README_1.md

Machine Learning Engineer Nanodegree

Capstone Project : Dog Breed Classification

Rahul Madan Raju
February , 2020

I. Definition

Classification is the process of classifying things based on the similarity of features. It is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observations. Classification is one of the several methods intended to make the analysis of very large datasets effective.

Some of the examples of classification using machine learning are:

- Sign Language Identification
- Speech Recognition
- Object Classification

Here, we will be discussing: Dog Breed Classification

Project Overview

In Dog Breed Classification, we will be classifying dogs based on their breeds.

We see the dataset contains images of dogs and humans to which we have to classify dogs based on their breeds. Then why Humans? the images of the humans are used to see what category of dog breeds will they be classified (for fun purpose)

Also, when observed, we categorize such works to the field of Computer Vision and Machine Learning, to which there are various works carried on in the above project.

Recently in 2019, Punyanuch Borwarnginn et al proposed the work on dog breed classification using the different approaches to classify them based on their breeds to tackle population control, disease breakout, vaccination control and legal ownership. He and his co-workers used

- Histogram Oriented Gradient
- Convolutional Neural Network using Transfer Learning for the classification purpose

On making a comparative study, they found that the Neural Nets had a better performance compared to the HOG descriptor.

Problem Statement

Here the goal is to create a Dog Breed Classifier and build an application for the same. The tasks involved are:

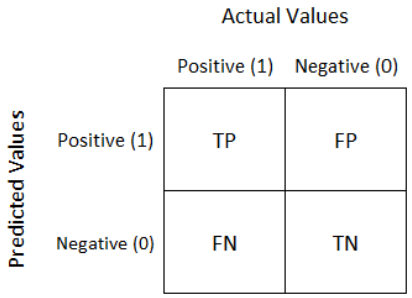
- Download and process the Images of the Dogs and Humans
- Detect the Dogs and Humans using the detector algorithms such as haar cascades and local binary pattern cascades
- Build and train a classifier to classify dog breeds using a pre-trained model (VGG-16 or RESTNET50) and custom model
- Also, train the model using transfer learning with an efficiency to be used for application
- Using the App, predict the breed of the dog and also the category of dog breed the human resembles

It is an application that can be quite handy to recognize the breeds of unknown dogs for the user and also have fun by creating a resemblance of a dog to the given human image

Metrics

The evaluation metrics that can be used to evaluate the performance of the machine learning models are:

- Accuracy: The ratio of correct predictions to the total size of the data (i.e. (TP+TN)/Data Size)
- Recall: The ratio of true positives to the true positive and false negative (i.e. TP/(TP+FN))
- Precision: The ratio of true positives to the true positive and false positive (i.e. TP/(TP+FP))



$$PRE = \frac{TP}{TP + FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

In our case, we will be using the accuracy as the metric of measurement to evaluate the performance of the model.

There are cases where the metrics such as accuracy might be misleading in the case of imbalance data. One such example is the fraudulent data in which we dealt in the earlier studies of this course. We have seen the percentage of fraud cases being drastically low compared to non fradulent cases in the data. However, in this data of dog breeds, as we observe in the visualization part, the data in each class is beyond certain threshold and looks ideally balanced to carry the classification work. Therfore, the accuracy can be used as the metric of evaluation in our work.

II. Analysis

Data Exploration

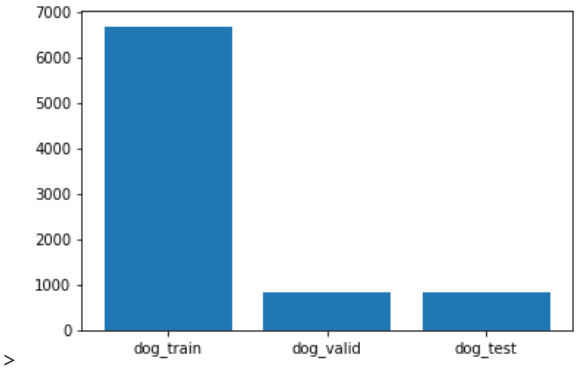
Here, in the Dog Breed Classification, the dataset contains the images of Dogs and Humans. There are a total of 133 breeds, 8351 images for dogs. Using these images as data, it has to be processed according to our needs and a model has to be designed to train our machine.

