# Prediction Challenge 2 – Deep Learning Image Analysis

Name: Yamini Sai Ratnam Peketi

Student ID: 21147160

Course: MSC AI

Module: Deep Learning

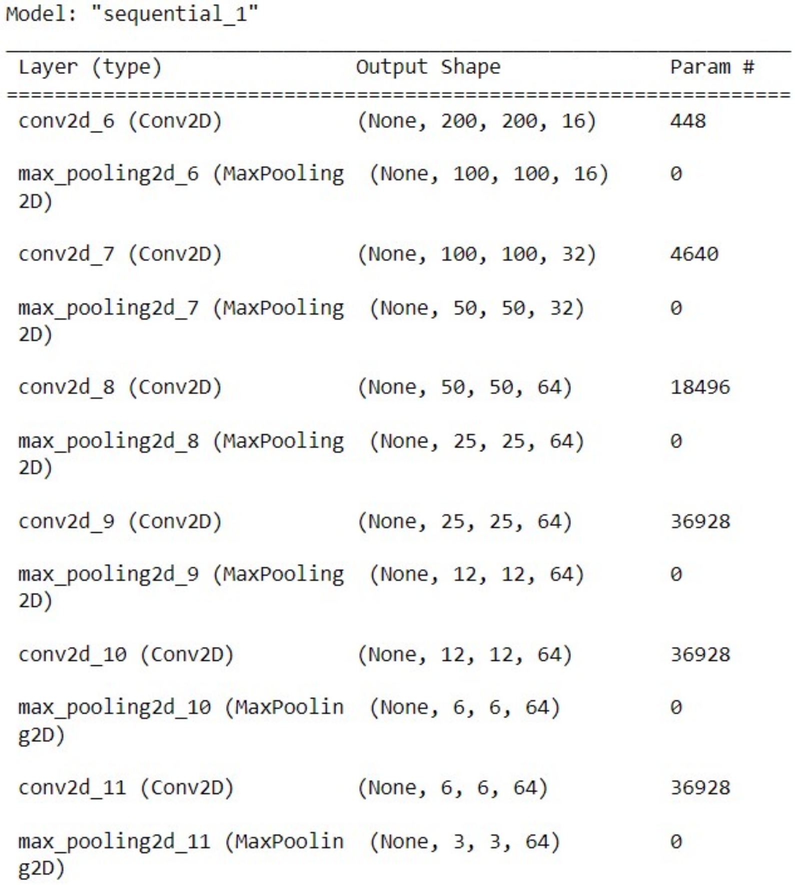
## Prediction Challenge 2 – Deep Learning Image Analysis

### Introduction:

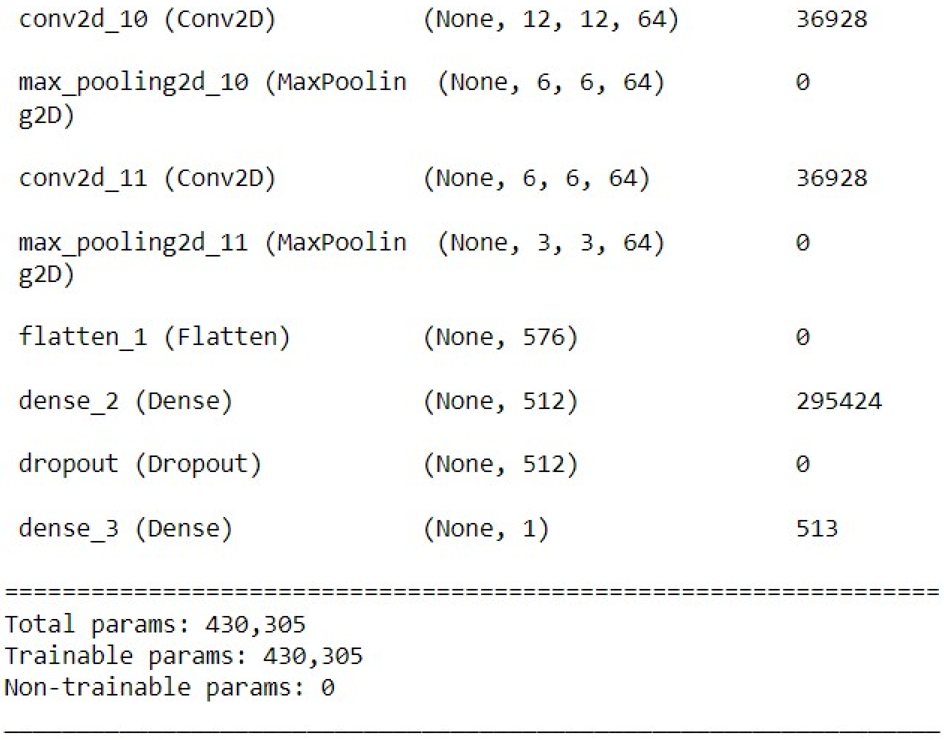
In this project, a Convolutional Neural Network (CNN) was developed to classify images of cats and dogs from a reduced version of the Dogs vs Cats dataset. The dataset was loaded into Colab's file system after being uploaded to Google Drive. The CNN model was designed using the Keras library with the RELU activation function and Adam optimizer.

