MACHINE LEARNING ENGINEER NANODEGREE

CAPSTONE PROJECT

DOG BREED CLASSIFIER

Kandukuri Ratna Prakarsha

September 26th, 2021

1. Definition

1.1 Project Overview

Human Beings have conquered the world with their cognitive capabilities, yet there are some tasks that would need external assistance. This Capstone Project aims to deal with one such case. Image classification is a modern problem, and a lot of development has been taking place in that area. With drastic advancements in the face of hardware technology, we are now able to concentrate more on problems at hand rather than the computational difficulties. For example, automation of major medical tasks like detecting tumors, cancers, cysts now allow the doctors to utilize their time effectively and pay attention to much more important tasks.

Consider dog breed classification, hundreds of varieties of breeds could be difficult for a human to memorize. Keeping the memory limitations of a human aside, it would take only a true canine expert to differentiate among all the breeds available, as there are breeds with extremely minimal differences. To deal with these problems, humans can use Machine Learning Models that help recognize the dog's breed through an image.

The chosen project is a classification problem. In this project we epitomize the use of CNNs (Convolutional Neural Networks) and demonstrate a dog breed classifier. A dataset provided by UDACITY is utilized for training, testing and validating the ML models. The dataset contains a huge number of dog images of various breeds and human images of varying persons. Two different models are built where one is a CNN built from scratch and the other is a CNN that uses transfer learning from a pre-trained model.

1.2 Problem Statement

The chosen project comes under the domain of image classification and uses CNNs to solve it. The goal of this project is to develop an

algorithm that could classify a dog breed and be used as part of a mobile or web app.

The algorithm will accept any user-supplied image as input and give output in the following manner:

- If a dog is detected in the image, it would provide the detected dog's breed.
- If a human is detected, it would provide an estimate of the dog breed that is most resembling.
- If neither is detected, it would return an error.

Firstly, to detect a human face, Haar feature-based cascade classifiers of OpenCV was used. Further, to detect a dog, a pre-trained VGG16 model that was trained using ImageNet data consisting numerous labeled dog images was implemented. Finally, to classify the breed, transfer learning from a pre-trained CNN model was used as it would already be capable of detecting the features of dogs to an extent. To be more specific VGG16BN model was used. VGG16BN model is VGG16 with slight variations in its architecture.

1.3 Metrics

The evaluation metrics used in this project is accuracy. Accuracy is simply the ratio of correct predictions to the total number of predictions made.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + F} \times 100$$

TP -> True Positive

TN -> True Negative

FP -> False Positive

FN -> False Negative

This metric is apt for evaluating the model because it gives a perfect measure of how well the model is performing by taking into account

the number of correct and wrong predictions. Also, as it is in percentage, it is readily understandable.

2. Analysis

2.1 Data Exploration and Visualization

The data source for this project is UDACITY itself as it provides all the data required for training, testing, and validating the ML model. There are 2 categories of datasets used in the project, one is a dataset of human images and the other is of dogs.

The human dataset contains 13233 images of real people overall. Each person's images are present in varying proportions. In Fig1 We can observe that although the overall data seems to be evenly distributed, there is one person with more that 500 pictures, while most of them have the count below 200. As the data is only being used for testing, this will not have any impact on the model performance.

