RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING-NANDYAL.
COMPUTER SCIENCE ENGINEERING AND BUSSINESS SYSTEMS

MACHINE LEARNING PROJECT ON CLASSIFICATION OF DOGS VS CATS

- Using CONVOLUTIONAL NEURAL NETWORK(CNN)

-B.Kalpana(22091A3409,CSE&BS,IIBtech)
-G.SwarnaLatha(22091A3437,IIBtech,CSE&BS)
Under the guidance of -Dr.KISHOR KUMAR (HOD OF CSE&BS,RGMCET)

RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING-NANDYAL.
COMPUTER SCIENCE ENGINEERING AND BUSSINESS SYSTEMS

Image Classification – Cat and Dog Images Using CNN

Abstract:

Using a CNN, automated pet doors can be designed to recognize whether it's a cat or a dog approaching, allowing only authorized pets to enter the house while keeping strays or other animals out. And also In animal shelters or veterinary clinics, CNN-based classifiers can be used to quickly identify whether an animal is a cat or a dog. This can help in managing records, tracking lost pets, and facilitating the adoption process. One of the major problem was that of image classification, which is defined as predicting the class of the image. Cat and Dog image classification is one such example of where the images of cat and dog are classified. Deep learning works like the human brain's ability to recognize an object. This paper aims to incorporate state-of-art technique for object detection with the goal of achieving high accuracy. we employ Convolutional Neural Networks (CNNs) to address the challenge of distinguishing between images of dogs and cats. The classification task involves training a deep learning model on a diverse dataset comprising images of both animals. CNNs are well-suited for image classification tasks due to CNNs in achieving high accuracy in discriminating between these two common pets, contributing to the broader their ability to automatically learn hierarchical features from raw pixel data. Our results demonstrate the potential of field of computer vision and image classification.

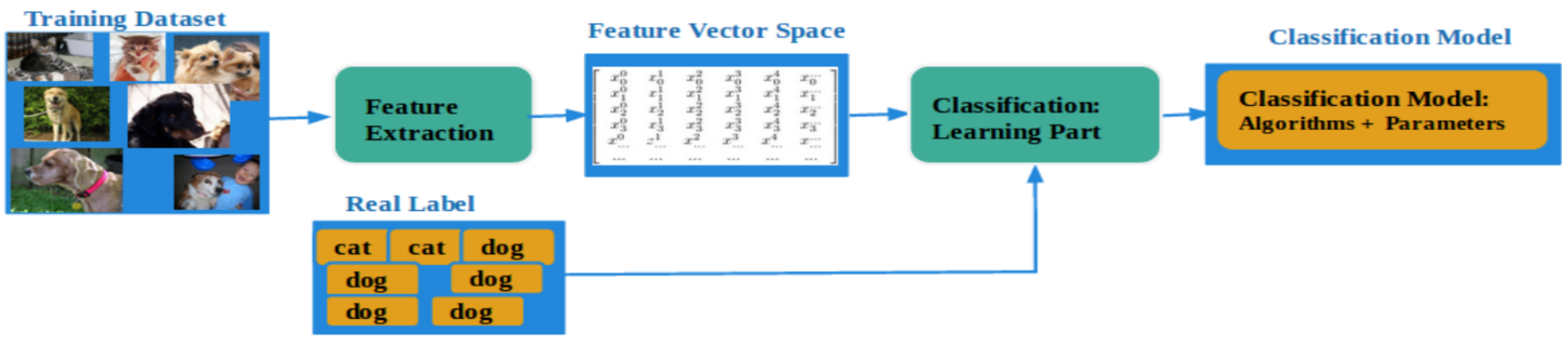
Key Words: Image Classification, Convolutional Neural Network, Keras, Deep Learning, Hyperparameter Tuning, Performance Metrics, Comparative Analysis.

