Dog Breeder Classifier Report

Project Overview:

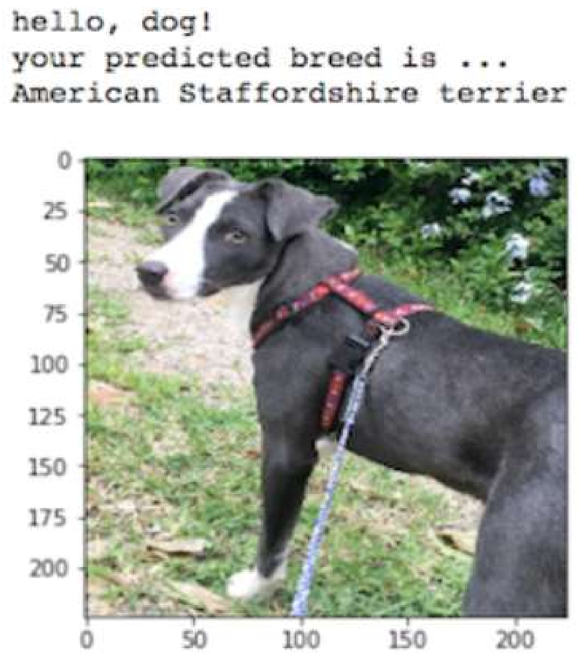
The project focusses on Convolutional Neural Networks (CNN) to classify images into different categories. The significant part is taking real-world images and classifying into different dog breeds using CNN models. The different dog breeds are difficult to identify even for the human eye. The model should be able to distinguish between different dog images and classify according to their breeds. The model should also recognize and distinguish human images from the dog images. It is challenging as dog breeds are difficult to identify even for a normal human eye. I have referenced the work on Very Deep Convolutional Networks for large scale image recognition[1] by Karen Simonyan and Andrew Zisserman.

Problem Statement:

From a given set of images:

a) Identify if the image is of a dog or human.
b) If the image is of a dog, identify the breed of dog.
c) If the image is of a human, identify the breed of dog a human face resembles.
d) If the image is not human or dog, give an error message.

Figure 1: Sample Expected Output



```
hello, dog!
your predicted breed is ...
American Staffordshire terrier
```

Method and Data Analysis:

The steps used are outlined as below:

1. Pre-process given data
2. Identifying Human and Dog faces

3. Transform images
4. Implement a CNN classifier to detect dogs (from scratch)
5. Improve accuracy with a better CNN classifier (Transfer Learning)
6. Predict the dog breeds
7. Test on images other than dataset

Data:

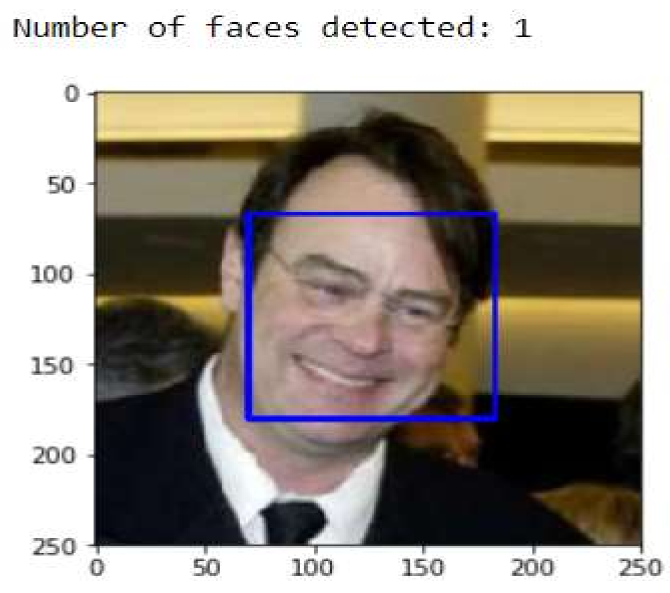
We are given two datasets - one of human images (13233 images) and other of dog images (8351 images). The dog data is divided into three subsets of training, validation and test images with 133 different classes. The training dataset is used to train the model. The validation data is used to review the progress of the model i.e. how good a model has been trained. The test dataset which is kept separately, should be used to check the accuracy of the model after it is well-trained.