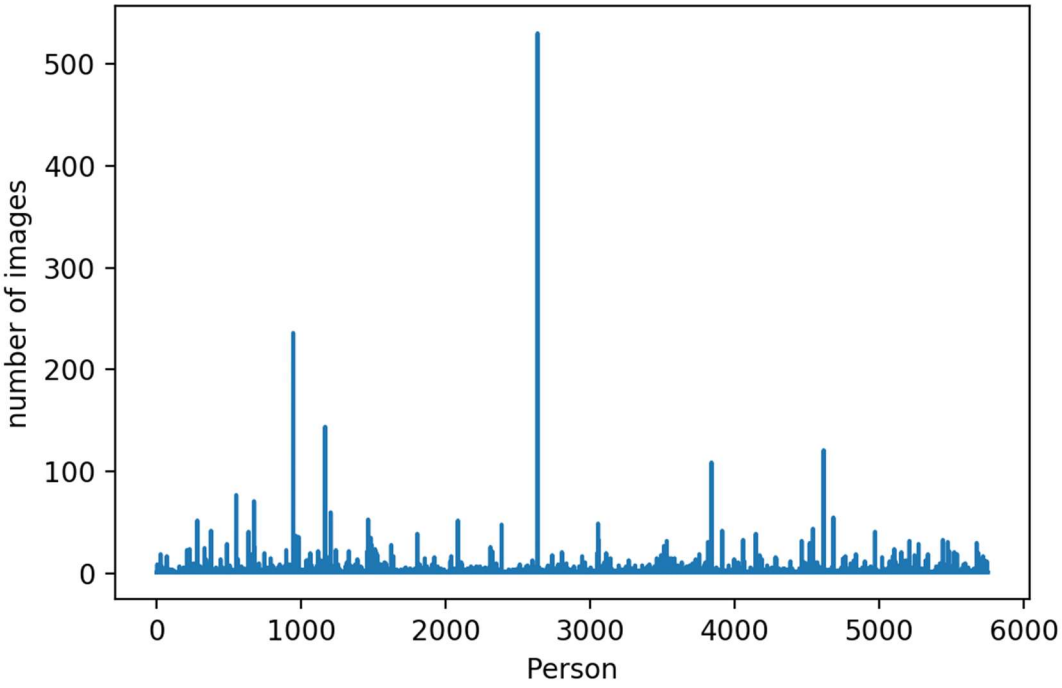


Fig1. Plot showing the count of images for each person in the human dataset

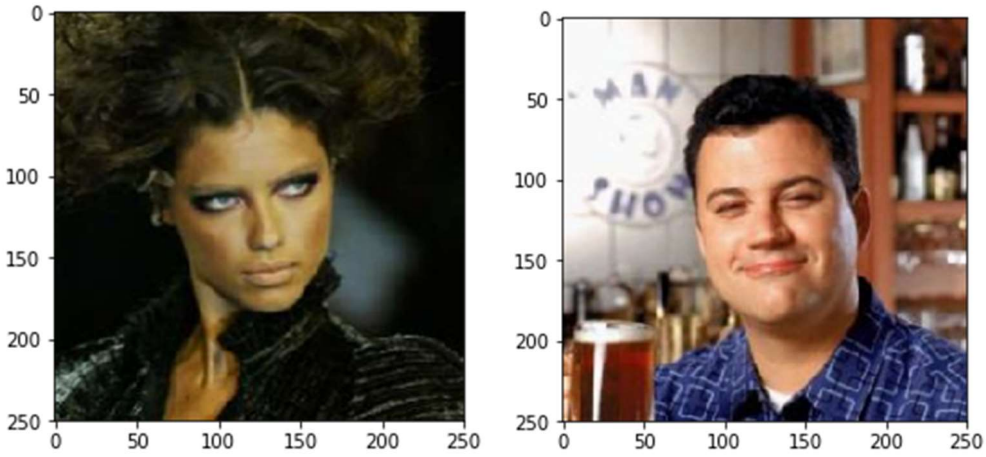


Fig2. Sample images from human dataset

The dog dataset has a total of 8351 images of various breeds. There are 133 breeds overall, out of which each breed's pictures are in varying amounts. The training dataset comprises 80% of the dataset and has 6689 images. The test and validation datasets sum up the rest of the 20% with 835 (10%) and 836 (10%) images respectively (Fig3).

