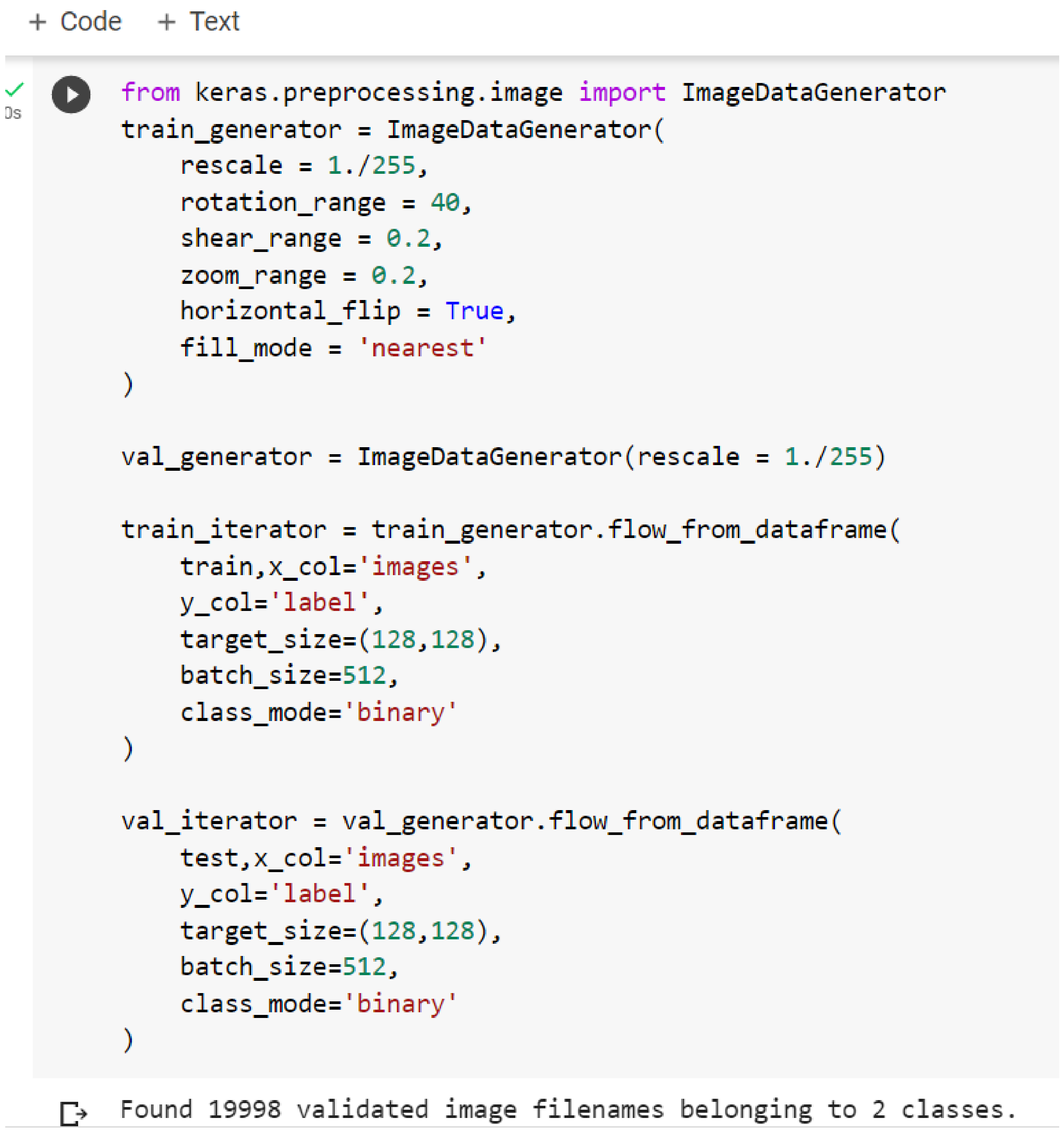
7)DATASET AND DATA AUGMENTATION:

The data-frame is divided into training and test sets by using scikit-learn library and train_test_split method to split the arrays into random train and test subsets. There are 19998 images of each class (viz.) cat and dog for training, whereas 5000 images of each class is assigned for testing/validation.

Data augmentation is the process of artificially increasing the amount of data by generating new data points from the existing data. ImageDataGenerator module provided by Keras is responsible for data augmentation. Images are resized to 128 x 128. By applying data augmentation, the diversity and size of the image can be increased.

Augmentation techniques used in this project are:

- rescale
- rotation_range
- shear_range
- zoom_range
- horizontal_flip
- fill_mode



```python
from keras.preprocessing.image import ImageDataGenerator
train_generator = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 40,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True,
    fill_mode = 'nearest'
)

val_generator = ImageDataGenerator(rescale = 1./255)

train_iterator = train_generator.flow_from_dataframe(
    train,x_col='images',
    y_col='label',
    target_size=(128,128),
    batch_size=512,
    class_mode='binary'
)

val_iterator = val_generator.flow_from_dataframe(
    test,x_col='images',
    y_col='label',
    target_size=(128,128),
    batch_size=512,
    class_mode='binary'
)
```

```
Found 19998 validated image filenames belonging to 2 classes.
```

