Capstone Project

Abdelrahman Elghamry

Machine Learning Engineer Nanodegree

May 20th, 2020

Dog Breed Classifier using CNN

Project Overview

The main purpose of this project is to classify the different dog breeds. If dog owners are unsure about their dog's breed, a dog-breed classifier can help them identify their dog's breed. This would be especially helpful for in-experienced dog owners.and for that reason The Dog breed classifier is a well-known problem in ML. the classification is totally based on images and the goal will be to classify 133 dog breeds usingImage A convolutional neural network, CNN and computer vision that can classify an image according to its visual content , the finished project is able to take in an image. Identify if image is a human , dog or unknown. Then if a dog or human guess the dog breed, or specifically which one of 133 different breeds. Or in the case of a human guess which breed of dog the subject resembles. The images to be used for the data were also provided ahead of time.

Problem Statement

As we mentioned The goal of the project is to build a machine learning model that can be used within web app to process real-world, user-supplied images. The algorithm is used to detect Dog Breed or detect human and identify the resembling dog breed.

The Algorithm will flow through the following steps,

1. Take in a image
2. Decide if the Image if a Human, Dog or other.
3. Output to the user the result of running said image through the model
   a. If a dog inform the user that the image belongs to a dog and what breed.
   b. If the image is of a human identifying they are human and what breed the subject

may resemble.
c. If neither human or dog say it is unable to tell what the input image is.

That is the base jupyter notebook itself has a set of problems to solve in order to dive into CNNs (convolutional neural network) and transfer learning. With the dog breed classification simply being an overall wrapper to tie the tasks together.

The notebook itself breaks this down into 6 steps.

● **Import Datasets** - Simply download and load the provided image datasets

● **Detect Humans** - Create a function to detect human faces with OpenCV's implementation of Haar feature cascades(4).
● **Detect Dogs** - Create a function to detect dogs based on a VGG-16 model trained on ImageNet(5).
● **Create a CNN to Classify Dog Breeds (from Scratch)** - This task was to create, train & test a CNN built myself from scratch, aka no transfer learning, to identify dog breeds. The cited goal being to reach at least 10% accuracy
● **Create a CNN to Classify Dog Breeds (using Transfer Learning)** - This step is essentially the same as step 4 but in this case I had to use transfer learning. Such as using an existing model like vgg16 as a base. In this case the target was at least 60% accuracy.
● **Write your Algorithm** - Here the goal is basically to combine steps 2, 3 & 5. Achieving the initial problem outlined above to detect Human or Dog then guess which breed the image resembles.
● **Test Your Algorithm** - This is simple to test the Algorithm against a set of 6 images, three human and, three dog.

Metrics

The data is split into train, test and valid dataset. The model is trained using the train dataset. We use the testing data to predict the performance of the model on unseen data. We will use accuracy as a metric to evaluate our model on test data.

As mentioned prior this project is measured against its accuracy in detecting what breed a dog belongs to. In the case of the CCN built from scratch it was required to reach at least 10%. Then in the case of the model built with transfer

learning it needed to hit at least 60%.

Data Exploration

For this project, the input format must be of image type, because we want to input an image and identify the breed of the dog. The dataset has pictures of dogs and humans.

Dog images dataset: The dog image dataset has 8351 total images which are sorted into train (6,680 Images), test (836 Images) and valid (835 Images) directories. Each of this directory (train, test, valid) have 133 folders corresponding to dog breeds. The images are of different sizes and different backgrounds, some images are not full-sized. The data is not balanced because the number of images provided for each breed varies. Few have 4 images while some have 8 images.

Human images dataset: The human dataset contains 13233 total human images which are sorted by names of human (5750 folders). All images are of size 250x250. Images have different background and different angles. The data is not balanced because we have 1 image for some people and many images for some.

Data Set links:
Dogs - https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip

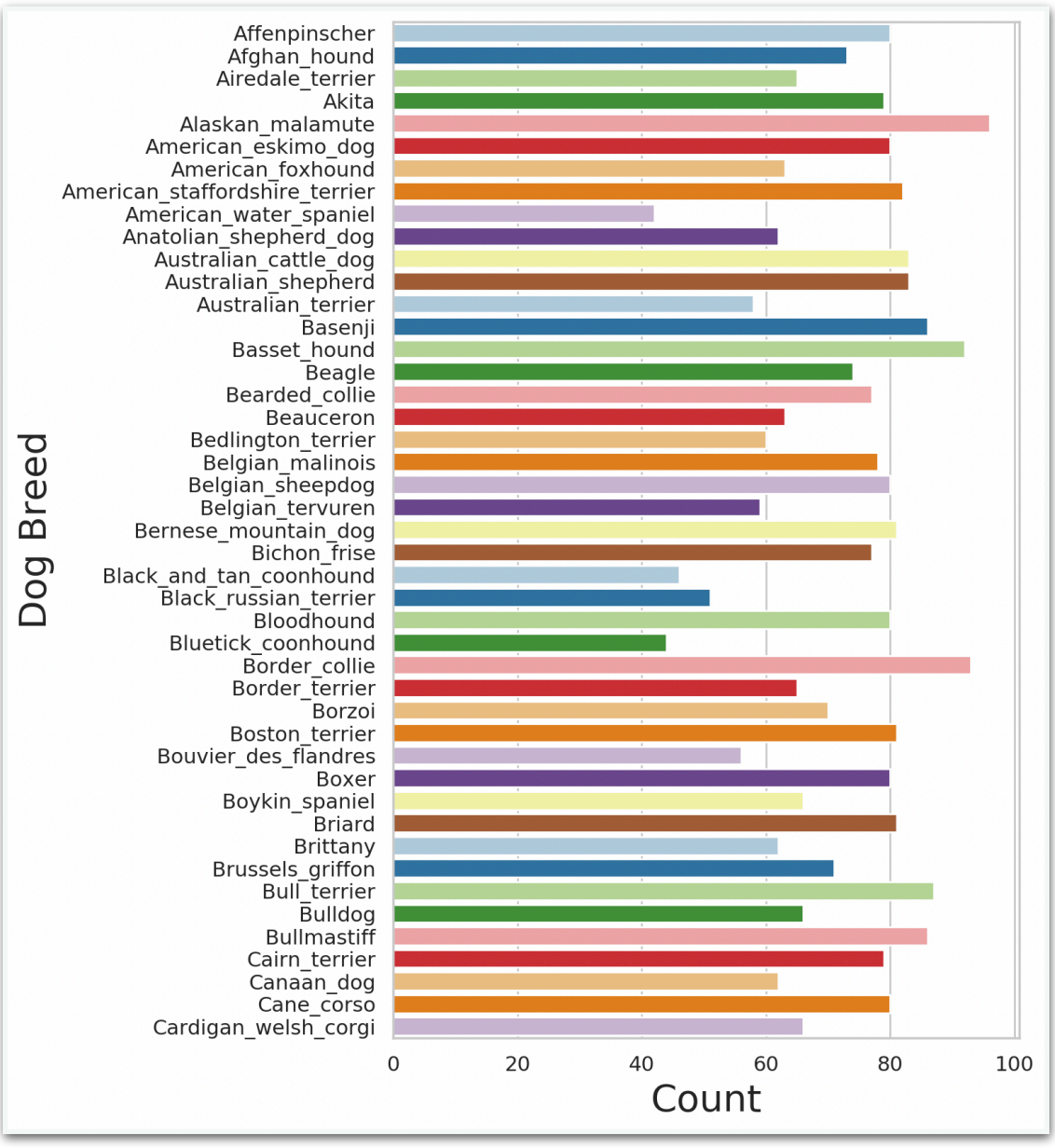
Humans - https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip

Examples from the dataset below,



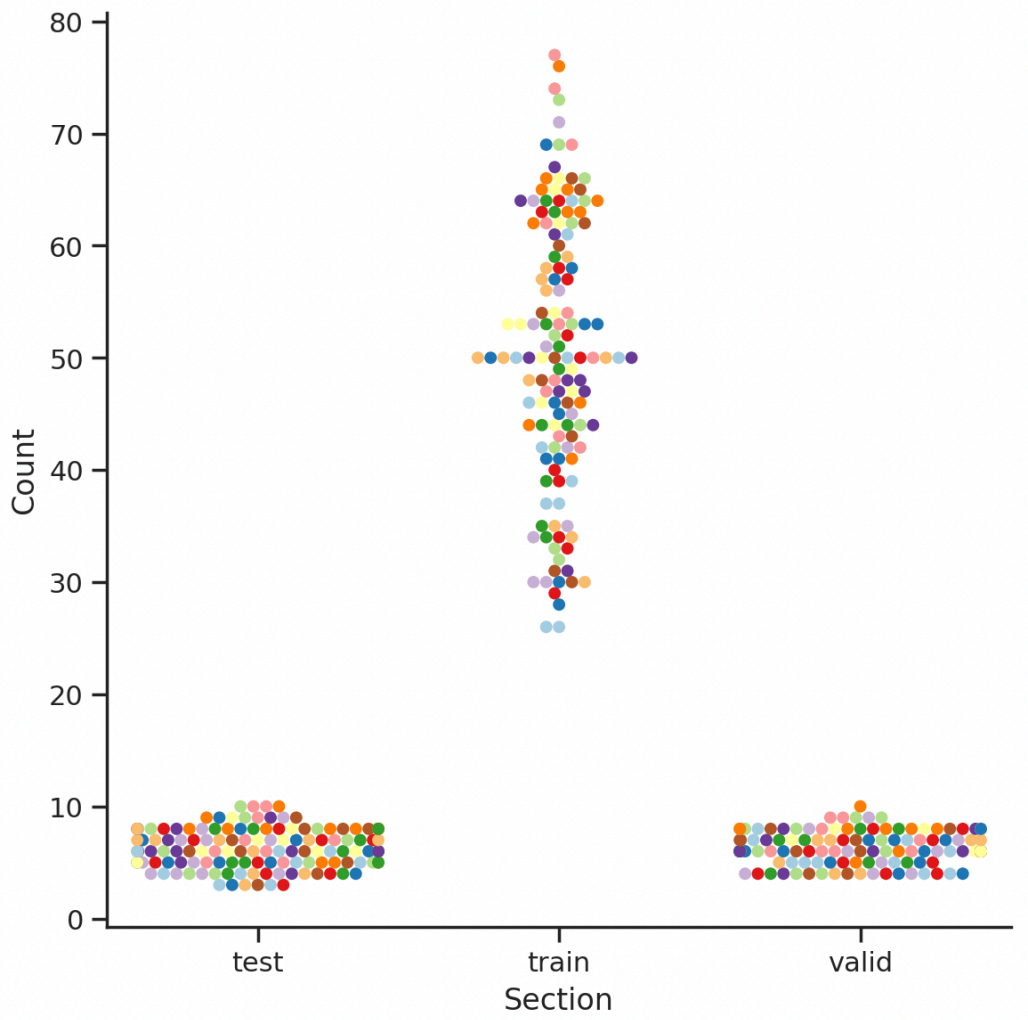
Exploratory Visualization

the task of assigning breed to dogs from images is considered exceptionally challenging. To see why, consider that even a human would have trouble to Classify the dog breed . According to the blow chart you can get an idea how many images the various dog breeds have. Along with the quantity of images. From this you can see the training data isn't exactly equal when it comes to the amount of data for each breed. The model gets over 70 images to learn what an Alaskan Malamute looks like but less than 30 to learn what a Norwegian Buhund looks like. This is something to bare in mind as the final model is likely to end up being better at recognizing some breeds over others.



This plot contains 45 dog breeds.

On the blow plot you can see the count of images of each dog breed across the sections training, test and valid:



Algorithms and Techniques

The used Algorithm and Techniques in our project are based on six steps.

1- we import and explore the datasets in order to understand how to use them and choose the proper algorithms.

2- we implement a Haar feature-based cascade classifier using OpenCV in order to detect faces in the human dataset.

3- we will use a pre-trained VGG16 model in order to detect dogs in the dogs dataset

4- we will create our architecture that uses CNN based on 4 layers and batched after each layer in order to classify the 133 dogs breeds and have an accuracy greater than 10%.

5- we will use the transfer learning technique in order to a pre-trained ResNet50 architecture and continue the training with the dogs dataset. The minimum accuracy required is 60% on the test set.

6- we will write a custom algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither.

Then we will Test the Algorithm with some random images found online.

Benchmark

The CNN model created from scratch must have accuracy of at least 10% and the CNN model created using Transfer Learning must have accuracy of at least 60%. According to my CNN model with 4 layers I got 24 % accuracy for the Scratch algorithm after training 20 times , And I got 75 % using transfer Learning “ResNet” .

Data Preprocessing

I put my image data through the following preprocessing.

1- cropping the images randomly to have the size of ImageNet images (224px).
2- adding random contrast: some dogs are dark so if we add contrast to the image, it will be more clear.
3- adding random rotation and flips.
4- adding normalization to help CNN perform better.

Finally, all the images are converted into tensor before passing into the model.

Implementation

For the Haar feature-based cascade classifier is used a computer vision pre-trained face detector .

For the VGG16 pre-trained model, is used PyTorch in order to make the predictions .