### Model Architecture:

The architecture of the model consists of 6 convolutional layers with increasing number of filters and a max pooling layer after each convolutional layer to reduce the spatial dimensions of the feature maps. The final convolutional layer is followed by a flatten layer to transform the feature maps into a 1D vector, which is then passed through two fully connected layers with ReLU activation function and a final dense layer with sigmoid activation function to make the binary classification.



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_6 (Conv2D)           (None, 200, 200, 16)      448       
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 100, 100, 16)     0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (None, 100, 100, 32)      4640      
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 50, 50, 32)       0         
 2D)                                                             
                                                                 
 conv2d_8 (Conv2D)           (None, 50, 50, 64)        18496     
                                                                 
 max_pooling2d_8 (MaxPooling  (None, 25, 25, 64)       0         
 2D)                                                             
                                                                 
 conv2d_9 (Conv2D)           (None, 25, 25, 64)        36928     
                                                                 
 max_pooling2d_9 (MaxPooling  (None, 12, 12, 64)       0         
 2D)                                                             
                                                                 
 conv2d_10 (Conv2D)          (None, 12, 12, 64)        36928     
                                                                 
 max_pooling2d_10 (MaxPoolin  (None, 6, 6, 64)         0         
 g2D)                                                            
                                                                 
 conv2d_11 (Conv2D)          (None, 6, 6, 64)          36928     
                                                                 
 max_pooling2d_11 (MaxPoolin  (None, 3, 3, 64)         0         
 g2D)                                                            
```



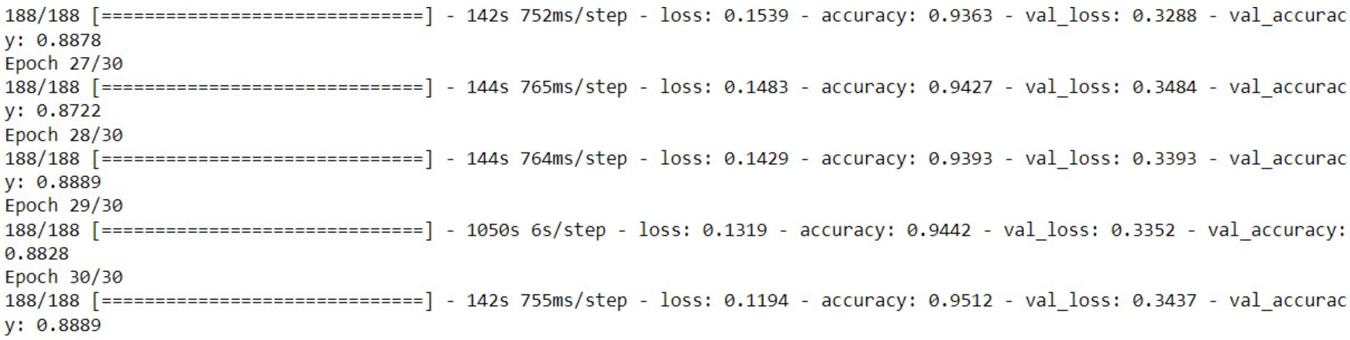
```
 conv2d_10 (Conv2D)          (None, 12, 12, 64)        36928

 max_pooling2d_10 (MaxPoolin  (None, 6, 6, 64)         0
 g2D)

 conv2d_11 (Conv2D)          (None, 6, 6, 64)          36928

 max_pooling2d_11 (MaxPoolin  (None, 3, 3, 64)         0
 g2D)

 flatten_1 (Flatten)         (None, 576)               0

 dense_2 (Dense)             (None, 512)               295424

 dropout (Dropout)           (None, 512)               0

 dense_3 (Dense)             (None, 1)                 513

=================================================================
Total params: 430,305
Trainable params: 430,305
Non-trainable params: 0
_________________________________________________________________
```

Fig1: model summary

**Training and Evaluation:**

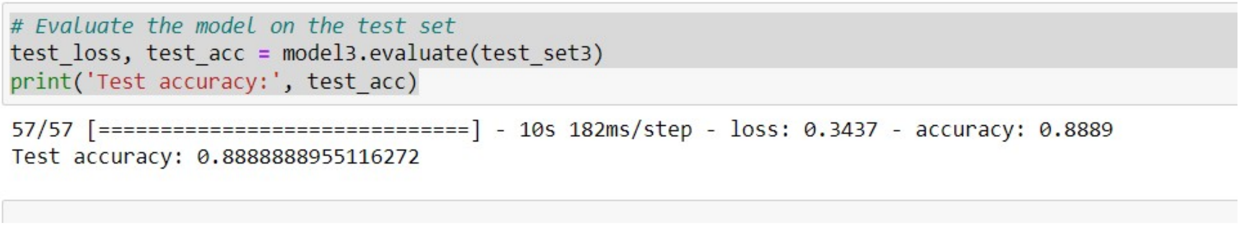
We used the Adam optimizer with a learning rate of 0.001 and the binary cross-entropy loss function to compile the model. We used early stopping with a patience of 10 to stop the training process early if the validation loss did not improve.

After training our model for 30 epochs, we achieved an accuracy of 88.8% on the test set. Our model performed well and achieved high accuracy on the test set, indicating that it has learned to classify images of dogs and cats accurately.