Fig: Dataset Augmentation

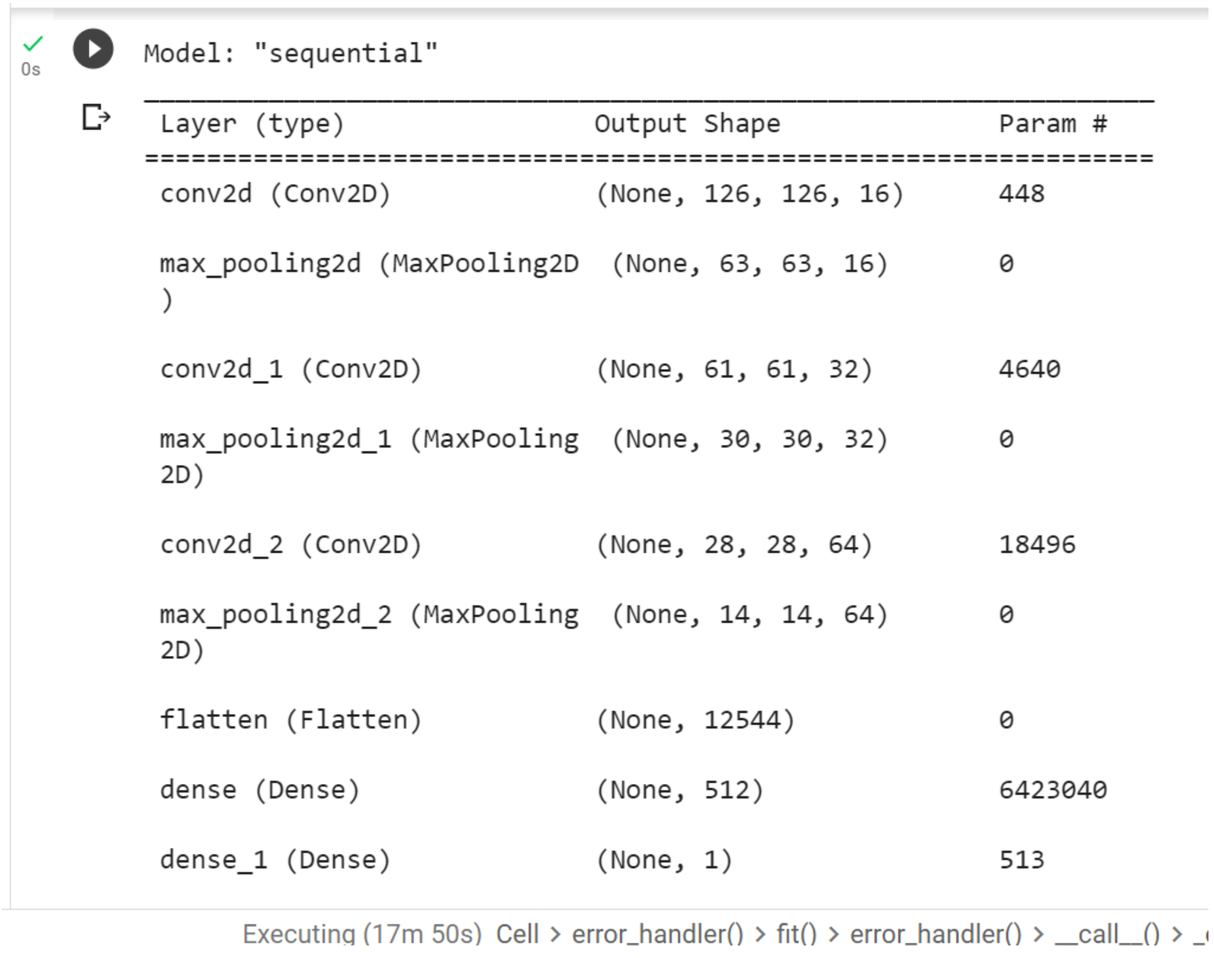
8)MODEL ARCHITECTURE:

- Classifier is given the name Sequential. The first layer is called Conv2D layer. Since it is the first layer of the model, input shape of the images being fed into the model is mentioned.

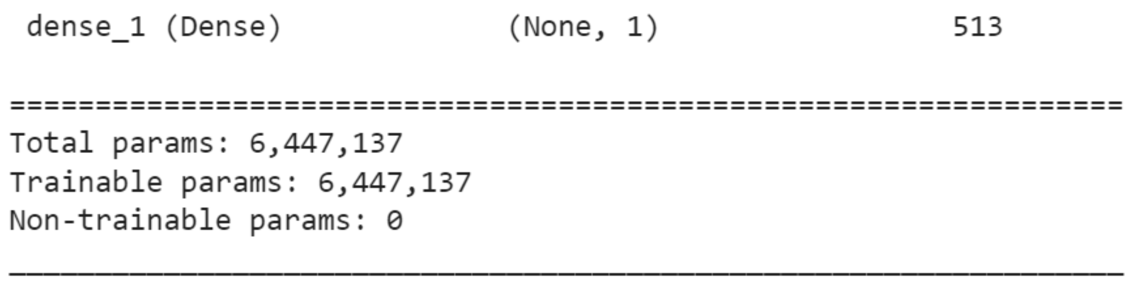
- The next layer is MaxPool2D layer, which is used to get maximum pixel value to the next layer. Then there is another Conv2D layer followed by MaxPool2D layer, later in this project, we have Flatten and Dense layers, flatten layers are used to convert all the resultant 2D arrays from the pooled feature maps into a single long continuous linear vector.

- The Sequential model API is used to build model. The sequential API allows you to create models layer-by-layer. The 'add()' function to add layers to our model. The model needs to know what input shape it should expect. For this reason, only the first layer in a Sequential model needs to receive information about its input shape.

- Maximum pooling or max pooling is a pooling operation that calculates the maximum or largest value in each patch of a feature map. The results are down sampled or pooled feature maps that highlight the most present feature in the patch.