On making the analysis on the data, we see that the resolution of the images are not the same for all images of dogs in thier respective breeds. The images have a varied resolution and they need to be resampled based on the requirement of our model. Here are some expamples of the images discussed in terms of resolution:

The images shown as the examples above belong to the same breed of dog. The first image has a resolution of 762x768 and the second image has the resolution of about 700x525. We see that the resolution of the images are not the same for the given images. In such cases, we might need to resample our images based on the needs of the model before training it.

In our case, we observe that the split of train and test data is 90%-10%, i.e. 90% for training and 10% for testing purposes. In the training data, we have reserved another 10% for validation. The resultant split of data can be observed in the below graph.

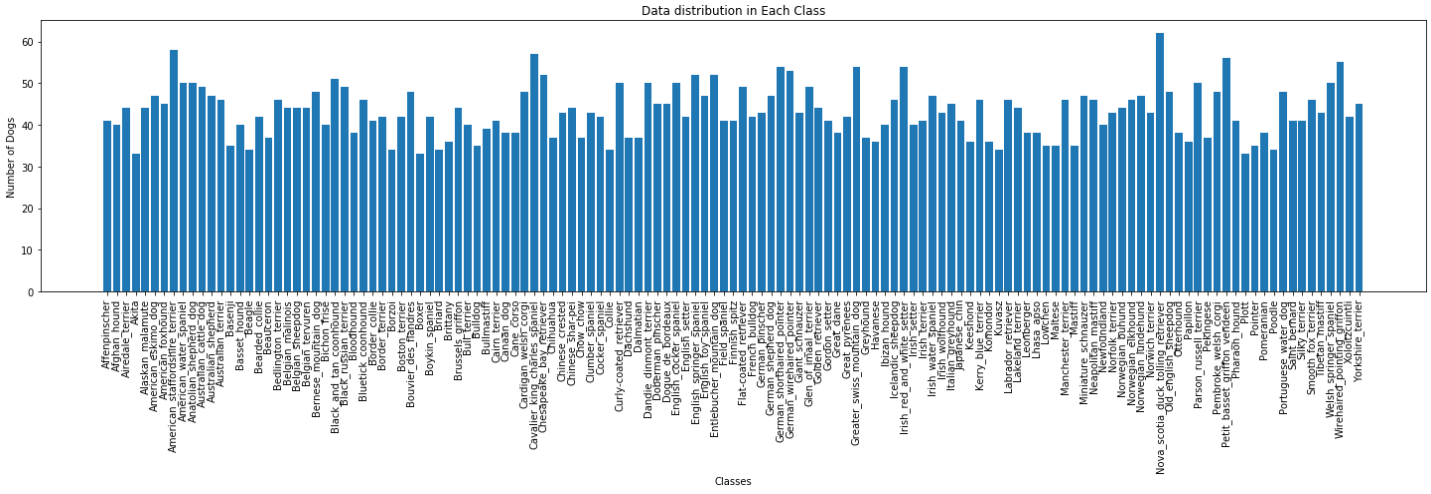
From the below plot we can observe that a total of 6680 images will be used to train our machine, to further fine-tune the parameters we use another 835 images for validating it. And, lastly, we will be using 836 images to test our model's performance for the evaluation of metric i.e. Accuracy.



>

Exploratory Visualization

The study on the distribution of data gives us the information on balance or imbalance in the data. If there is an imbalance in the data beyond a certain threshold, we must see that the data is balanced by adding relevant images. If the balance in the data is comparatively near to the threshold, it is good to carry forward with the operation. Let us see how it works with our data in the below figure. The plot shows a clear description on breed class with the number of dogs.



On observing the distribution of data within each class, the number of images is beyond a certain threshold value (i.e. nearly 40 per class). Though the data is not distributed evenly along the graph, the number of images is sufficient to predict the class of a particular breed and looks balanced.

Provided the data is small, not sufficient and the model is getting to overfit. In such a case, we need to augment the data to increase the number of image samples for the model. Therefore, augmentation makes an impact of bringing down the overfit condition and gives transparency to the model to make a predictive decision on test samples.