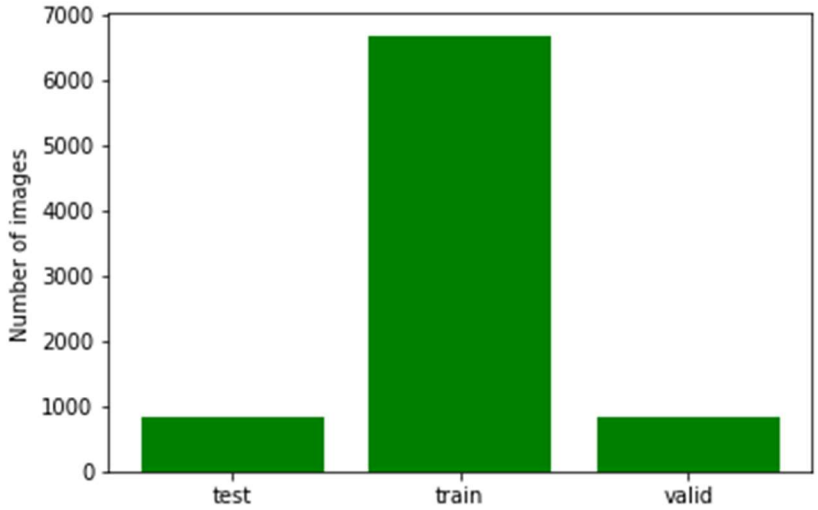


Fig3. Plot showing the count of images for train, test and validation datasets of dog images

It can be understood from Fig4, Fig5 and Fig6 that all three datasets have different counts of each breed. Although the distribution of test and validation does not matter, the uneven data distribution of train dataset might impact the performance of the model.