0s
Executing (17m 50s) Cell > error_handler() > fit() > error_handler() > __call__() >

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 16)      448

 max_pooling2d (MaxPooling2D  (None, 63, 63, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 61, 61, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 30, 30, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 28, 28, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 512)               6423040

 dense_1 (Dense)             (None, 1)                 513
```



```
 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 6,447,137
Trainable params: 6,447,137
Non-trainable params: 0
_________________________________________________________________
```

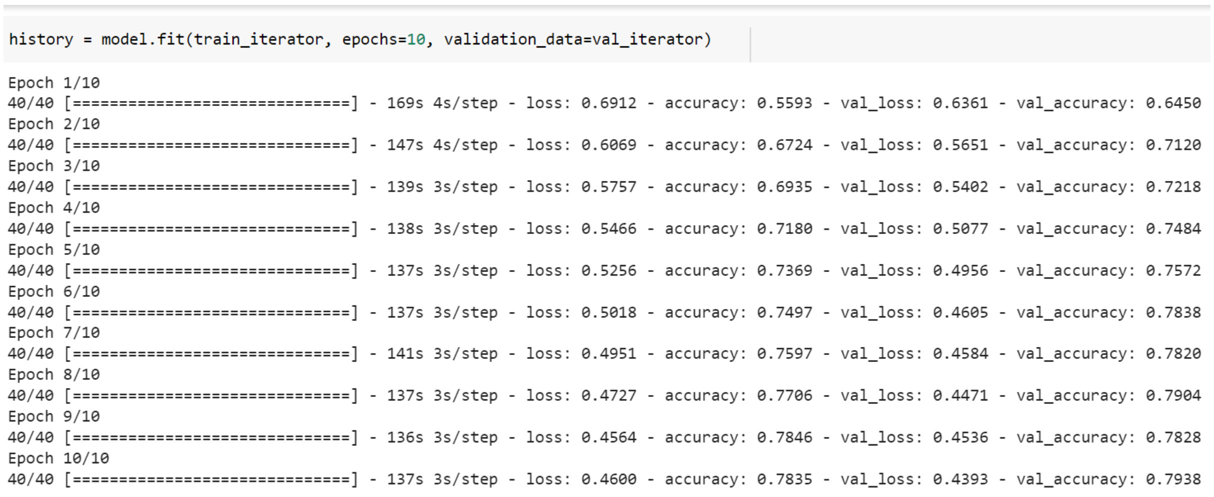
Fig: Architecture of CNN

9)MODEL COMPILATION AND TRAINING:

In the model compilation, the optimizer algorithm, loss function and metrics are the parameters are passed.

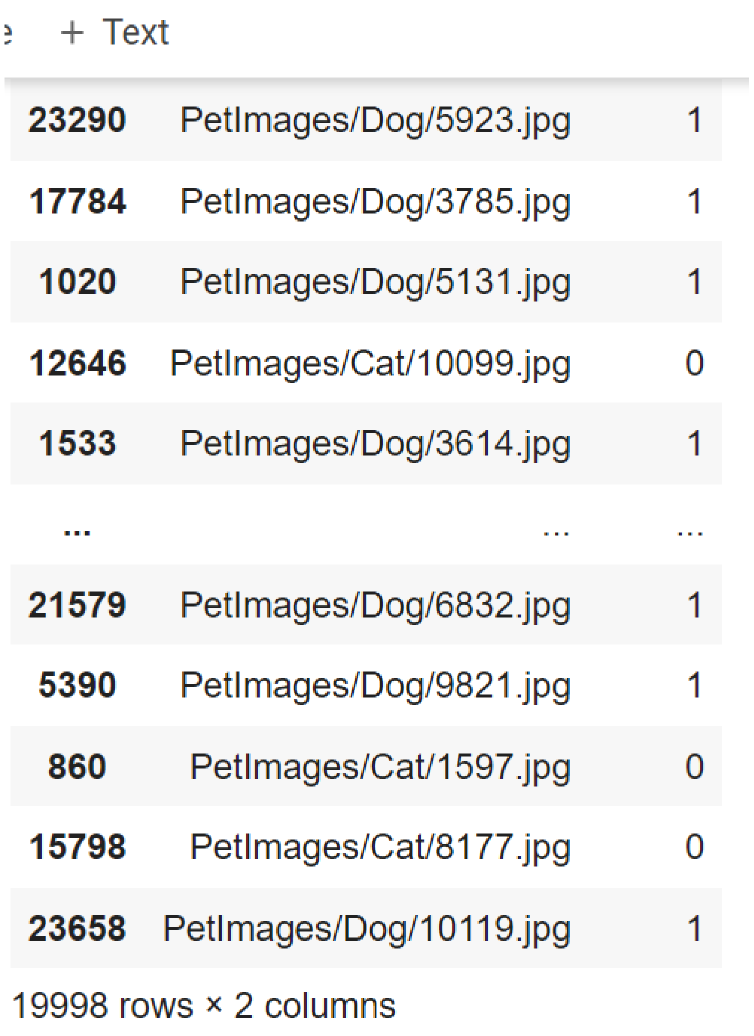
- **Adam:** Optimization Algorithm
- **Binary Cross Entropy:** It is the loss function for binary outputs.
- **Accuracy:** Metric used

Sequential models model.fit() method is used to train the model with train_iterator, validation_iterator and the model is trained for 10 epochs.



```
history = model.fit(train_iterator, epochs=10, validation_data=val_iterator)

Epoch 1/10
40/40 [==============================] - 169s 4s/step - loss: 0.6912 - accuracy: 0.5593 - val_loss: 0.6361 - val_accuracy: 0.6450
Epoch 2/10
40/40 [==============================] - 147s 4s/step - loss: 0.6069 - accuracy: 0.6724 - val_loss: 0.5651 - val_accuracy: 0.7120
Epoch 3/10
40/40 [==============================] - 139s 3s/step - loss: 0.5757 - accuracy: 0.6935 - val_loss: 0.5402 - val_accuracy: 0.7218
Epoch 4/10
40/40 [==============================] - 138s 3s/step - loss: 0.5466 - accuracy: 0.7180 - val_loss: 0.5077 - val_accuracy: 0.7484
Epoch 5/10
40/40 [==============================] - 137s 3s/step - loss: 0.5256 - accuracy: 0.7369 - val_loss: 0.4956 - val_accuracy: 0.7572
Epoch 6/10
40/40 [==============================] - 137s 3s/step - loss: 0.5018 - accuracy: 0.7497 - val_loss: 0.4605 - val_accuracy: 0.7838
Epoch 7/10
40/40 [==============================] - 141s 3s/step - loss: 0.4951 - accuracy: 0.7597 - val_loss: 0.4584 - val_accuracy: 0.7820
Epoch 8/10
40/40 [==============================] - 137s 3s/step - loss: 0.4727 - accuracy: 0.7706 - val_loss: 0.4471 - val_accuracy: 0.7904
Epoch 9/10
40/40 [==============================] - 136s 3s/step - loss: 0.4564 - accuracy: 0.7846 - val_loss: 0.4536 - val_accuracy: 0.7828
Epoch 10/10
40/40 [==============================] - 137s 3s/step - loss: 0.4600 - accuracy: 0.7835 - val_loss: 0.4393 - val_accuracy: 0.7938
```

Fig: Training of CNN Model