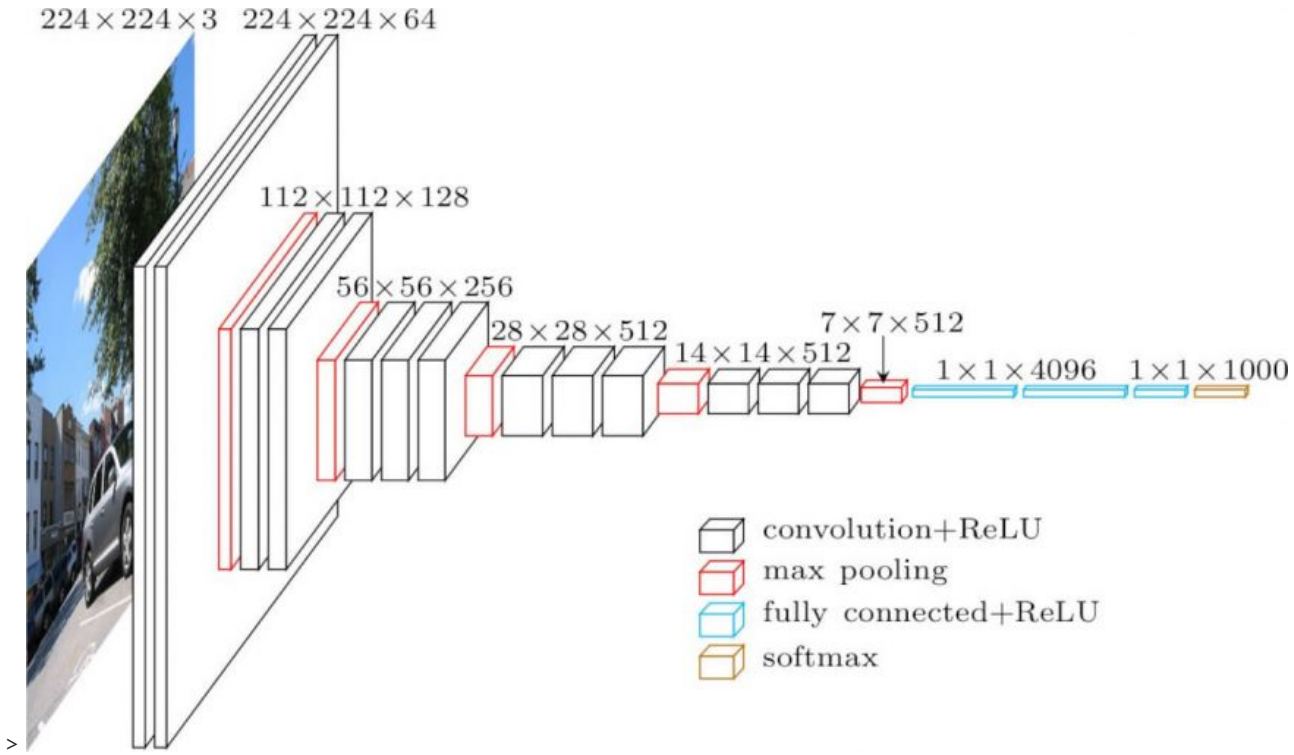
Algorithms and Techniques

From exploring the data, we visualize the actual distribution of data in each class. In order to classify the images based on their breeds, we need a classification algorithm to do the work. In this case, we use the pre-trained model such as VGG16 and a custom model for classification purpose. The intention of using the VGG16 is the need of performing transfer learning to our self built custom algorithm.

VGG-16 Algorithm Model

VGG 16 is a type of Convolutional Neural Network trained on a dataset of over 14 million images belonging to 1000 classes. The algorithm was proposed by K. Simonyan and A. Zisserman in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The network consists of:

- A stack of convolutional layers
- Three fully connected layers The first two FCN consists of 4096 channels followed by 1000 channels for the last FCN as each belong to one class.
- The last layer is a softmax layer which gives a probabilistic distribution of the respective class.



>

Looking onto the diagram above, we see the input size required for the model is 224x224. Therefore our model should be resized to 224 to pass in the image as input to the model of VGG-16.