Benchmark Models:

Face Detection Test :

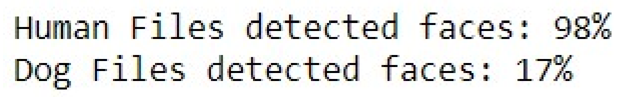
For human face detection, Open CV implementation of Haar Cascades is used. First the color image is converted to grayscale. This grayscale image is run through the classifier for face detection. Each detected face has four parameters specifying the x and y coordinates of the bounding box, the width and height of the face. This face detector is used on our human files and dog files data (first 100 images). Out of the 100 human images, it correctly detected 98% human faces. However, it detected 17 human faces in 100 dog images. The results are shown below for the face detector:

Figure 2: Face Detector Test on a Test Image



```
Number of faces detected: 1
```

Figure 3: Face Detector Accuracy Results

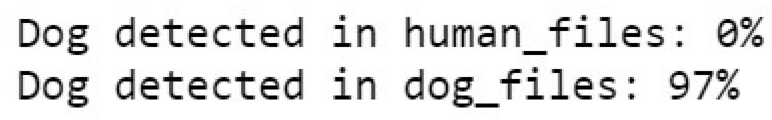


```
Human Files detected faces: 98%
Dog Files detected faces: 17%
```

Dog Detection Test:

For dog detection, VGG-16 a pre-trained model is used. The transformed image is run through the VGG-16 pre-trained model which returns an index corresponding to a class in ImageNet. For classes 151 to 268 in ImageNet correspond to dogs. The dog detector function counts number of index predictions within this range of 151 to 268. From a set of 100 dog images, the function correctly identifies 97 images. It does not recognize any dog amongst the 100 human images. The results are show below for the dog detector:

Figure 4: Dog Detector Test Accuracy Results

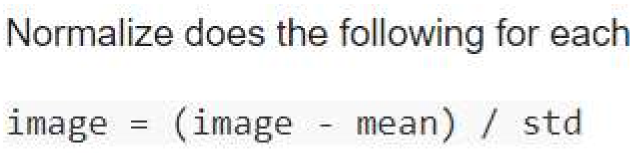


```
Dog detected in human_files: 0%
Dog detected in dog_files: 97%
```

Transform Images:

All the images used as input need to be same size and shape. Hence I have resized each image to a size of 256 x 256 before center cropping to a size of 224 x 224 to all train, validation and test datasets. I have used Image Augmentation for my training data using Random Horizontal Flip and Random Rotation of 10 degrees. The images are further normalized between 0-1 using mean and standard deviation.

Figure 5: Formula to Normalize each image



Normalize does the following for each

```
image = (image - mean) / std
```

VGG16 Architecture:

For large image datasets, it is advantageous to use VGG16 model for image classification. As seen in Figure 1, the VGG network has a characteristic pyramidal shape, where the bottom layers close to the image are wide and the top layers are deep. VGG contains subsequent convolutional layers followed by pooling layers. The pooling layers are responsible for making the layers narrower.