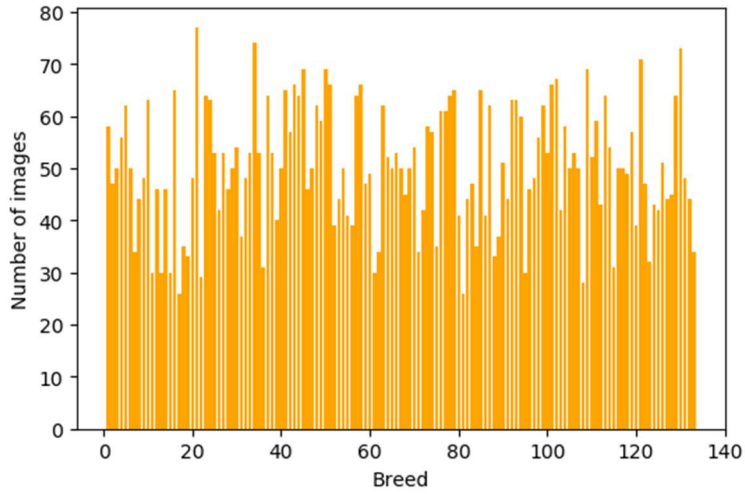


Fig4. Plot showing the distribution of each breed in train dataset

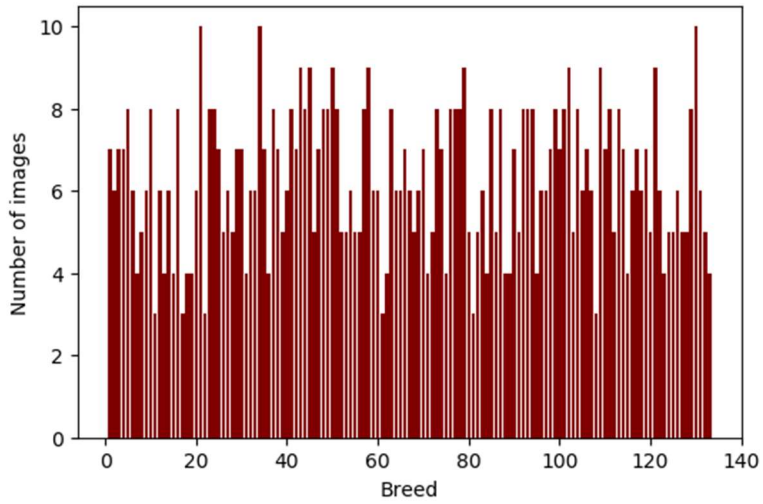


Fig5. Plot showing the distribution of each breed in test dataset

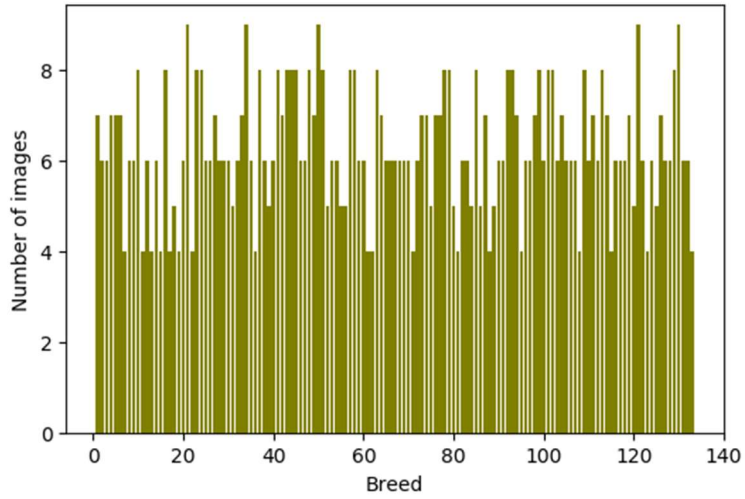


Fig6. Plot showing the distribution of each breed in validation dataset

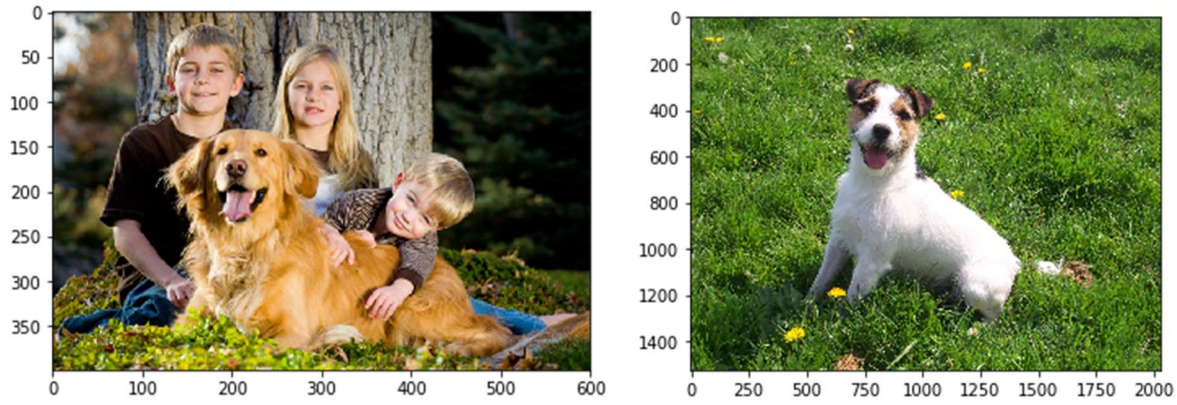


Fig3. Sample images from dog dataset

2.2 Algorithms and Techniques

The algorithm and techniques used to achieve the aim of this project are discussed below.

1. Human Face Detection:

   To detect a human face, Haar feature-based cascade classifiers of OpenCV is used. The pre-trained face detector (haarcascade_frontalface_alt.xml) is given as input to Haar Classifier as an xml file. The pre-trained models are stored as xml files on GitHub. The image is made to go through the required pre-processing and then fed into the model to make an inference and get the result. (Pre-processing will be discussed in Methodology later.)