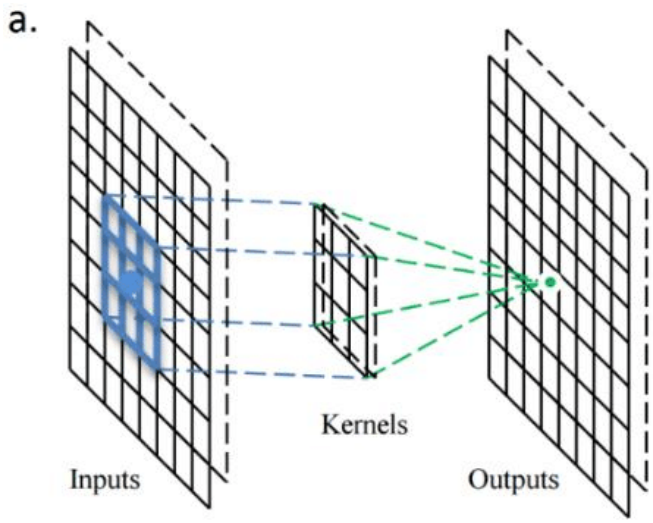
Some of the drawbacks of the VG166 are:

- It is slow to train
- The network architecture weights themselves are quite large

Convolutional Neural Networks

Convolutional Neural Network (CNN) is an algorithm from deep learning which is most commonly applied to visual imagery and are used for image processing, classification, detecion and image segmentation purposes. The convolutional neural netowrk takes in the image as input and gives the respective class of the image or probabilistic percentage of the class of the given input. The major advantage of CNN is requires very less pre-processing techniques. Also the feature engineering is done by passing a filter and carrying convolutional operations to obtain required features to train the model.

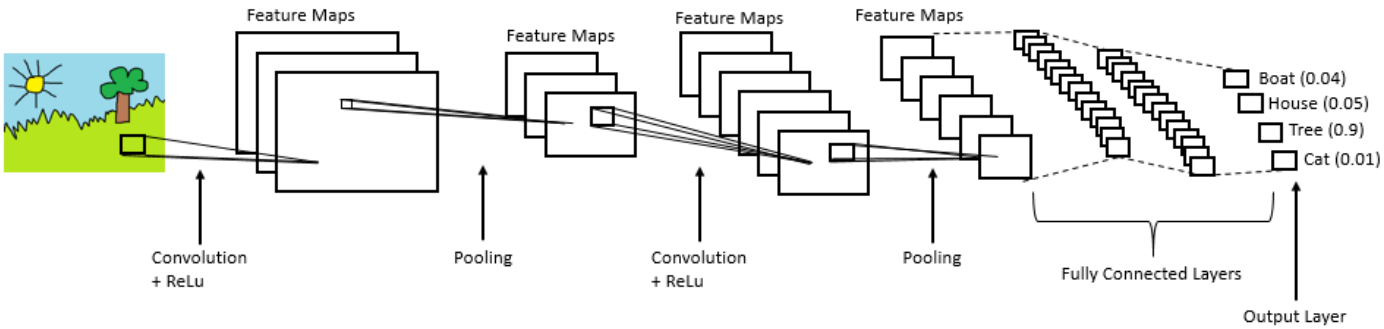
Each convolutional layer contains a series of filters known as convolutional kernels. The filter is a matrix of integers that are used on a subset of the input pixel values, the same size as the kernel. Each pixel is multiplied by the corresponding value in the kernel, then the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output channel/feature map.



>

There are various layers in the convolutional networks:

- Input layer: will hold the raw pixel values of the image,along with three color channel i.e. RGB
- Convolutional layer: This is the layer from which the features are extracted. It will compute the output of neurons that are connected to local regions to the input, carrying computation for each a dot product between their weights and a small region they are connected to in the input volume.
- Pooling layer: The layer that is used to downsample or reduce the dimensionality of the the number of images when the size of the data is relatively large. The pooling layers are of different types: i.e. Max pooling, Min pooling and Sum pooling.
- Fully connected layer: In this layer, the images are flattened and converted to column vector. The column vector is fed to a feed-forward neural network and backpropagation applied to every iteration of training helps the model to learn. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them.



>

In order ot get the best resultant outcome from the CNN model, we need to tune the neccessary hyperparameters which play a vital role to determine the models performance. Some of the hyperparameters based on the spatial and traing hyperparametes are:

- Number of convolutional layers: here we need to determin the reuired number of convolutional layers to perform convolutional operations on the images. So that the good amount of features are exracted from the images.
- Learning rate: a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient.Tuning the learning rate enables us to achieve the best global optimum result from the model
- Epochs: epochs is the number of complete passes through the training dataset. The parameter enables us to determine the right amount of training required based on the error rates of the model.
- Dropout: is the regularization technique to avoid overfitting of the model. The dropout is usually maintained between 20% - 50%. A probability too low has minimal effect and a value too high results in under-learning by the network.