INTRODUCTION

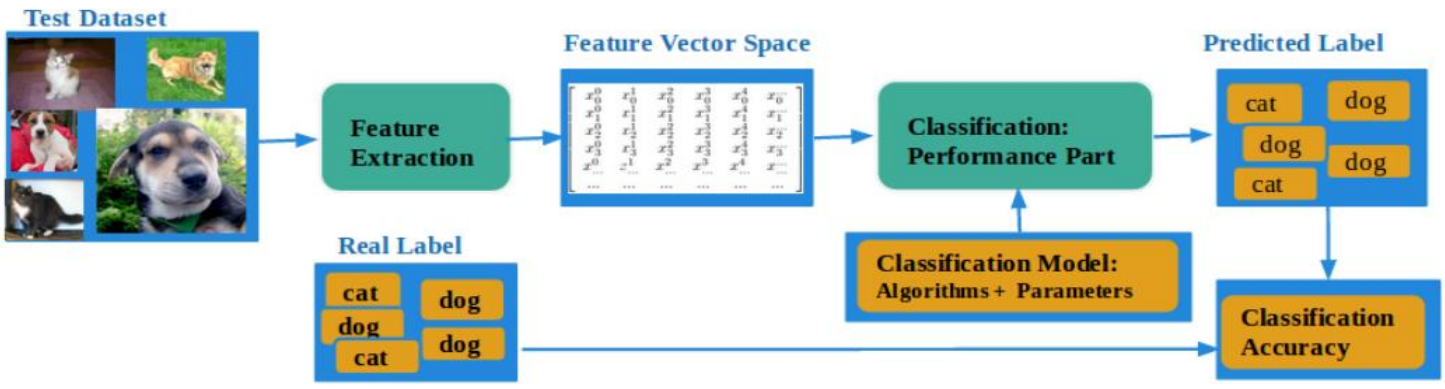
The Dogs vs. Cats image classification has been around for a long time now. The Dogs vs. Cats competition from Kaggle is trying to solve the CAPTCHA challenge, which relies on the problem of distinguishing images of dogs and cats. It is easy for humans, but evidence suggests that cats and dogs are particularly difficult to tell apart automatically. There are so many issues in image classification in real time like Security Systems, pet Identification Systems, Automated Pet doors etc.

Many people has worked or are working on constructing machine learning classifiers to address this problem. In my project I am going to build a **convolutional neural network** to solve the problem and achieve higher performance and better results.

In my project instead of using the Kaggle data set comprising of total 25000 images, I would be working on subset of these images that contain both cat and dog .Keras would used for model building and all the code would be implemented on Anaconda Jupyter Notebook.I also importing tensorflow.



Learning Task Architecture



Testing Performance Architecture of a Model

Convolutional Neural Networks (CNNs): Unraveling the Power of Image Recognition

Convolutional Neural Networks (CNNs), a class of deep neural networks, have revolutionized the field of computer vision by showcasing exceptional prowess in image recognition tasks. Initially devised to mimic the human visual system, CNNs excel in extracting intricate features from raw pixel data, making them indispensable in various domains such as image classification, object detection, and facial recognition.

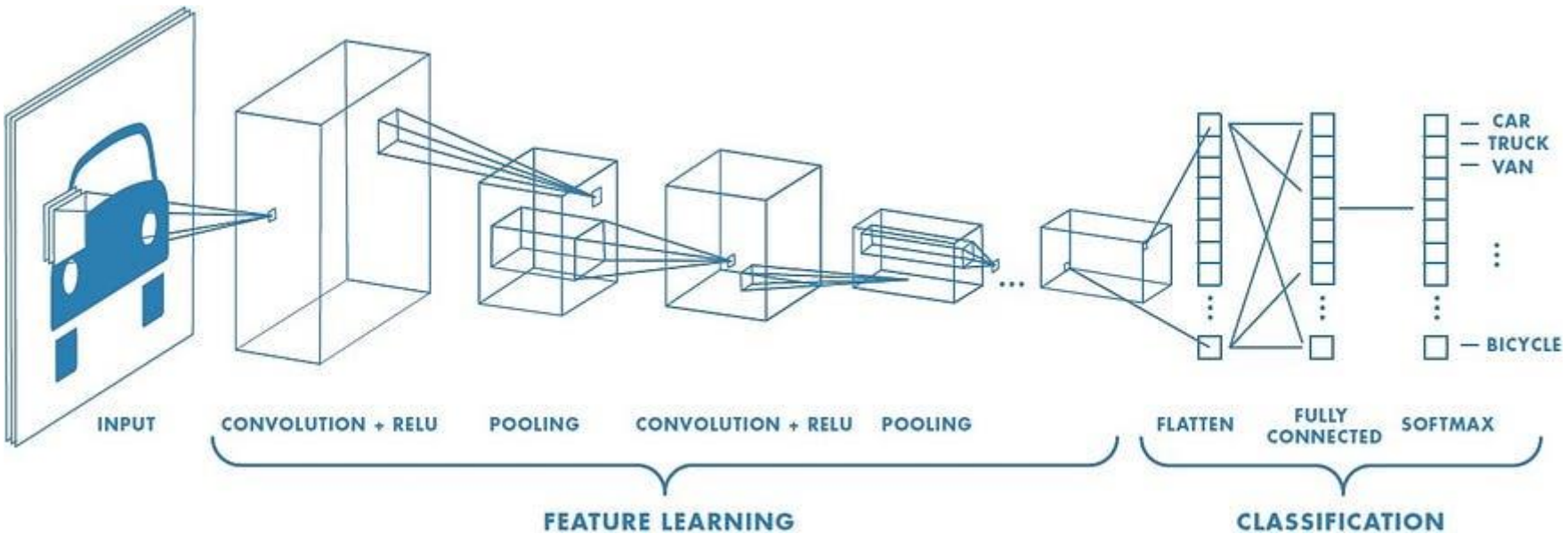
A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product.