2. Dog Detection:

   To detect a dog, a pre-trained model, VGG16 is downloaded along with weights. This model is trained on a very popular dataset called ImageNet and hence, is expected to give a high accuracy. As the model weights are also downloaded, there is no need of training. The image is pre-processed as necessary and fed into the model. (Pre-processing will be discussed in Methodology later.) The model then returns a numeric value ranging between 0 and 999 which represents the ImageNet class of the predicted output. The categories corresponding to dogs appear in an uninterrupted sequence and correspond to dictionary keys 151-268, inclusive, to

include all categories from 'Chihuahua' to 'Mexican hairless'. Hence, if the obtained output ranges between 151-268 then that means a dog is detected.

3. Dog Breed Classification:

   To classify the dog breed, a pre-trained VGG16BN model is trained on the provided dataset using transfer learning. VGG16BN is same as VGG16 with slight architectural differences. The BN in VGG16BN stands for 'Batch Normalization' as it undergoes batch normalization for dense layers. After training the model, the model is then given a suitably pre-processed image for which the model returns the breed of the dog in the given image. (Pre-processing will be discussed in Methodology later.)

   Parameters required to tune while training are:

   1. Epochs: An epoch refers to one cycle through the full training dataset.
   2. Batch Size: Batch size refers to the number of training examples utilized in one iteration.
   3. Learning Rate: The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated.
   4. Dropout rate: Dropout is a technique used to prevent a model from overfitting. Dropout works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase.
   5. Activation Function: An activation function is a function used in artificial neural networks which outputs a small value for small inputs, and a larger value if its inputs exceed a threshold. If the inputs are large enough, the activation function "fires", otherwise it does nothing
   6. Optimizer: Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate in order to reduce the losses.

2.3 Benchmark

Benchmark model chosen satisfies the following criteria:

- The CNN trained from scratch needs to classify the images in the test dataset with at least 10% accuracy. The reason for having only 10% as accuracy is because training a CNN from scratch is a very challenging task as there are dog breeds with very little differences.

- The CNN model trained using transfer learning must achieve a minimum of 60% prediction accuracy on the test dataset. It's justified to have such an accuracy as benchmark because with transfer learning the model would already be able to identify and understand features in the images to an extent.

3.Methodology

3.1 Data Pre-processing

The data pre-processing required in the project are discussed below:

1. Human Face Detection:
   - Training: For face detection a pre-trained model is being used, hence there is no training process involved here. Therefore, no data pre-processing is required.
   - Inference: Here there is only one step involved, the image needs to be converted into grayscale before being fed to model.
2. Dog Detection:
   - Training: For dog detection a pre-trained model is being used, hence there is no training process involved here as well. Therefore, no necessity of data pre-processing.
   - Inference: Here the image needs to be pre-processed before being fed to the model.
     - ✓ The image is first resized to 256 x 256 dimensions and then centre cropped to 224 x 224. This is because VGG16 needs the dimensions of the images to be so.
     - ✓ The 3 channels of an image, red, green blue are normalized with a mean of 0.485, 0.456 and 0.406

respectively, and a standard deviation of 0.229, 0.224 and 0.225. Data normalization is an important step which ensures that each input parameter (pixel, in this case) has a similar data distribution. This makes convergence faster while training the network.

3. Dog Breed Classification:

   Both for training and inference, the following pre-processing is applied to the images:

   - ✓ The image is first resized to 256 x 256 dimensions and then centre cropped to 224 x 224. This is because VGG16BN needs the dimensions of the images to be so.
   - ✓ The 3 channels of an image, red, green blue are normalized with a mean of 0.485, 0.456 and 0.406 respectively, and a standard deviation of 0.229, 0.224 and 0.225. Data normalization is an important step which ensures that each input parameter (pixel, in this case) has a similar data distribution. This makes convergence faster while training the network.
   - ✓ Random flips and rotations are also added so that the model understands pictures taken in a variety of angles. This step is required only for training images.

3.2 Implementation

This section discusses the flow of implementation of the whole project.

