Dog Breed Classifier using CNN

Domain Background:

There are a set of famous problem statements in the field of Artificial intelligence, one of which is the classification of a dog breed. This particular project functions as a dog breed classifier. There is an input image given to the model, if a dog's image is given it identifies what breed it belongs to and if a human's image is given it tries checking if the face resembles any dog breed and mention it. This is a problem which can be solved using multi-class classification through supervised machine learning. I was interested in this particular project as I felt that this would be a really fun project to work on. It also gave me the opportunity to build CNN from scratch and deploy cool machine learning models, and hence I chose this as my capstone project.

Problem Statement:

The problem statement is that the model which I built should be able to import the dataset and process the input images given by an user and be able to identify the breed of a dog if an image of a dog is given and if an image of a human is given it should see if the human's face resembles any one breed of a dog and should mention it.

Datasets and Inputs:

For the training of my model I used images as the type of input since the model should identify the breed of a dog. I used the dataset provided by Udacity. The dataset will contain images of dogs and human beings.

The dataset is classified into types:

DATASET-1: This dataset totally has 8351 dog images. They are divided into 3 directories. 'Train' is the 1st directory which has 6,680 Images, 'test' is the 2nd directory which has 836 Images and 'valid' is the 3rd directory which has 835 Images. There are 133 folders similar to dog breeds in each of this directory. A lot of dissimilarities are observed in the input images, either they may not have

the same image size or there may be a difference in the background or the surrounding environment. The dataset also is not balanced, the images provided for each breed may vary, one breed may have 1 image while another breed may have 10 images.

DATASET-2: This dataset totally has 13233 human images, it has 5750 folders in it. A lot of dissimilarities can be observed in the human images, different background or environment. However the size of the image is the same (250 x 250). Just like the 1st dataset the dataset here is not balanced as there maybe 1 image for a person while there might be 5 images for others.

Sample images

Evaluation Metrics:

As mentioned above the input data is split into 3 datasets- training, testing and validation datasets. I used the data from the training dataset to train the model. I used the data from the testing dataset for predicting the performance of the model on new data. I used 2 evaluation metrics “log loss” and “accuracy”. The log loss is used to measure the performance of a classification model where the prediction input is a probability value between 0 and 1. Accuracy is easily suited for a multiclass classification problem. Accuracy is the proportion of true results among the total number of cases examined. The main role of accuracy as a metric came when I evaluated the model on test data.

Accuracy formula: (TP+TN)/(TP+FP+FN+TN)

Accuracy=Number of items correctly classified/ All classified items

While training the model, I calculated the log loss by comparing the test data prediction with validation dataset to find which model’s performance was the best . The Log loss takes into the account of uncertainty of prediction based on how much it varies from actual label and this will help in evaluating the model.

Algorithms and techniques:

As mentioned above this project involves solving a multiclass classification problem through supervised machine learning. we can use Convolutional Neural Network to solve the problem. For doing this image classification I made use of Convolutional Neural Network (CNN) which is a Deep Learning algorithm that takes in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

One of the main advantages of using a CNN is that it has a good accuracy in image classification problems. The CNNs represent a huge breakthrough in image recognition. They’re most commonly used to analyse visual imagery and are frequently working behind the scenes in image classification. Image classification is the process of taking an image as an input and sending an output which specifies a class (for example a ‘cat’ or a ‘dog’) or an output which

is a probability which specifies that the input belongs to a particular class (for example ‘there’s a 90% probability that this input is a cat’)

Firstly the model processes the input images and tries to detect the human images by using algorithm like OpenCV’s implementation of Haar feature based cascade classifiers, this is an existing algorithm. The model also tries to detect the dog images by using a pretrained VGG16 model. Finally, when the image is identified as a dog image or a human image, the image is passed to the CNN model which I built. The CNN processes the input image and predicts the breed that matches the best out of the given 133 breeds.

Different people use different types of CNN. I have taken use of the ResNet (residual network) as they are easy to optimize and they can easily gain accuracy from greatly increased depth, producing results which are better than previous networks. There are other people who use AlexNet as an option.

Benchmark Model:

A benchmark model is basically a simple or historical model or a result to compare the defined solution to. For this I created a CNN model from scratch. The benchmark model had a minimum cut-off accuracy of at least 10%. This is to confirm that the model is working because a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%.

Data Preprocessing:

For data preprocessing I have applied resizing, horizontal flip, random rotation and normalization in the training dataset. To reduce the risk of overfitting image augmentation is done. In the validation and testing dataset I have tried to resize and Center-crop the images to a specific size and also applied normalization. All the images are resized to 224*224 and the normalization is applied to the images of all the three datasets(‘train’, ’test’, ’validation’). After doing all the above steps the images are passed to the model after converting them into tensor.

Implementation:

I had to build CNN twice, one was built from scratch while the other was built using using transfer learning. The CNN model from scratch was built for the benchmark model. I have made use of three convolutional layers each with kernel size of 3 and stride 1. One is the pool layer to reduce the amount of parameters and computation in the network (here it reduces the input size by 2). I have also made use of the linear layer which is basically a single layer. I have used the ReLU layer which changes all the negative activations to 0. I have also used dropout to avoid overfitting. The model has two fully connected layers which produces a 133-dimensional output. I trained for 15 epochs. The epoch was used to indicate the number of passes of the entire training dataset which has been completed

The CNN model built from scratch gave the following results:

Test Loss: 3.979316

Test Accuracy: 12% (104/836)

Refinement:

The CNN model built from scratch meets the benchmark model's minimum cut-off accuracy (10%).I later built a CNN model using transfer learning. I used the Resnet101 architecture which is pre-trained on ImageNet dataset. I chose it over alexnet as it provides good accuracy and the architecture has 101 layers. within just 10 epochs.