```
+ Text

23290   PetImages/Dog/5923.jpg   1
17784   PetImages/Dog/3785.jpg   1
 1020   PetImages/Dog/5131.jpg   1
12646  PetImages/Cat/10099.jpg   0
 1533   PetImages/Dog/3614.jpg   1
  ...                      ...  ...
21579   PetImages/Dog/6832.jpg   1
 5390   PetImages/Dog/9821.jpg   1
  860   PetImages/Cat/1597.jpg   0
15798   PetImages/Cat/8177.jpg   0
23658  PetImages/Dog/10119.jpg   1
19998 rows × 2 columns
```

Fig: Training Set

GRAD-CAM

- Deep learning methods are considered to have a major issue of model interpretability, in other words, humans cannot interpret what is happening once the model is being trained, so they are called “black box methods”.

  There are a few important factors which needs to be considered for model interpretability:

  - Where the network is “focussing” in the input image.
  - Which series of neurons activated in the forward-pass during inference/prediction.
  - How the network arrived at the final Output.

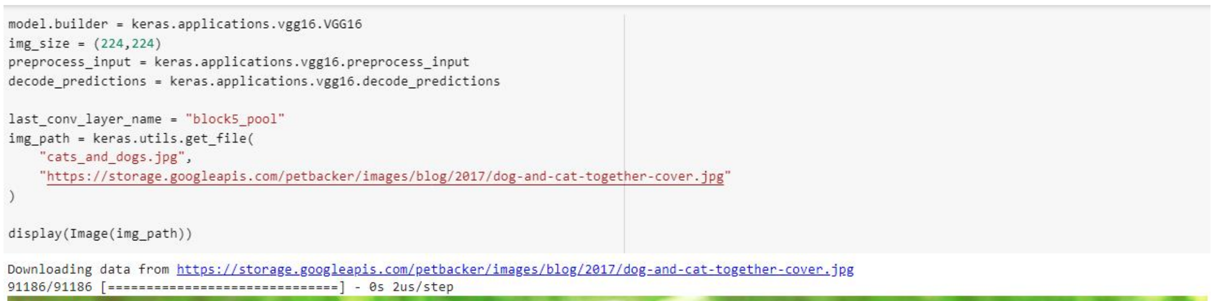
Definition of GRAD-CAM:

GRAD-CAM uses the gradient of any target concept (for example dog in a classification network flowing into the final convolutional layer to produce a localization map highlighting the important regions of an image for predicting the concept.

Grad-CAM is applicable for various CNN model families:

- CNN with fully connected layers.
- CNN used for structured outputs.
- CNN used for multimodal inputs.

In this project, I have obtained Grad-CAM heatmap representation of images which are trained by VGG16 architecture.



```
model.builder = keras.applications.vgg16.VGG16
img_size = (224,224)
preprocess_input = keras.applications.vgg16.preprocess_input
decode_predictions = keras.applications.vgg16.decode_predictions

last_conv_layer_name = "block5_pool"
img_path = keras.utils.get_file(
    "cats_and_dogs.jpg",
    "https://storage.googleapis.com/petbacker/images/blog/2017/dog-and-cat-together-cover.jpg"
)

display(Image(img_path))
```

```
Downloading data from https://storage.googleapis.com/petbacker/images/blog/2017/dog-and-cat-together-cover.jpg
91186/91186 [==============================] - 0s 2us/step
```

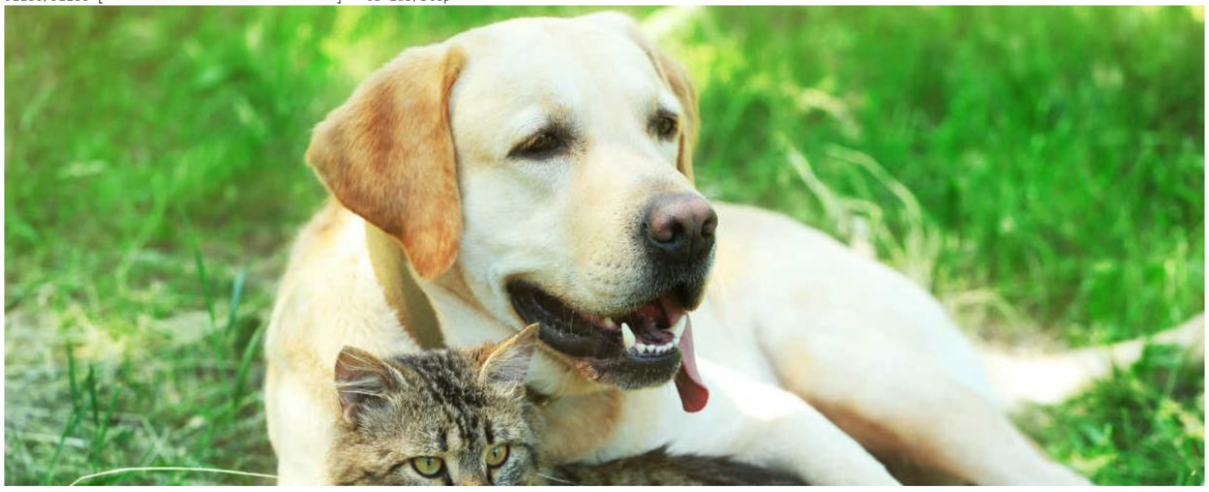


Fig: Image of Cat and a Dog imported for GRAD-CAM