1. Human Face Detector

   As mentioned in ‘Algorithms and Techniques’, to detect a human face, Haar feature-based cascade classifiers of OpenCV is used. The pre-trained face detector (haarcascade_frontalface_alt.xml) is given as input to Haar Classifier as an xml file. The image to be fed into model is converted to grayscale using ‘cv2.COLOR_BGR2GRAY’. ‘detectMultiScale(image)’ is used to count the number of faces detected in the picture.

   A function is implemented which takes the image path as an input parameter, converts the image into grayscale, counts the number of

faces, if the number of faces are more than 0 then it returns true, else it returns false.

2. Dog Detector

   Using PyTorch, pre-trained VGG16 model is downloaded and implemented. A function is built which takes an image path as input parameter, performs the transformations mentioned in 'data pre-processing' on the image, and feeds it into the model for inference. If the model returns a number between 151 and 268 (inclusive) then it returns true, else it returns false.

3. Dog Breed Classifier

   Using PyTorch, pre-trained VGG16BN model is first downloaded and trained further using the data provided. The model is pre-trained hence, we need not configure the number of CNN layers, 2 fully connected layers are added where the first one takes 25088 input features, and has 4096 nodes in the hidden layer, the second layer gives out 133 output classes. 133 output classes because the dataset has 133 different dog breeds. The model is then trained for 25 epochs using adam optimizer with a learning rate of 0.01, cross entropy loss, ReLU activation funtion and a dropout of 0.5 are also added.

   A function is implemented which takes in the path of an image as input parameter, transforms the image, feeds it into the model and returns the dog breed.

4. Final App:

   This part is a mixture of all the above implementations. Here a function is built where it accepts the path of an image as input parameter, checks if the given image has a human face, if yes, then it returns the dog breed the face matches to, if not then it checks if the image consists a dog, if yes, it returns the dog breed else it returns an error.

3.3 Refinement

It took experimenting with various models to arrive at the final model.

- Initially, VGG16 model was used with the same parameters mentioned in the 'Implementation part'. It gave only 66% accuracy. Although it satisfies the benchmark criterion efforts were made it improve it further.
- VGG19, VGG19BN were also implemented keeping the parameters constant but they did not even pass the benchmark.
- VGG16BN was trained with the parameters documented in 'Implementations' section and it achieved an accuracy of 76%.
- This way we arrive at our final model with the following specifications:
    - ✓ Model: VGG16BN
    - ✓ Epochs: 25
    - ✓ Learning Rate: 0.001
    - ✓ Optimizer: Adam
    - ✓ Loss function: Cross Entropy
    - ✓ Dropout: 0.5
    - ✓ Activation function: ReLU

4. Results

4.1 Model Evaluation and Validation

After a lot of fine tuning the best performing hyperparameters were chosen for the final model. While training the model was constantly evaluated using validation data. Later the model was also tested on the test data where it achieved an accuracy of 76%. The model was then tested using some more unseen data, where the model performed reasonably well.

4.2 Justification

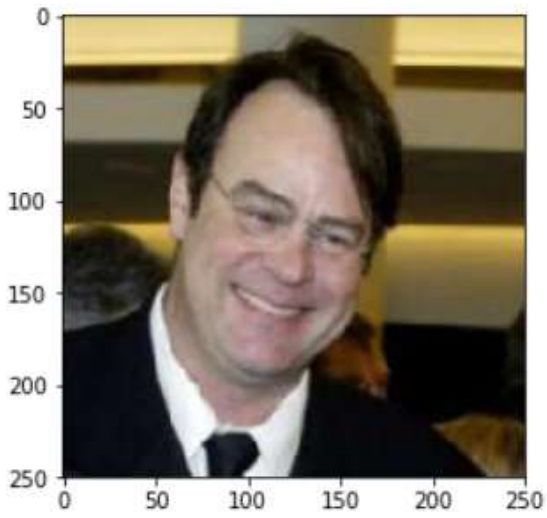
The project meets the benchmark criteria as the model built from scratch got an accuracy of 14 % and the model built using transfer learning achieved an accuracy of 76%.