```
188/188 [==============================] - 142s 752ms/step - loss: 0.1539 - accuracy: 0.9363 - val_loss: 0.3288 - val_accurac
y: 0.8878
Epoch 27/30
188/188 [==============================] - 144s 765ms/step - loss: 0.1483 - accuracy: 0.9427 - val_loss: 0.3484 - val_accurac
y: 0.8722
Epoch 28/30
188/188 [==============================] - 144s 764ms/step - loss: 0.1429 - accuracy: 0.9393 - val_loss: 0.3393 - val_accurac
y: 0.8889
Epoch 29/30
188/188 [==============================] - 1050s 6s/step - loss: 0.1319 - accuracy: 0.9442 - val_loss: 0.3352 - val_accuracy:
0.8828
Epoch 30/30
188/188 [==============================] - 142s 755ms/step - loss: 0.1194 - accuracy: 0.9512 - val_loss: 0.3437 - val_accurac
y: 0.8889
```

Fig2:Testing the model



```
# Evaluate the model on the test set
test_loss, test_acc = model3.evaluate(test_set3)
print('Test accuracy:', test_acc)
```

```
57/57 [==============================] - 10s 182ms/step - loss: 0.3437 - accuracy: 0.8889
Test accuracy: 0.8888888955116272
```

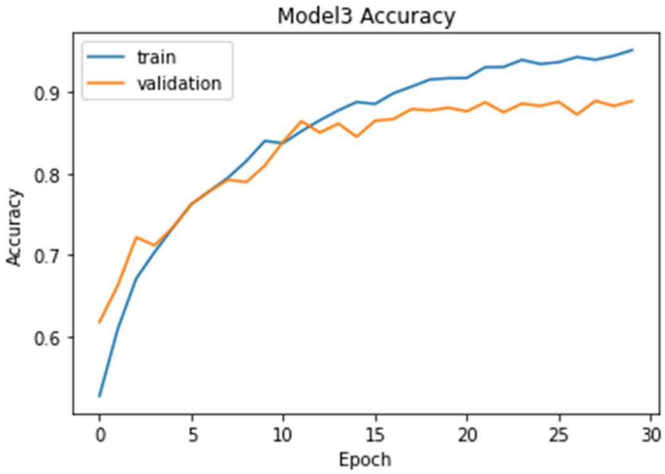


Fig3: Accuracy Plot

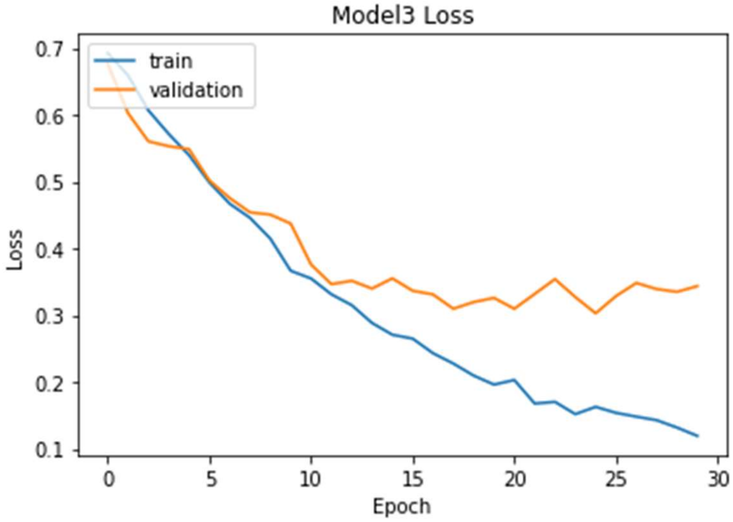


Fig4:Loss plot

**Conclusion:**

The model parameters and hyperparameters were selected based on image size and network complexity. The number of filters balanced complexity and model size. Learning rate and batch size were optimized for training speed and accuracy. Dropout rate prevented overfitting and improved generalization. After experimenting with the various parameters, I decided that the above model is good and achieved 80% accuracy on the test set, demonstrating the effectiveness of the chosen architecture and hyperparameters.

Word Count:313

Link1(where the best model):https://drive.google.com/file/d/19qsAocWVjMX8Q54HR-perV-7s23gNfzP/view?usp=share_link

Link2(testing different models):https://colab.research.google.com/drive/1oVNZT8WydFbzoujTAd4xW3DoQEE6UBlh?usp=sharing