Convolutional Layers: The hallmark of CNNs, these layers employ convolutional operations to detect patterns and features within local receptive fields. Filters or kernels are applied across the input image, capturing spatial hierarchies.

Pooling Layers: Often following convolutional layers, pooling layers reduce the spatial dimensions of the feature maps, preserving essential information while enhancing computational efficiency.

Fully Connected Layers: Towards the end of the network, fully connected layers aggregate features for final classification. These layers provide a comprehensive understanding of global relationships within the input data.

The activation function is commonly a RELU layer or Softmax or Sigmoid , and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. The final convolution, in turn, often involves backpropagation in order to more accurately weight the end product.The output neuron gives whether the image is dog or cat.



Methodology:

Dataset and Data Augmentation:

The dataset in keras is divided into folders for each class. The dataset is divided into training and testing set by using validation split at the time of compilation of model. The training set and the test set compose of 2 folders one for cat images and other for dog images. There are some images of each cat and dog for training and some image of each cat and dog for testing. The images are of varying shape and sizes, but in order to train a CNN the images should be of same size. Data Augmentation is being carried out by using the ImageDataGenerator module provided by Keras.The input from the system was an image with a size of 50 x 50 pixels, with 3 channels, namely R, G, and B. The CCN architecture used 3 layers of convolution. The layer was convolution 1 using a 3x3 kernel, filter 16, the second convolution using a 3x3 kernel, filter 32, and the third using a 3x3 kernel, filter 64.

Model Architecture:

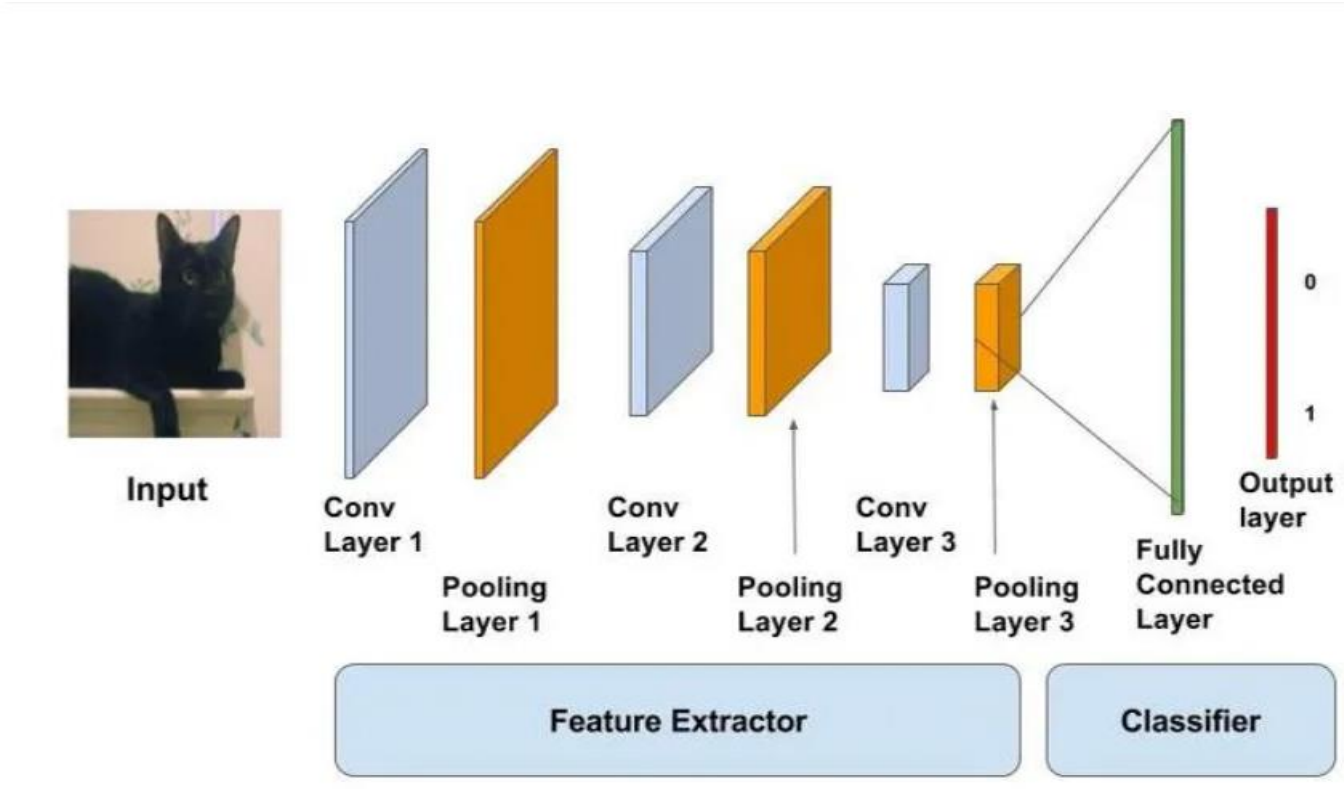
At first ,I am importing data set and some Libraries or packages like Tensorflow,keras,cv2,os for image accesing from file and matplotlib etc. Classifier is the name given to the Sequential model. The model's first layer is a Conv2D layer. Since, it is the first layer of the model, input shape of the images that are going to be supplied to the model is being mentioned. Next layer is a batch normalization layer or convolutional layer . Then one activation layer corresponding to the conv2d layer. Further there is another set of conv2d, batch normalization and activation layer with different number of kernels in

the conv2d layer. After that a max Pooling layer is there and then a dropout layer is there. The same set of layers is again repeated with different number of kernel's and dropout rate. The convolution layers end with this set. Next are the fully connected layer. The Sequential model 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. Activation layer are used to apply the activation function to the output of that layer. The purpose of the activation function is to introduce non-linearity into the output of a neuron. Relu and Sigmoid activation are used in the model. Batch Normalization is used for improving the speed, performance, and stability of artificial neural networks. Batch normalization is a method we can use to normalize the inputs of each layer and achieve faster convergence. Maximum pooling, or max pooling, is a pooling operation that calculates the maximum, or largest, value in each patch of each feature map. The results are down sampled or pooled feature maps that highlight the most present feature in the patch.

Dense layer implements the operation: output = activation (dot(input + kernel) + bias, activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use bias is True).

There are total 62,367,500 total parameters, out of which 43,657,259 are trainable parameters and 18,710250 are non-trainable parameters. The major number parameters are form the conv2d layer. Batch normalization and dense layer also contribute few of the parameters.



Model Compilation and Training:

During the model compilation, the optimizer algorithm, loss function and the list of metrics are parameters which are to be taken care of. Adam is used as the optimization algorithm, sparse_categorical_crossentropy is used as the loss function and accuracy is the only metric used. Early Stopping and ModelCheckPointer callbacks are used to prevent overfitting and save the best state of the model. These callbacks are mentioned as a list during the training. Sequential models fit_generator() is used to train the model. Model is trained for 15 and 25 epochs with EarlyStopping.

Model Evaluation and Results:

The model trained for 15 epochs after which it stopped due to the presence of Early Stopping callback which had the patience parameter set to 5. The training accuracy kept on increasing but the validation accuracy started to decrease which might be due to overfitting. That was the reason Early Stopper check pointer was used to prevent results obtained due To overfitting. Below is the table showing the end result of training i.e train accuracy, test accuracy, train loss, test loss and epochs.