Figure 6: The VGG16 architecture is depicted below

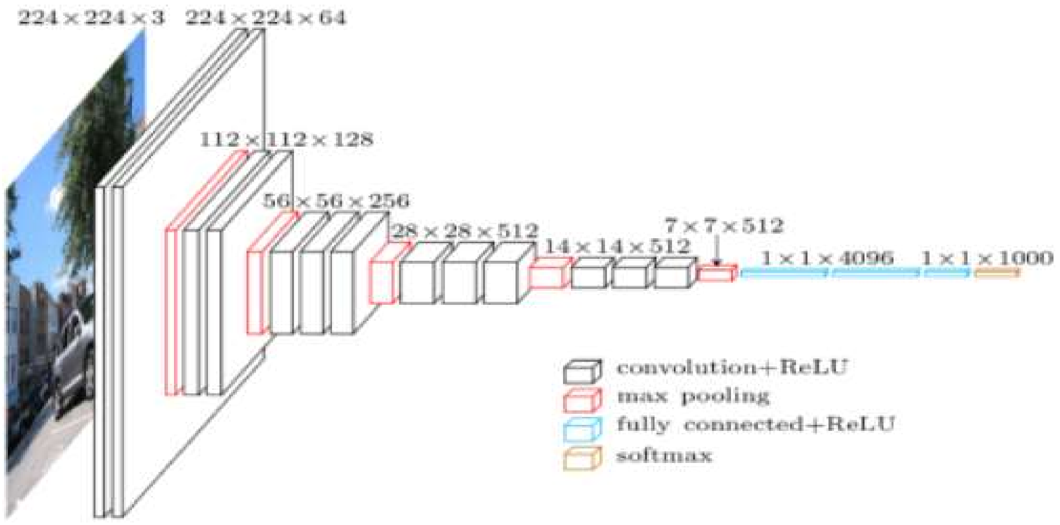
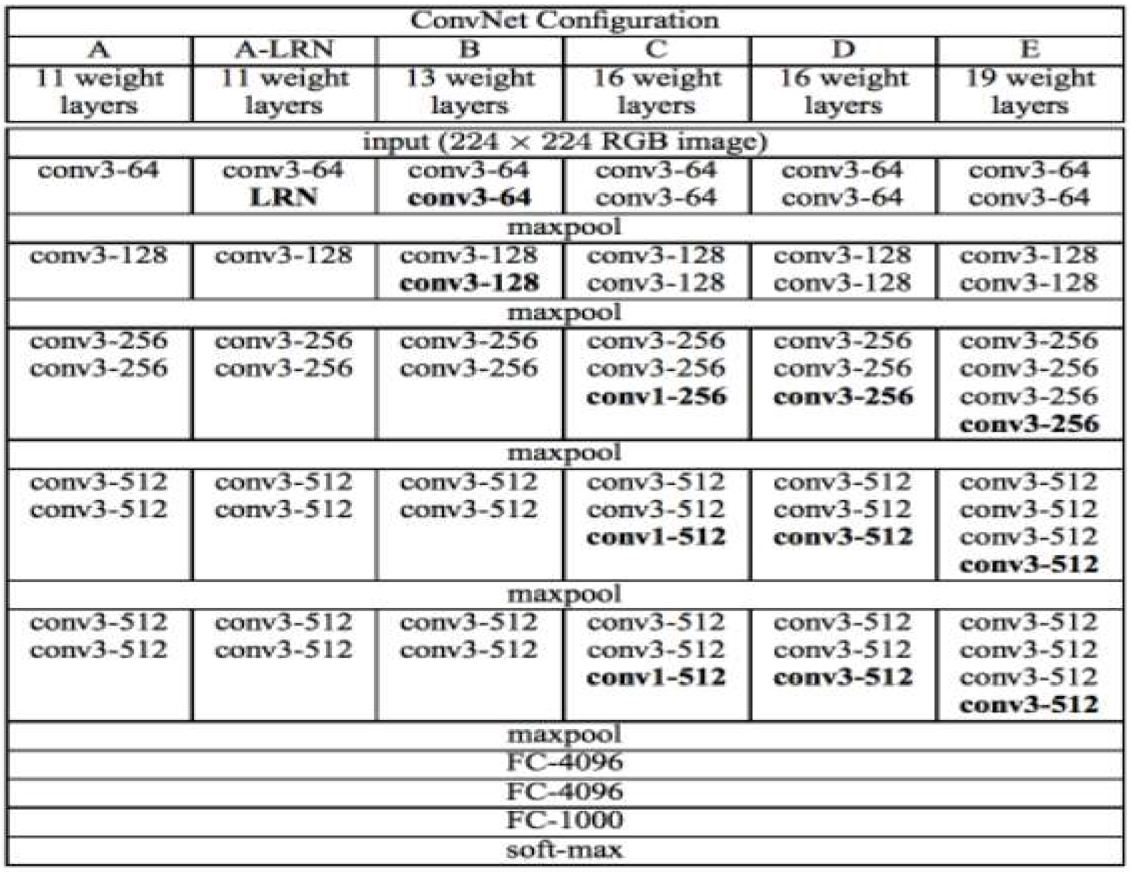


Table 1: Typical VGG architecture with different depths



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

CNN Classifier from Scratch :

First I have used CNN classifier from scratch to classify dog breeds. The classifier consists of 3 convolutional layers and two fully-connected layers. First two convolutional layers have a kernel size of 3, stride of 2 and padding of 1. This will downsize the input image by 2. The max pooling layer with stride 2 will further downsize the image by 2. The third convolutional layer with kernel size of 3 and no stride will not downsize the image. I have used a dropout of 0.25 for optimum accuracy which prevents overfitting. Dropout is typically between 0.20-0.50 which removes prevents overfitting due to a layer's over-reliance on a few of its inputs. Because these inputs aren't always present during training (i.e. they are dropped at random), the layer learns to use all of its inputs, improving generalization.

After three convolutional and maxpool layers, we end up with a 128x7x7 feature map. These feature maps are then flattened to a vector of length 128x7x7 and fed into the fully connected (FC) layers for classification. Since we have only 133 classes, the two fully connected layers reduced the number of layers per node to 133 (not 1000 as given in the paper[1]).