5. Conclusion

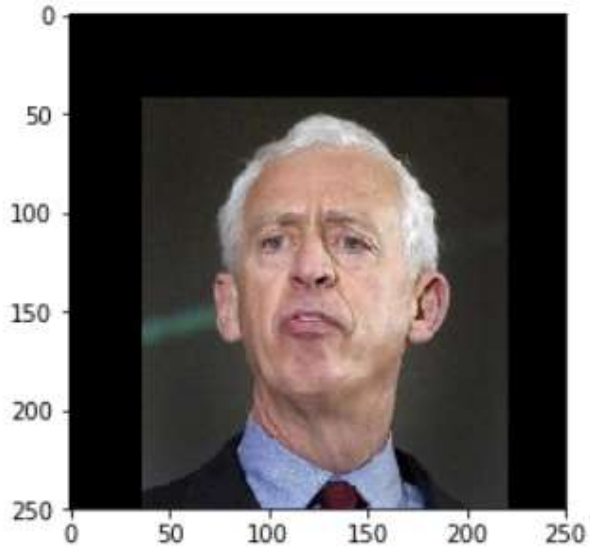
5.1 Free-Form Visualisation

Below are some images from the outputs of the project.

```
============================================================
You are a Human!
```




```
But are you sure that you are not a Cardigan welsh corgi?
============================================================
============================================================
You are a Human!
```




```
But are you sure that you are not a Bull terrier?
============================================================
```

==========
Hey doggie!

Did you know that you are a Mastiff?
==========

==========
Hey doggie!

Did you know that you are a Beauceron?
==========

```
==========================================================
Error: Oops! Who are you?
```


```
==========================================================
```

5.2 Reflection

This project is a result of extensive research and development. The most difficult part of this project is the hyperparameter tuning for dog breed classification. This section contains a brief overview of the complete project.

- The data required for the project is imported and analysed.
- OpenCV's Haar feature-based cascade classifier is implemented to detect humans in a given image.
- Using pre-trained VGG16 model, a dog detector is created.
- A CNN to classify dog breeds is built from scratch. The model has an accuracy of 14%.
- A CNN to classify dog breeds is built using transfer learning using a VGG16BN pre-trained model. The model achieved an accuracy of at least 76%.

- An algorithm is built that identifies if an image has a dog or a human. Further, it returns the dog breed in case a dog is detected, else it returns the dog breed that the human most resembles. If neither a human nor a dog is detected, an error is returned.

5.3 Improvement

Following are some aspects in which the project can be improved:

1) Accuracy can be further improved.

2) In pictures having both dog and human, model should be able to detect them both.

3) A webapp could be created to make user experience smooth.

REFERENCES

1. https://deepai.org/machine-learning-glossary-and-terms/epoch
2. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/
3. https://deepai.org/machine-learning-glossary-and-terms/activation-function
4. https://radiopaedia.org/articles/batch-size-machine-learning
5. https://towardsdatascience.com/machine-learning-part-20-dropout-keras-layers-explained-8c9f6dc4c9ab
6. https://towardsdatascience.com/a-beginners-guide-to-convolutional-neural-networks-cnns-14649dbddce8
7. https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6
8. https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258
9. https://pytorch.org/docs/stable/index.html
10. https://pytorch.org/vision/stable/models.html
11. https://www.researchgate.net/figure/Pre-trained-VGG-and-VGG-BN-Architectures-and-DNNs-Top-1-Test-Accuracy-versus-average-log_fig3_330638379
12. https://machinelearningmastery.com/use-pre-trained-vgg-model-classify-objects-photographs/
13. https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html
14. https://www.analyticsvidhya.com/blog/2019/10/how-to-master-transfer-learning-using-pytorch/