The graph below shows training accuracy and testing accuracy vs the number of epochs and also shows the train loss and test loss vs the number of epochs.

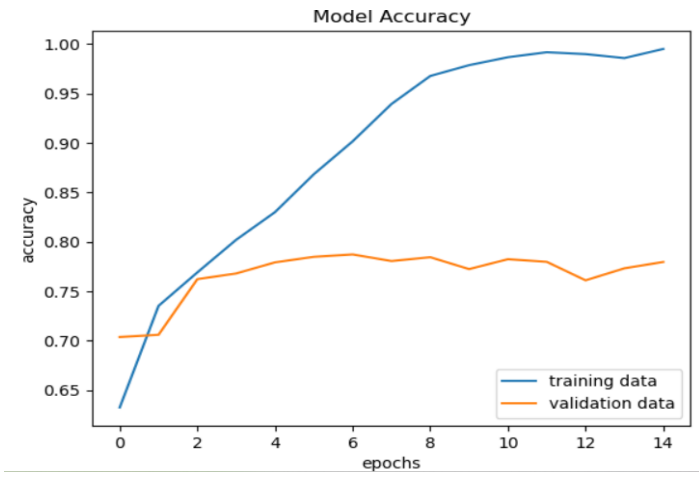
And again we are trying to trained with more epochs like 5,10 etc..The training accuracy may be increasing so…we did again with more epoch rate.Then finally rate of accuracy of model Increases.

For epoch size:15

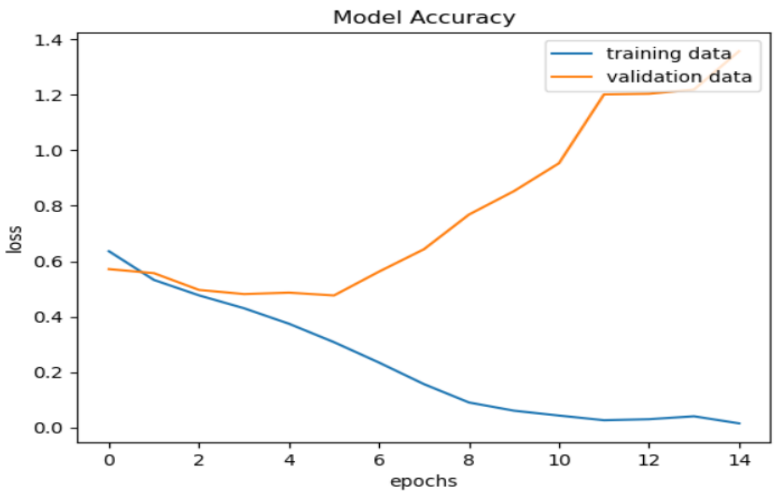
```
history=model.fit(x,y,epochs=15,validation_split=0.3)
```

```
Epoch 1/15
546/546 [==============================] - 157s 286ms/step - loss: 0.6358 - accuracy: 0.6326 - val_loss: 0.5715 - val_accuracy: 0.7038
Epoch 2/15
546/546 [==============================] - 163s 299ms/step - loss: 0.5323 - accuracy: 0.7353 - val_loss: 0.5568 - val_accuracy: 0.7061
Epoch 3/15
546/546 [==============================] - 160s 292ms/step - loss: 0.4768 - accuracy: 0.7692 - val_loss: 0.4964 - val_accuracy: 0.7623
Epoch 4/15
546/546 [==============================] - 169s 309ms/step - loss: 0.4303 - accuracy: 0.8021 - val_loss: 0.4815 - val_accuracy: 0.7681
Epoch 5/15
546/546 [==============================] - 164s 300ms/step - loss: 0.3744 - accuracy: 0.8301 - val_loss: 0.4866 - val_accuracy: 0.7793
Epoch 6/15
546/546 [==============================] - 2483s 5s/step - loss: 0.3079 - accuracy: 0.8684 - val_loss: 0.4764 - val_accuracy: 0.7849
Epoch 7/15
546/546 [==============================] - 69s 126ms/step - loss: 0.2344 - accuracy: 0.9017 - val_loss: 0.5624 - val_accuracy: 0.7873
Epoch 8/15
546/546 [==============================] - 89s 164ms/step - loss: 0.1565 - accuracy: 0.9394 - val_loss: 0.6432 - val_accuracy: 0.7806
Epoch 9/15
546/546 [==============================] - 126s 231ms/step - loss: 0.0904 - accuracy: 0.9678 - val_loss: 0.7683 - val_accuracy: 0.7845
Epoch 10/15
546/546 [==============================] - 134s 245ms/step - loss: 0.0608 - accuracy: 0.9787 - val_loss: 0.8528 - val_accuracy: 0.7725
Epoch 11/15
546/546 [==============================] - 174s 318ms/step - loss: 0.0433 - accuracy: 0.9867 - val_loss: 0.9535 - val_accuracy: 0.7825
Epoch 12/15
546/546 [==============================] - 167s 306ms/step - loss: 0.0264 - accuracy: 0.9918 - val_loss: 1.2017 - val_accuracy: 0.7798
Epoch 13/15
546/546 [==============================] - 175s 320ms/step - loss: 0.0301 - accuracy: 0.9899 - val_loss: 1.2039 - val_accuracy: 0.7611
Epoch 14/15
546/546 [==============================] - 158s 289ms/step - loss: 0.0406 - accuracy: 0.9859 - val_loss: 1.2191 - val_accuracy: 0.7733
Epoch 15/15
546/546 [==============================] - 145s 266ms/step - loss: 0.0152 - accuracy: 0.9951 - val_loss: 1.3574 - val_accuracy: 0.7797
```