Custom Built Algorithm Model

Based on the design of VGG16, we build our model on similar terms. In our model we have:

- Five convolutional layers
- Followed by the three fully connected layers
- A pooling layer
- Lastly the dropout layers

In our algorithm, we use the relu as the activation function and a pooling layer to reduce the dimension of the image data. In the first FCN, there are 25088 channels which are converging to 133 in the third FCN. Where 133 belongs to the number of classes from the last fully connected network. We also have dropout layers in our model to make sure the model does not get into overfit conditions.

In order to evaluate the performance of the above model, we verify it trough transfer learning and compare it with the other benchmark works carried before. The benchmark works carried out in this field is discussed in detail at the benchmark section below.

Benchmark

To tackle such data, we use a benchmark model to build a basic pipeline and a well-versioned model to improvise our classification rate. Such a methodology is carried to tune our model for better prediction of results. The benchmark model helps us to make a comparison and reduce the overfitting or underfitting condition and tune the model for a better outcome. Logistic Regression, KNN are such examples of the benchmark. We can also use the predefined image classifiers such as ImageNet, ResNet, VGG16, etc. to classify our images and later optimize our pipeline for better evaluation of metrics.

In the works of Dog Identification in Kaggle, we see that "Mohamed Sheik Ibrahim" used the VGG19, a predefined base model and carried various processing techniques such as data augmentation to improvise the results obtained from the predefined model. He also used logistic regression to classify the images of dogs and achieved an accuracy of 68%.

Considering another work performed on the same data, using the inception v3, the pre-trained model for image classification, Carey B achieved an accuracy of 83%, which is considered to be a good classification rate based on the performance of the model.

Using the above understandings, We will be using VGG16 for our data for classifying the breeds of the Dogs, Later we build a Convolutional Neural Network and tune the parameters, Using transfer learning through these models, make a comparative study and analyze the performance of the model.

III. Methodology

Data Preprocessing

In the pre-processing part we will be doing following:

- As the size/resolution of the images remain different we need to resize the images to the same scale. Therefore, studying the input requirements of the VGG-16, we know that it requires an image of the size 224. As the reason we are resizing the images of the data to 255 along with a center crop of 224 as needed fr VGG16.
- Also, as mentioned above, we will augment the data to increase the robustness of the model. In data augmentation, we will be increasing the robustness of the model through transformations of the image by rearranging information at different orientations. The transformations such as rotation, translation, etc are performed so that the the model is robust to classifiy the images having informations at different orientations.

The training data is augmented for the custom model using the augmentation techniques such as rotation and flip. The rotation of image is made to certain angle of 10 degrees and images are flipped horizontally to increase the qulaity of robustness of the model to classify the images containing infomration at different orientations.

Implementation

The metric of evaluation used to test the model's performance is accuracy. The determination of accuracy plays a huge role as we need to satisfy the pre-set conditions in prior. The conditions are as follows:

- The custom model should have a test accuracy greater than 10%
- The model after the transfer learning should give a test accuracy greater than 60%

As we get the data ready from the previous stage based on the preprocessing methods, we need to build the algorithm for the implementation phase for the purpose of classification. The input image is resized to the needed dimension of VGG16. Also, the image is center cropped to the required dimension in order to keep the relevant information and discard the irrelevant ones.

In our case, we are also normalizing the images to a certain range. The normalization is carried to reduce the skewness or varied range of dimensions of images to the fixed range in the model and also helping it to learn faster. Since we are using the color images which are having three channels, therefore we need a tuple to each channel to carry the normalization. We use the relevant normalization parameters as shown below.

```
transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

The formula for normalization for each channel is given by:

image = (image - mean) / std

Implementation of pre-defined model VGG-16:

Using the pre-defined model of VGG-16 as explained in the algorithm section, we used the preprocessed data as the input to the model to make the prediction. Once the prediction has been done, we see the resultant outcome from the model within a range from (151-268) belonging to a certain class.

The model is evlauated by passing a data of dog and it is seen that the model predicts the output in the given category i.e. 243, which lies in between 151-268. The respective predict function performed by the VGG16 is gven below:

```
VGG16_predict(dog_train[0])
```

We see the implementation carried for a single data of dog. However we need to verify if the predicted output is correct. Therefore hte dog detector function created uses the VGG16 model to predict and test if the predicted answer is right or wrong. The function is implemented on 100 images of dog and human data. Based on the results we see that:

- Total number of dog faces detected in human data: 1
- Total number of dog faces detected in dog data: 100

The result shows that the function is carried in a proper sense to evaluate the predict function of VGG 16 as required for our purpose.

Therefore, now we are able to predict the dog and human from the images. However, we need to predict the breed of the data given, as the reason we need to build the custom model to predict the breed of the data given.

Keeping the pre-trained model ready, we now bulid our custom model as explained in the algorithm section.

As the data is split for train, validation and test. We need to load the respective data accordingly. Based on the requirement as mentioned above, we will augmenting the data in this case along with other preprocessing techniques.

The training data is augmented for the custom model using the augmentation techniques such as rotation and flip. The rotation of image is made to certain angle of 10 degrees and images are flipped horizontally to increase the qulaity of robustness of the model to classify the images containing infomration at different orientations.

Implementation of Custom Model from Scratch:

The architecture for the custom model is built and implemented as under:

- A total of 5 convolutional layers are used with a kernel size of 3 and padding equal to 1. The convolutional layers are necessary to extract the low-level features from the images.
- The Max Pooling layers are used to downsize the spatial dimension, reduce the computations and remove the unnecessary information from the images.
- 3 fully connected layers are used as a decision making layers for classification purposes. They take in a total of 25088 features and downsize it to 133 classes of dog breeds
- a single layer of dropout is used to make sure the model is not getting overfitted on training it.
- Setting the loss function to cross-entropy and optimizer to Stochastic Gradient Descent

Therefore for every convolutional, we have the pooling performed to downsize and also dropout for every fully connected layer to prevent the model from overfitting condition. From the list Fully connected network we have an output of 133 which is the class breeds of the dog.

The train function being designed plays a vital role to train the custom model for classification purposes. Within the train function we carry the following operations:

- make use of the GPU to carry the work at a much faster pace.
- evaluate the training loss
- evaluate the validation loss

Setting the parameters of the train function we will train the custom model using the training data of dog images. The epoch is set to 25 and the training is carried out on the custom model. We see there is a continuous drop in the loss till epoch 20. At the epoch 20 we see that the training loss is about 3.978901 and validation loss is about 3.993918. The training loss is less than the validation loss and leads to overfitting condition of the model.

However, on testing the model, we need to satisfy the pre-set condition of accuracy being more than 10%. In our case, on testing the data from the trained custom model, we achieve an accuracy of 12% of which 102 out of 836 are predicted right. Therefore, we satisfy the required condition of the needed test accuracy.

Implementation of Model for Transfer Learning

Initially, we are calling the pre-trained model of VGG-16 with the pre-trained weights. As a reason, the pre-trained parameters is set to True. We are also freezing the parameters i.e. weights of the VGG-16 model in lower convolutional layers and adding the custom model with a layer of trainable parameters as required. We set the last classifier layer with 4096 features and 133 as the number of classes as required as our output.

Also specifying the loss function to Cross-Entropy and optimizer to Stochastic Gradient Descent, we are training the machine by making use of the cuda for training purposes. The parameter - epoch is set to 8 and the model is trained and we also observe there is a considerable decrease in the loss. Using the best validation accuracy of the model by saving it on training, we are testing our model's performance on the test data.