CNN Classifier using Transfer Learning:

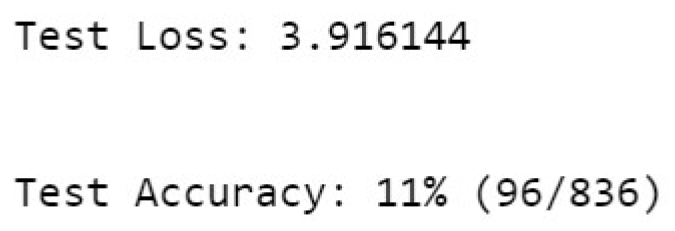
The CNN model from scratch showed reasonable performance, much better than a random guess with probability 1/133. To improve the performance of the model, it is better to use transfer learning. Transfer Learning involves using a pre-trained model and modifying its classifier head to suit a given problem. Since VGG-16 itself is a pre-trained model, I decided to use the same model and change its classifier to improve performance. The datasets transformations and weights are kept constant.

The transfer learning classifier, consists of 3 fully connected layers with two dropouts and two ReLu activations in between, will reduce the number of parameters.

Evaluation Metrics:

The models with CNN classifier from scratch and transfer learning were both trained for an epoch of 20 for comparison purposes. The test accuracy of CNN classifier from scratch was 11% as shown in Figure 6:

Figure 7: Test Accuracy of CNN Classifier from scratch

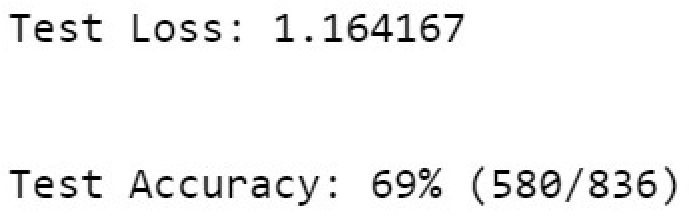


```
Test Loss: 3.916144


Test Accuracy: 11% (96/836)
```

The transfer learning classifier showed improved accuracy of 69% as seen in Figure 7:

Figure 8: Test Accuracy of CNN Classifier (Transfer Learning)



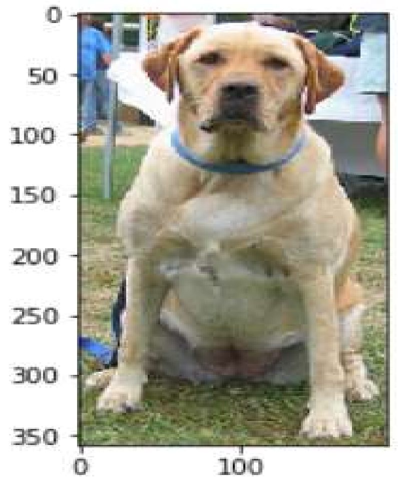
```
Test Loss: 1.164167


Test Accuracy: 69% (580/836)
```

Results:

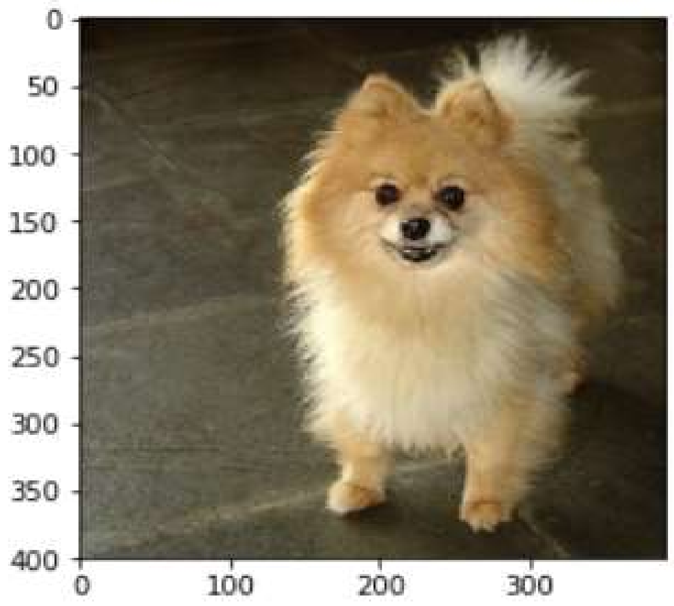
The classifier was also tested on custom random images- both dog and human images. The classifier was correctly able to distinguish human face from a dog and identify the breed of dog. Some of the results are shown below:

Figure 9: Sample Result of Dog Breed Identification



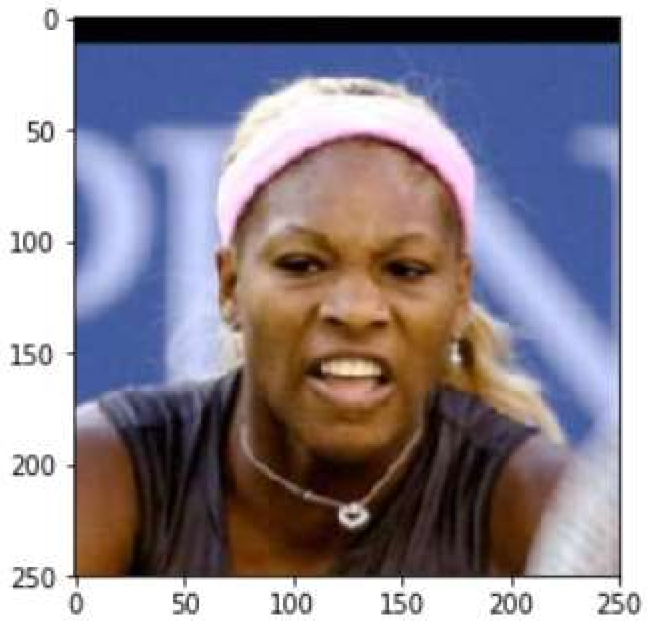
```
Dog Detected!
It looks like a Labrador retriever
```

Figure 10: Sample Result of Dog Breed Identification



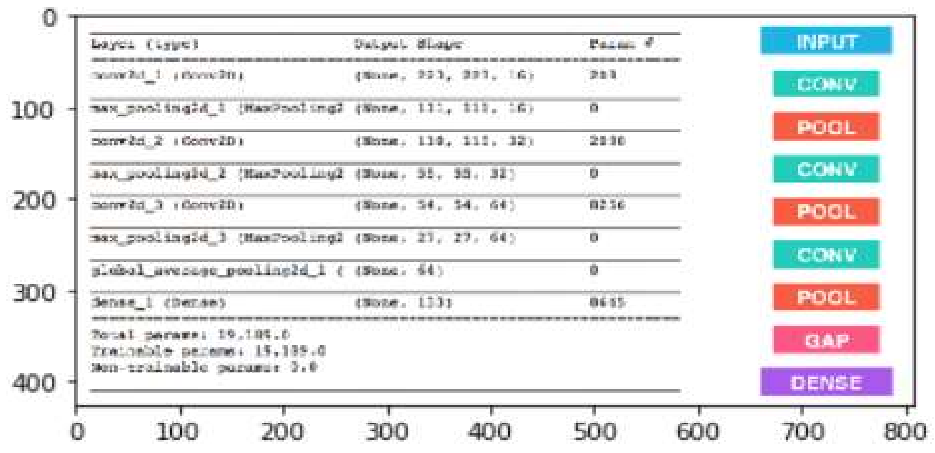
```
Dog Detected!
It looks like a Pomeranian
```

Figure 11: Sample Result of Human Identification



```
Human Detected!
If you were a dog, you may look like a Dogue de bordeaux
```

Figure 12: Sample Result of Random Image



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 223, 223, 16)      208
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 110, 110, 32)      2080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 55, 55, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 54, 54, 64)        8256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 27, 27, 64)        0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 133)               8645
=================================================================
Total params: 19,189.0
Trainable params: 19,189.0
Non-trainable params: 0.0
_________________________________________________________________
```

```
Error in detection.
```

Conclusion:

The model with transfer learning classifier showed test accuracy of 65% on being trained with 15 epochs. The classifier was able to correctly detect the human face and a dog face at almost all times. It could identify an image without human face or dog to show an error message.

Scope of Improvement:

The model accuracy can be further improved by incorporating –

1. Hyper-parameter tuning – Altering the learning rate, initial weights, drop-outs, optimizers, and batch size will be helpful to improve performances.
2. More Image augmentation – More random flipping, rotating etc. could improve the performance.
3. Two detectors – The two detectors ideally should never fail to distinguish between humans and dogs. The model should be able to correctly distinguish the primary image in focus and ignore the background Eg. The image of human in focus and dog in background or image with multiple dogs should always be correctly detected in an app.
4. Ensembles – Instead of single prediction networks, power of ensemble learning could be used to improve performance

References:

1. ***Very Deep Convolutional Networks for Large-Scale Image Recognition***
2. ***Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps***
3. ***https://towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch-dd09190245ce***
4. ***https://www.analyticsvidhya.com/blog/2017/08/10-advanced-deep-learning-architectures-data-scientists/***