The CNN model built using transfer learning gave the following results:

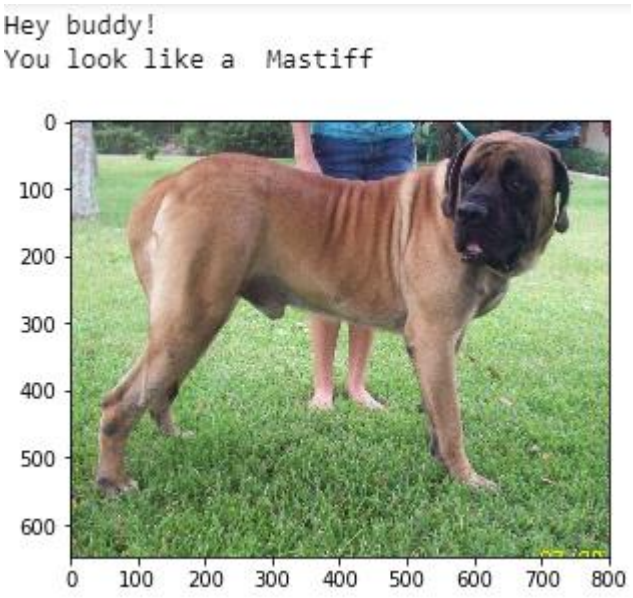
Test Loss: 0.862247

Test Accuracy: 74% (623/836)

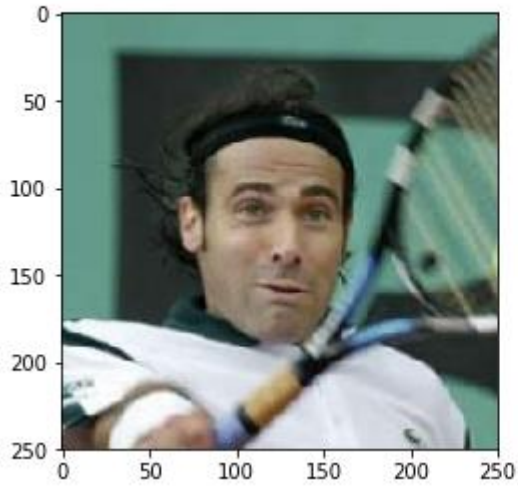
The accuracy can be made increased by increasing the number of epochs. Resnet101's last convolutional output is given as the input of the model.

Model Evaluation and Validation Human Face detector:

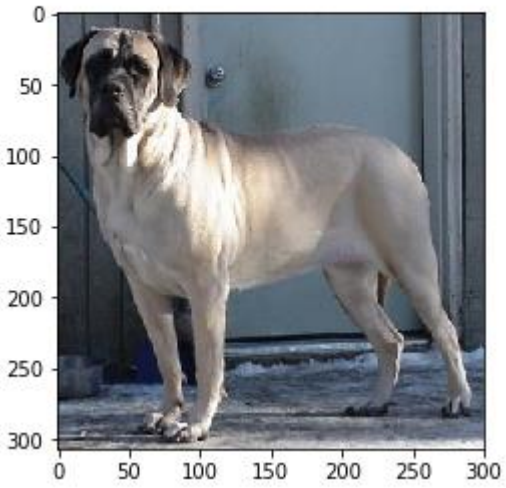
The model detected human images using the human_face_detector function b y making use of OpenCV's implementation of Haar feature based cascade classi fiers and the percentage of human faces correctly classified was 98% and the p ercentage of dog faces mistakenly classified as human faces was 17%. The mod el detected dog images using the dog_face_detector function by making use of pre-trained VGG16 model. The percentage of dogs detected in the first 100 images in human_files was 1.0% and the percentage of dogs detected in the first 100 images in dog_files was 100.0%. The CNN built from scratch gave an accuracy of 12%. The model correctly predicted breeds for 104 images out of 836 total images. The CNN model built using transfer learning gave an accuracy of 74% on test data. The model correctly predicted breeds for 623 images out of 836 total images.



```
Hey man!
HaHa If you were a dog you would look like a  Chinese shar-pei
```




```
Hey buddy!
You look like a  Mastiff
```



Justification:

I felt that the performance(accuracy) had improved tremendously while using the CNN built using transfer learning, this model gave an accuracy of 74% while the CNN model built from scratch gave an accuracy of 12%.

Scope of Improvement:

The model's performance by increasing the number of epochs. I would try to in crease the number of input images in the dataset, and I would try using a different model other than SGD. I would also try to increase the number of breeds as the current model works with 133 breeds of dog.

References used:

1. https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification
2. https://datascience.stackexchange.com/questions/24511/why-should-the-data-be-shuffled-for-machine-learning-tasks
3. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
4. https://www.quora.com/What-are-the-differences-among-AlexNet-GoogleNet-and-VGG-in-the-context-of-convolutional-neural-networks
5. http://wiki.fast.ai/index.php/Log_Loss
6. https://en.wikipedia.org/wiki/Convolutional_neural_network