The result of the transfer learning model when evaluated on the testing data gives us an accuracy of about 85%. This satisfies the required condition set in prior.

Implementation of application for the prediction of dog breed using the model

To implement the application for dog breed prediction we are constructing a function to predict the dog breed and another function to run the app.

The implementation of the function for predicting the dog breed contains:

- loading the image path
- pre-processing the images which are required to be passed as an input
- using the model to evaluate the image and make a prediction
- return the result of predicted breed

The implementation of the function for running the application contains:

- if the dog detected, using the prediction function designed, predict the breed of dog.
- if the dog not detected and human detected, predict the dog breed for the human suing the previously designed function.
- if neither dog nor human detected, throw a message displaying the error in the image

Lastly, the implemented algorithm is tested and evaluated for various images of humans and dogs. The resultant outcome is shown in the code output.

Refinement

Augmentation of the data: The images for the training data are augmented in order to increase the number of training images. This helps the model to distinguish data irrespective of the transformations done to the images.

Tuning of the hyperparameters: Epoch size for custom model Initially, the model was trained for 100 epochs and it was observed that the model was overly trained. Later, the model was trained for 15 epochs but the test accuracy of the model dropped below 10% and hence the epochs were reduced based on the tuning and set to 25 to which the condition was satisfied.

Tuning of the hyperparameters: Epoch size for transfer learning model Following the same procedures above, the epoch was set to 5 initially but the resultant outcome of the model was not satisfactory and later on tuning to various sizes, it was set to 8 and the resultant best accuracy was achieved.

The other parameters that were tuned for the better prediction of results are the dropout value and the learning rate. On tuning this, we observe that the model was much stable in prediction.

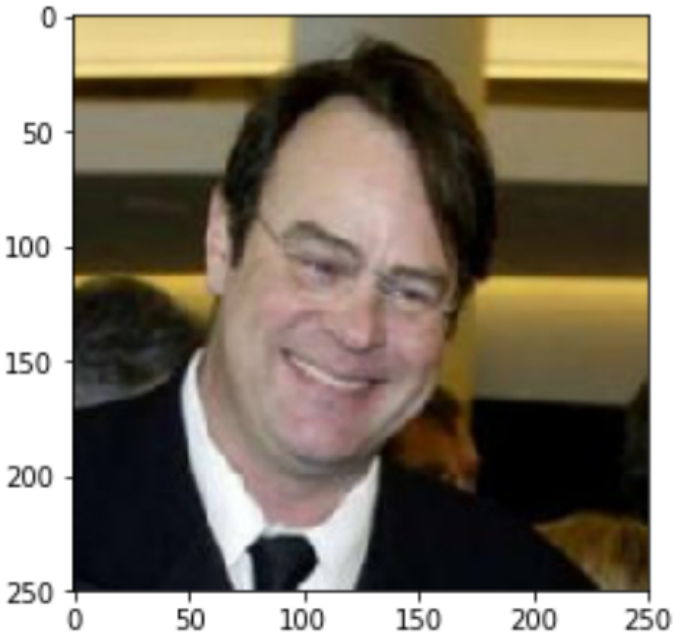
IV. Results

In this section, we observe the performance evaluation and results of the final implemented model. Initially, we observe the study made on the detection algorithm and compared their results to see their behavior. On implementing the haar cascade and local binary pattern, we observe there were 98 faces detected for haar cascade and 95 faces in local binary pattern of human images. The results show the haar cascade has a better performance compared to the local binary pattern.

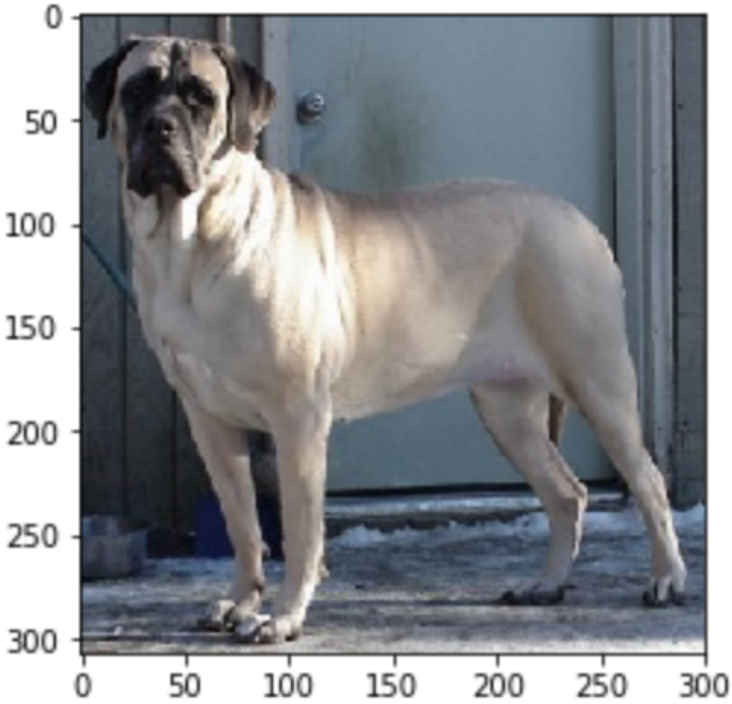
In the implementation phase of the pre-trained model VGG 16, we test the prediction of the model and we also observe that the results show the right prediction value of the dog. On implementing the custom Convolutional Neural Network model, we construct out the relative stack of layers required for our purpose and tune the necessary hyperparameters of the model.

On testing the performance of the custom model through the metric evaluation i.e. accuracy in our case, we observe that the results give us a result of 12% in terms of accuracy. The project had the condition to achieve the results of more than 10% accuracy. However, our model outperformed the required condition set and achieved an accuracy of 12% for the custom CNN model.

After the implementation and training of the transfer learning model for a total of 8 epochs, we need to evaluate the performance of the given test data of dog images so that the model should predict the breed of the dog with utmost accuracy. Therefore, getting the model ready for evaluating the performance on the test data, we used the test function that was previously defined for the custom model and evaluated the performance of the transfer learning model. We see that the model achieved an accuracy of 85% outperforming the pre-set condition of about 60%, we see that there is a considerable rise in the performance of the model after combining the predefined model with our custom model.