```
heatmap = make_gradcam_heatmap(img_array, model, last_conv_layer_name)

plt.matshow(heatmap)
plt.show()
```

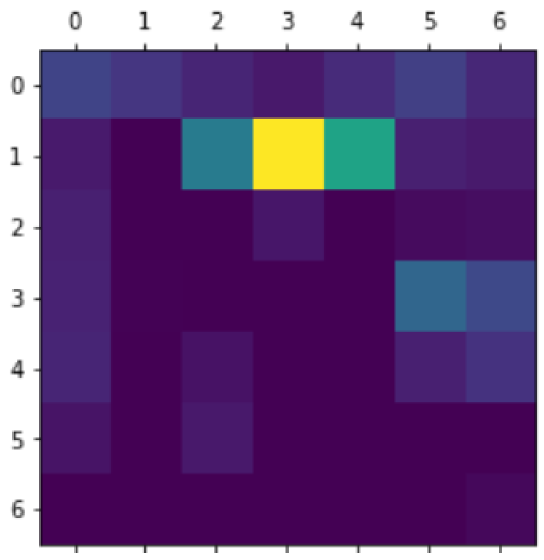


Fig: Heatmap representing the presence of cats and dogs in the entire image

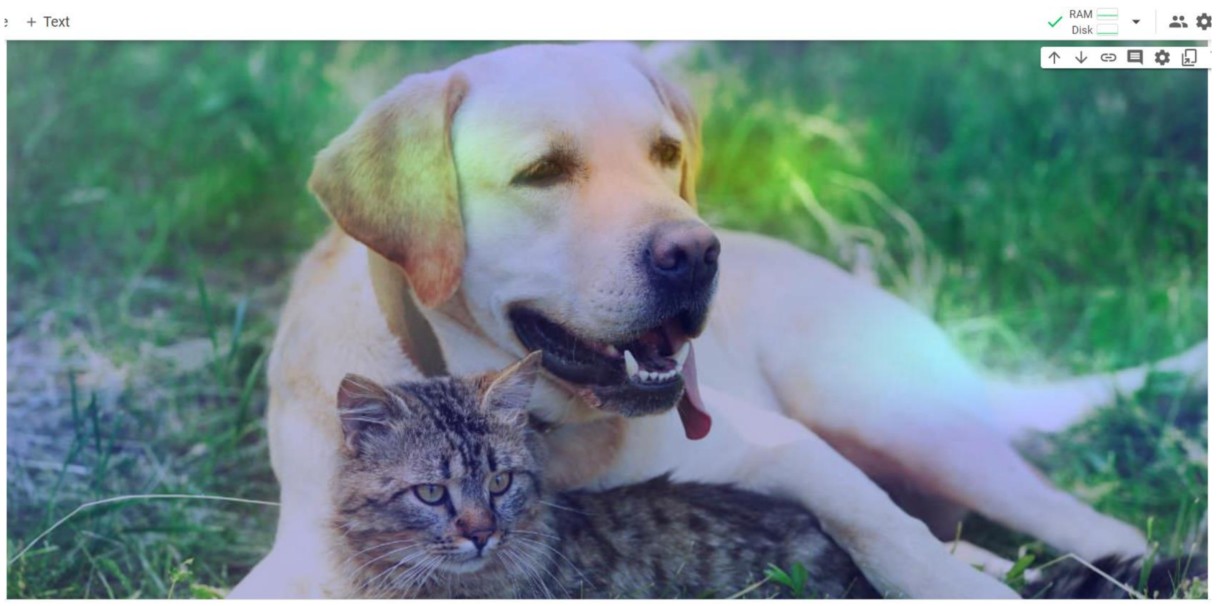
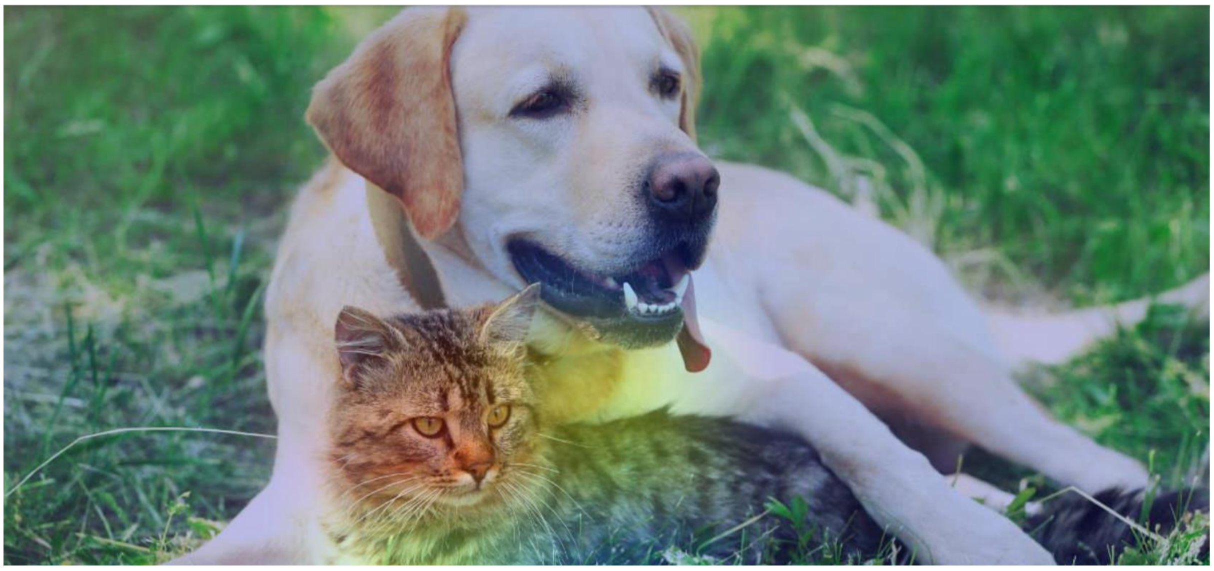
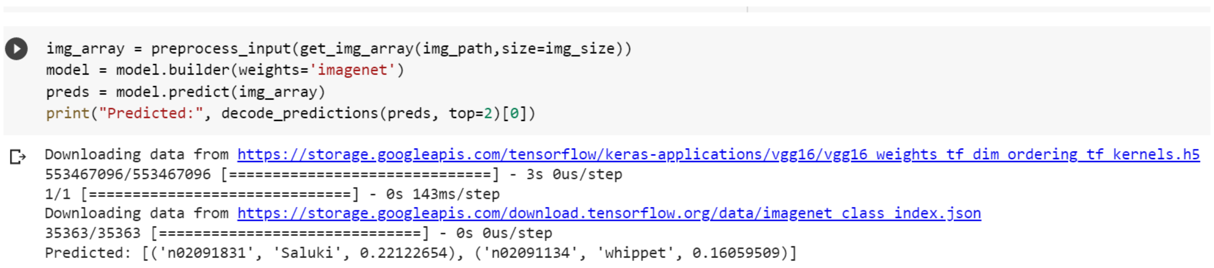


Fig: Image representing the evidence that dog is present



<u>Fig: Image highlighting the localization of cat</u>