MODEL ACCURACY -78%



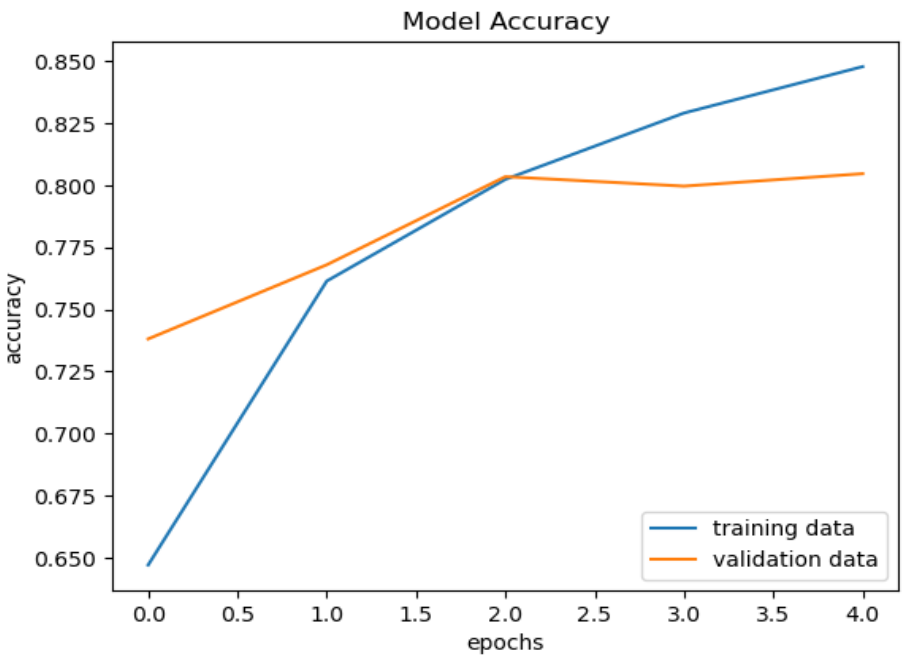
validation split is 0.3 and epochs=15

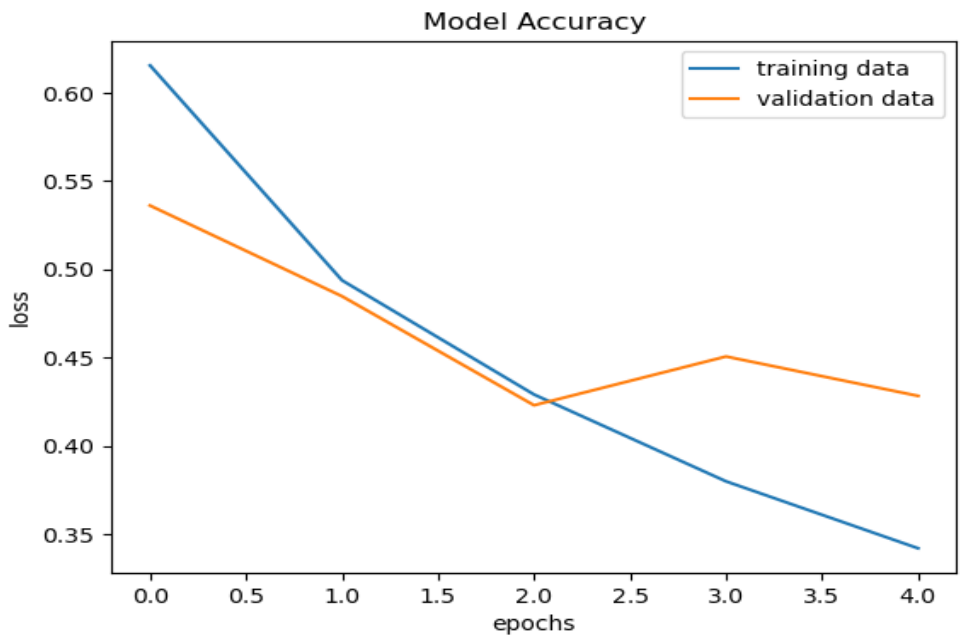


Model Loss

CNN Model

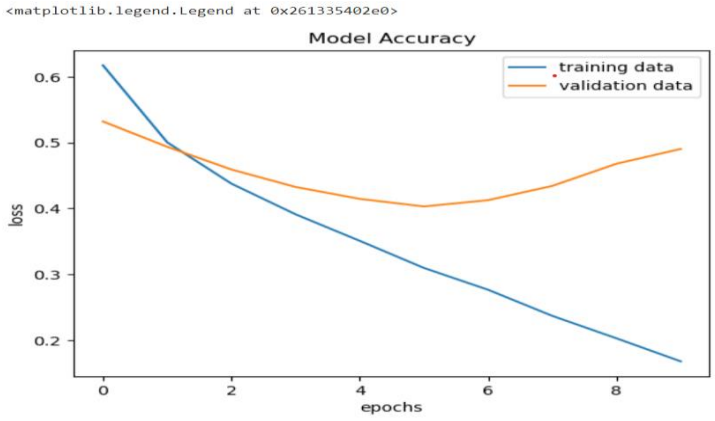
For 5 epochs:

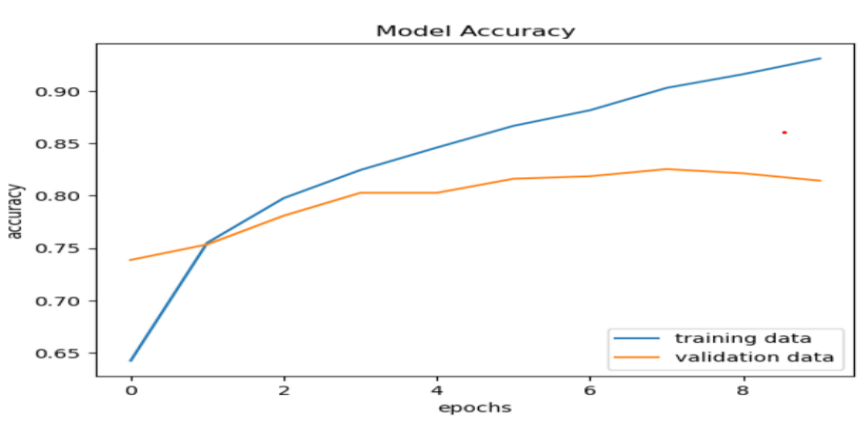




For 10 epochs:

```
<matplotlib.legend.Legend at 0x261335402e0>
```





From Above graphs,We attain a different accuracy values at different epochs. The final accuracy was obtained by initializing the model with the weights that were stored during training. Final accuracy of 83.74% was obtained on the testing data.

Conclusion:

In conclusion, the implemented Convolutional Neural Network (CNN) effectively classified images of dogs and cats with a notable accuracy. Despite successful outcomes, challenges such as data imbalances were encountered. Future improvements may involve refining hyperparameters, exploring advanced architectures, and considering real-world applications, such as pet identification systems, security systems, Automated Pet Doors etc. The project highlights the potential of CNNs in image classification tasks, with opportunities for further enhancements and practical implementations. Despite of using only a subset of the images an accuracy of 84.7% was obtained.