Then I have built a CNN model from scratch to solve the problem. The model has 4 convolutional layers . I chose the first output layer to be 64. What I found was that my model trained really slowly so I modified my first output layer to 32 , I tried two versions. The first version had four layers but the first layer had a stride = 1 and the other version had stride = 2. I got better training and testing performance when using stride = 2. We have two fully connected layers that finally produces 133-dimensional output. A dropout of 0.3 is added to avoid over overfitting. After that I choose the resnet model in CNN using transfer learning. The only modification I made to the model was to change the final connect layers output to 133 for the 133 dog breeds that we're trying to predict.

Refinement

The CNN created from scratch have accuracy of 24%, Though it meets the benchmarking more than 10% , the model can be significantly improved by using transfer learning. To create CNN with transfer learning, I have selected the Resnet50 architecture which is pre-trained on ImageNet dataset, the architecture is 50 layers deep. The last convolutional output of Resnet50 is fed as input to our model. We only need to add a fully connected layer to produce 133-dimensional output (one for each dog category). The model performed extremely well when compared to CNN from scratch. With 20 epochs, the model got 75% accuracy. More than needed “60%” . These are some parameters toned during the process: Training epochs, learning rate, dropout value .

Model Evaluation and Validation

Human Face detector: The human face detector function was created using OpenCV's implementation of Haar feature based cascade classifiers. 98% of human faces were detected in first 100 images of human face dataset and 17% of human faces detected in first 100 images of dog dataset.

Dog Face detector: The dog detector function was created using pre-trained VGG16 model. 100% of dog faces were detected in first 100 images of dog dataset and 1% of dog faces detected in first 100 images of human dataset.

CNN using transfer learning: The CNN model created using transfer learning with ResNet50 architecture was trained for 20 epochs, and the final model produced an

accuracy of 75% on test data. The model correctly predicted breeds for 635 images out of 836 total images.

Accuracy on test data: 75% (635/836)

Justification

All the model trained meet the expectation of the project . The model created using transfer learning have an accuracy of 75% compared to the CNN model created from scratch which had only 24% accuracy.

Improvement

Three possible points of improvement for your algorithm :
1-We can increase the number of training images, so that the models learns better
2-We can change parameters like the optimizer, the learning rate, or increase the number of epochs.
3-We can apply more data augmentation , may increase the accuracy. In agumentation mainly transpose was not added and that may be consider as improvement

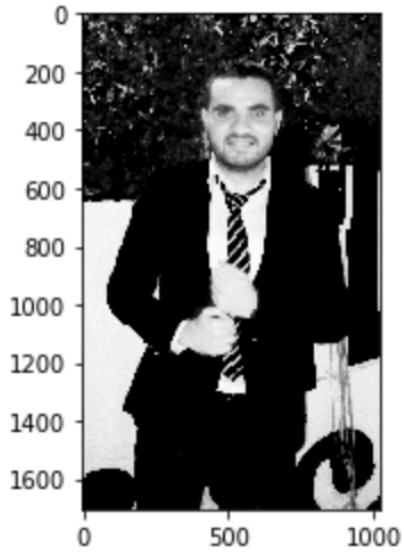
Conclusion

Free-Form Visualization

Here some results from the final algorithm that combine the face detector with the ResNet50 dog breeds model:

```
In [79]: run_app(my_images[0])
```

```
Human Detected!
```

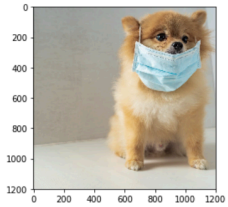



```
You look like a ...
 Lakeland terrier
```

```
Human Detected!
```


```
You look like a ...
 Neapolitan mastiff
```

```
In [81]: run_app(my_images[2])
Dog Detected!
```



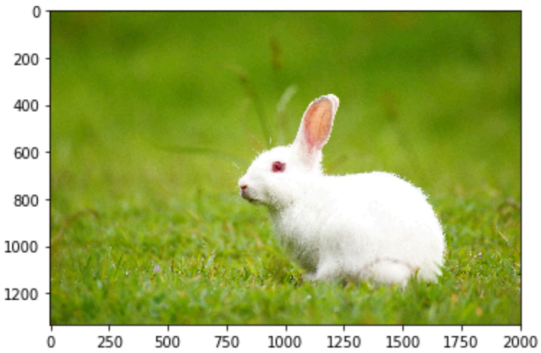

```
You look like a ...
 Pomeranian
```

```
In [82]: run_app(my_images[3])
Dog Detected!
```


```
You look like a ...
 American staffordshire terrier
```

In [83]:
```
run_app(my_images[4])
```

```
Neither a human, nor a dog!
```




```
Tell us if we made a mistake and your picture is a dog or human
```