```
img_array = preprocess_input(get_img_array(img_path,size=img_size))
model = model.builder(weights='imagenet')
preds = model.predict(img_array)
print("Predicted:", decode_predictions(preds, top=2)[0])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 [==============================] - 3s 0us/step
1/1 [==============================] - 0s 143ms/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
35363/35363 [==============================] - 0s 0us/step
Predicted: [('n02091831', 'Saluki', 0.22122654), ('n02091134', 'whippet', 0.16059509)]
```

<u>Fig: Prediction part using GRAD-CAM</u>

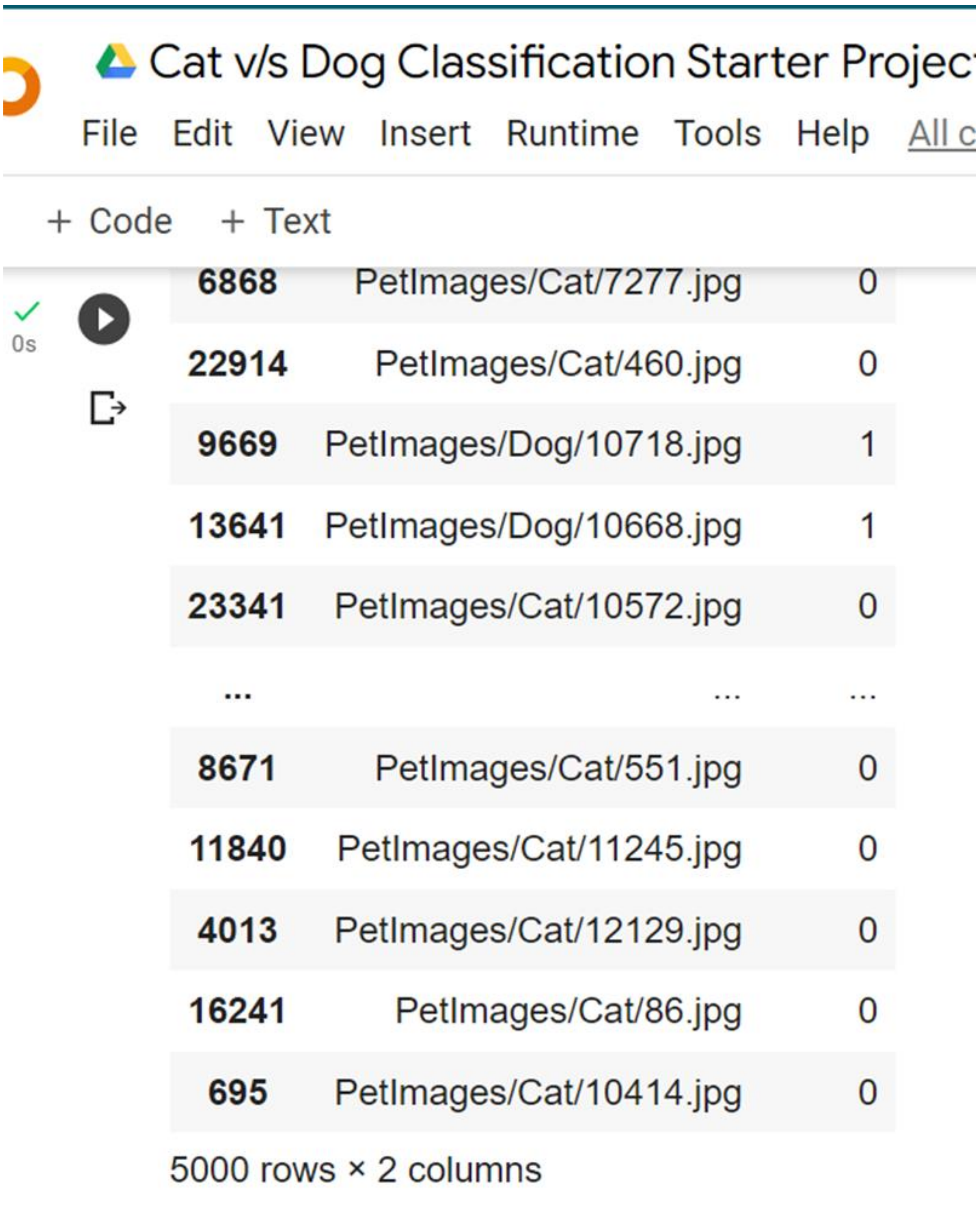
RESULTS AND ANALYSIS

The model was trained for 10 epochs. The observations which I came across while training the model are:

- Training accuracy and validation accuracy increases every iteration.
- Training loss and validation loss decreases every iteration.

COMPARISON OF CNN AND VGG PERFORMANCE

CNN	**VGG16**
Non-trainable parameters are 0	Trainable parameters are 0
Input Shape is of size 128 x 128.	Input shape is of size 224 x 224.
Training accuracy and validation accuracy increases every iteration.	Training accuracy and validation accuracy first increases and later becomes inconsistent
Validation accuracy begins from smallest value and later increases.	Validation accuracy begins from a larger value.
Given an input image, it might lead to misclassification due to certain features of images.	Chances of misclassification is minimal, since it is a pretrained network.


Cat v/s Dog Classification Starter Projec
File Edit View Insert Runtime Tools Help All c
+ Code + Text
0s