```
Hello, human!
You look like a...
Brittany
```



```
Hello Doggie!
You belong to a breed named: Bullmastiff
```

Justification

Initially, we had discussed the works carried in this field and also observed the results achieved from these benchmark models. While one of the works has achieved an accuracy of 68% and other of about ana accuracy of 83%, our model achieved a result of 85% and outperformed the results achieved the mentioned benchmark models.

The results achieved are based on the analysis of our model's performance for the given data of the breed of dogs. It has justified the pre-defined conditions set forth in this project and also outperforms the benchmark model in terms of performance.

V. Conclusion

In this project, we see the implementation of Dog Breed Classification. This gives us a detailed description of how the classification of dogs based on their breeds are carried out and implemented. From the data, we study and make an analysis of the split of data, type of data, distribution of data and also the visualization on how the data is distributed. From the split, we see that 80% of the data is used for training, 10% for validation and the remaining 10% for the test data. From the visualization of data distribution, we see that though the data is not completely in balance, but the data in all the classes are above a certain threshold and distributed evenly. We also make a comparative study of the detecting algorithms using haar cascade and local binary pattern.

In the preprocessing phase, we see how the data has been preprocessed using resize, center crop and normalization. Selectively for training, we also observe the data augmentation been done using the transformation techniques to increase the dataset size for training purposes.

In the implementation phase, we implement the pre-trained model, custom model and transfer learning model. In the pre-trained model, we use the model to directly make a prediction on the given data. Building our own custom model, we see that the model is implemented using the convolutional architecture as shown in the implementation phase above. We also observe that the custom model gives us a test accuracy of 12% satisfying the required condition and also improvising on fine-tuning the hyperparameters such as epochs, dropout value, and learning rate.

In the transfer learning, we see the involvement of a custom model with the pre-defined model and training it for the whole data and see a significant rise in the accuracy level to an output of 85% in test accuracy. On making the comparative study with the benchmark model, we see our model has outperformed the benchmark and also satisfying the condition of above 60% accuracy. However, by using the improvisation techniques, the model can be further improved in terms of evaluation and performance.

Improvement

Further, the project work can be improvised by using the following improvisations.

- Firstly, an advanced detection algorithm can be used which is having better precision such as the MTCNN face detection algorithm.
- Secondly, The task becomes much tedious and difficult to classify the images that contain both dogs and humans, therefore the algorithm can be improvised on such a dataset.
- Thirdly, a real-time application can be achieved on the production scale where when the user passes an input image, we can predict the put of the image in real-time using the API.
- Fourthly, as the classification rate depends on the pre-trained models-preferring a better pre-trained model, will have a good chance of improvising our prediction rate of the model.
- Lastly, tuning architectural factors can also be considered for the improvisation of the model's performance.

References

https://towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch-dd09190245ce

https://discuss.pytorch.org/t/transfer-learning-using-vgg16/20653

https://discuss.pytorch.org/t/questions-about-imagefolder/774/5

https://towardsdatascience.com/a-beginners-tutorial-on-building-an-ai-image-classifier-using-pytorch-6f85cb69cba7

https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148