```
6868     PetImages/Cat/7277.jpg   0
22914     PetImages/Cat/460.jpg   0
9669    PetImages/Dog/10718.jpg   1
13641   PetImages/Dog/10668.jpg   1
23341   PetImages/Cat/10572.jpg   0
...                         ...  ...
8671      PetImages/Cat/551.jpg   0
11840   PetImages/Cat/11245.jpg   0
4013    PetImages/Cat/12129.jpg   0
16241      PetImages/Cat/86.jpg   0
695     PetImages/Cat/10414.jpg   0
5000 rows × 2 columns
```

Fig: Test Set given to CNN Architecture

PERFORMANCE MEASUREMENT OF CNN

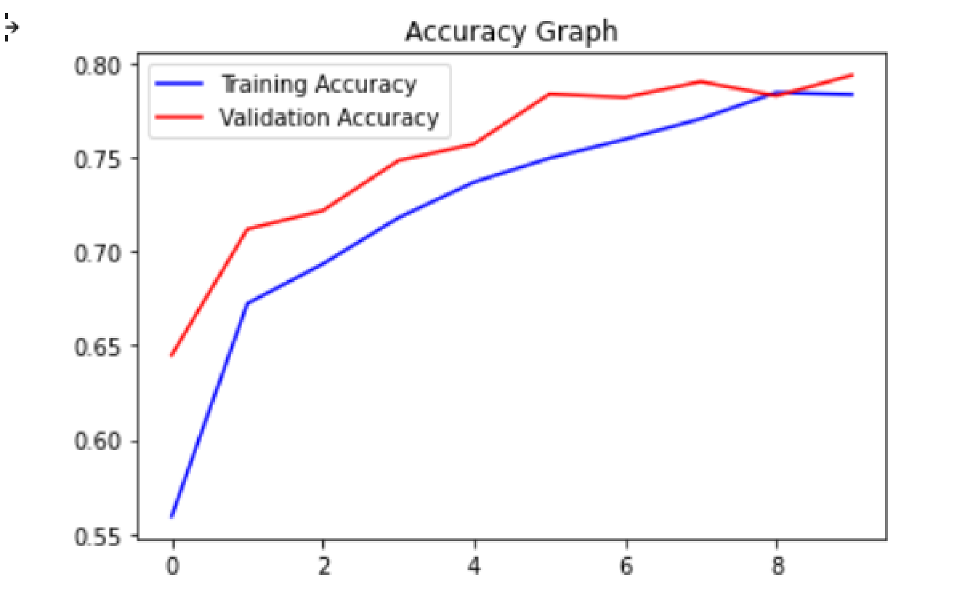
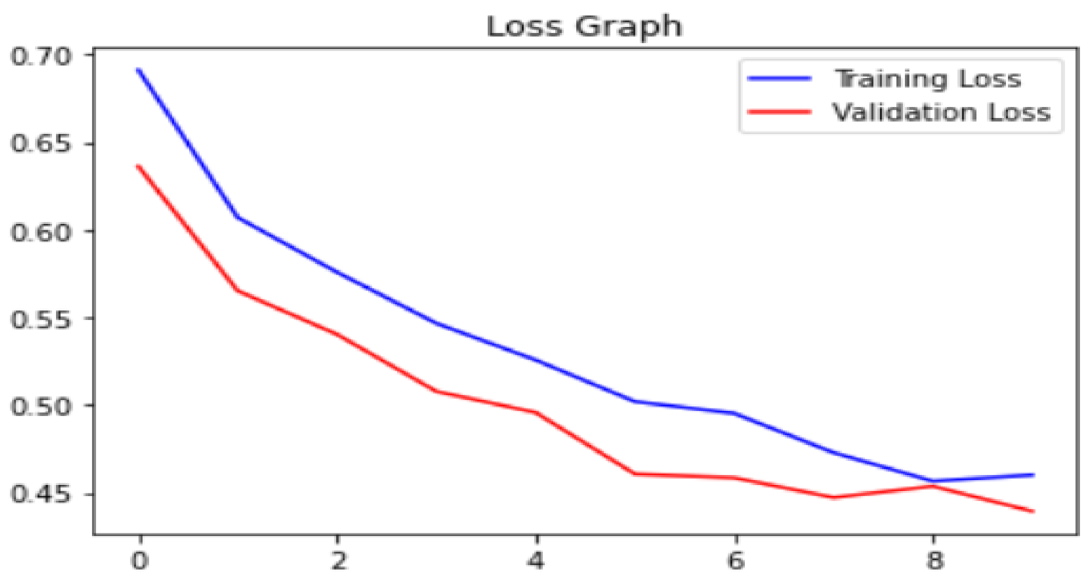


Fig: Accuracy v/s epoch

- From the above graph, we observe that the highest training accuracy is approximately at 79%.
- Highest validation accuracy is approximately at 77.5%.
- As the number of epochs increases, the training and validation accuracy goes on increasing.



<u>Fig: Loss v/s epoch</u>

- From the above graph, it can be observed that, lowest training loss is at 47.50%.
- Lowest validation loss is approximately at 22.50%.
- It can further be observed that both the training loss and validation loss continues to decrease as the number of epochs increases.
- To decrease the loss percentage, we need to change the number of epochs.

PERFORMANCE OF VGG16

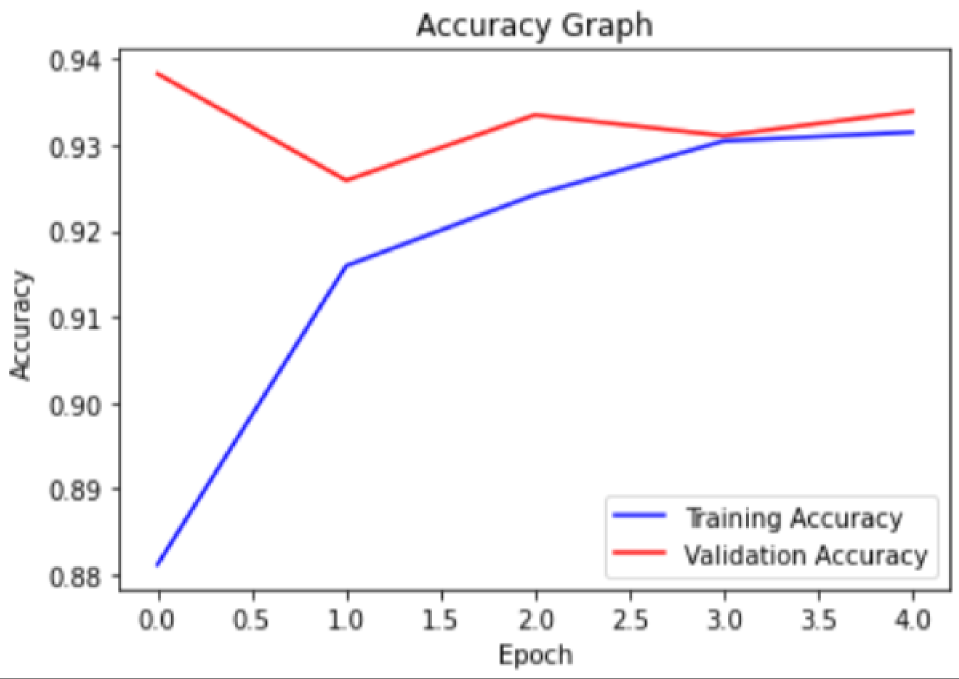


Fig: Accuracy v/s Epoch for VGG16

- VGG16 behaves slightly different as compared to CNN architecture.
- Training accuracy increases up to 93.0% and later remains constant.
- Validation accuracy begins with a higher value and later decreases.
- Highest training accuracy is at 93.00%.
- Highest validation accuracy is at 94.00%.

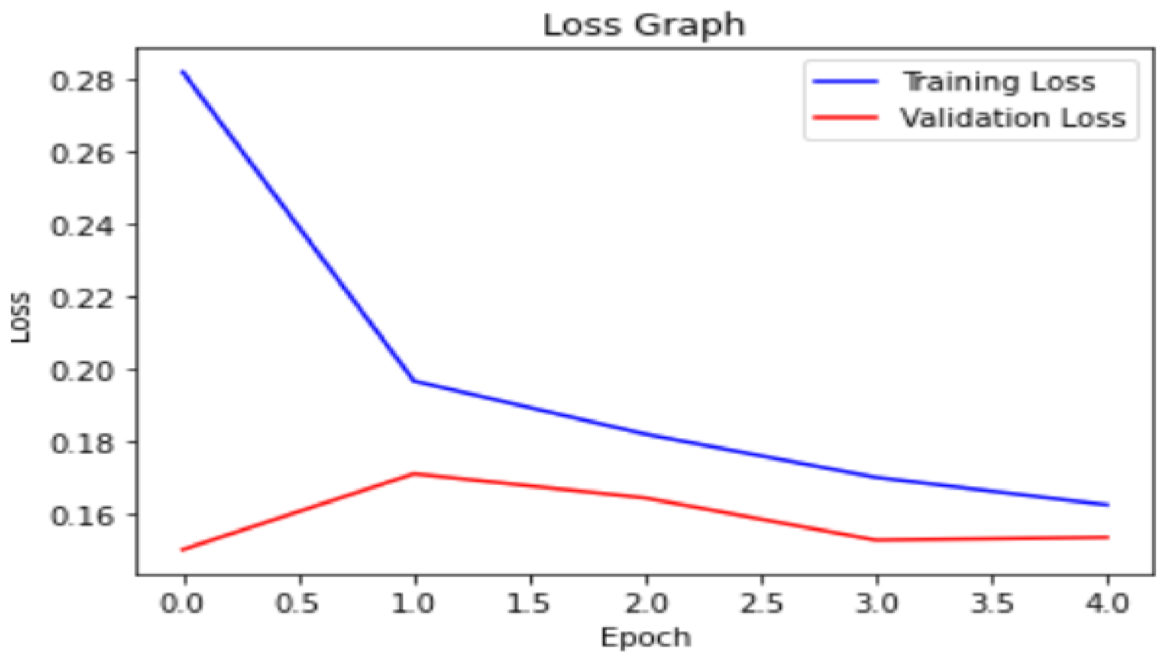


Fig: Loss v/s Epoch for VGG16

- Training loss begins with a higher value and continues to decrease as the number of epochs/iterations increases.
- Validation loss begins with a comparatively smaller value and increases for certain number of epochs and later decreases.
- Lowest training loss is at 17.00%, which is much less than CNN architecture.
- Lowest validation loss is at 16.00%
- As both types of losses are less, VGG16 gives more accurate results as compared to CNN.

CONCLUSION

Hence the given dataset of cats and dogs are carefully processed, analysed and trained using CNN and VGG16 models and their accuracies are determined and compared.

The accuracy is more for VGG16 architecture.

Later for enhancing model Interpretability, I have used Grad-CAM, to highlight important regions in an image.

REFERENCES

1) https://www.geeksforgeeks.org/introduction-convolution-neural-network/ - For CNN
2) https://www.irjet.net/archives/V6/i12/IRJET-V6I1271.pdf - Image classification Research Paper
3) https://arxiv.org/pdf/1610.02391.pdf - Grad-CAM research paper
4) https://towardsdatascience.com/understand-your-algorithm-with-grad-cam-d3b62fce353 - Blog
5) https://keras.io/examples/vision/grad_cam/ - Website

CODE REPOSITORY

I have hereby uploaded my project onto my Github repository.

Name of the repository:

https://github.com/Surya0907/Ai-Bharata-Internship

It contains the report as well as code file which is uploaded from google colaboratory and the problem statement file also being uploaded.