MACHINE LEARNING ENGINEER ND

DOG BREED CLASSIFIER USING CNNs

Report

Yasir Almutairi

CAPSTONE PROJECT

31 December 2020

Project Overview

Identifying a dog breed is a difficult problem, the problem I'm trying to solve is categorizing the breed of dog, currently only identifying 133 types of breeds. After being provided by an image of a dog the algorithm would return a predicted breed. If however it's of a human image it will return the name of a dog breed that resemble. This problem can be categorized as multi-class classification problem. It also worth noting that identifying different dog breeds is not an easy task. Some dog breeds only have a small variation which is difficult for us human let alone computes.

Problem Statment

My goal in this project is to define a pipeline function that describe the behaviour of what could be a web-app, given an image provided as input by the user, the output would in two ways:

Dog face detector: the goal of this function is to return the predicted breed of a dog.

Human face detector: The goal of this function is to return the resembling dog breed. As mentioned in the project overview section.

Metrics

Being a multi-classification problem as discussed earlier. It would be ideal to use a classification metrics such as Accuracy to evaluate the performance of the model. The model is trained on the training dataset, and evaluate the performance of this model on unseen data, The test data. Which give us an estimate of how it would perform on the real world data.

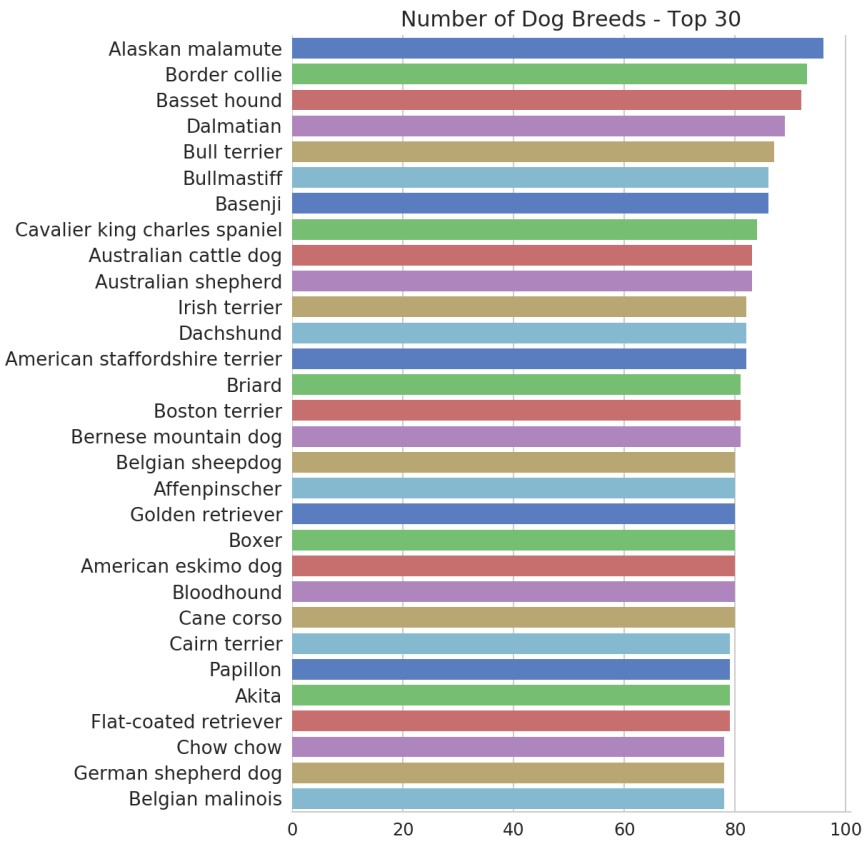
Accuracy = Number of Correctly classified dog breeds / All predictions

Data Exploration

There are two datasets both provided by Udacity, a human images dataset and dog images dataset. The number of images in both of theses datasets as follows:

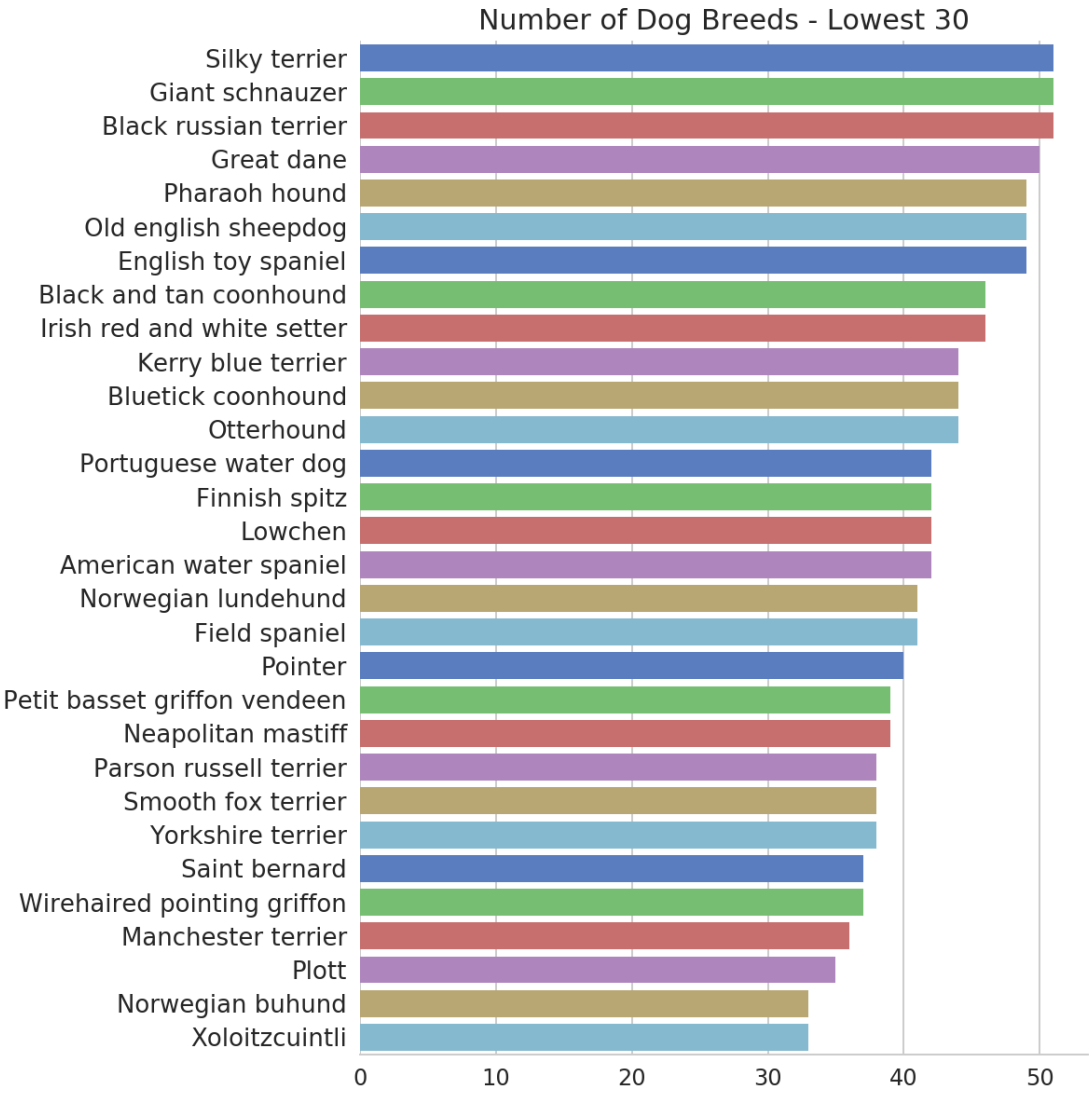
- *Dog images datasets*: 8,351 total images of dogs, after splitting into training, testing, validation. Training contains 6,680, Testing contains 836, and lastly validation contains the remaining 835 images.
- *Human images datasets:* 13,233 images of humans, All of these images are of size 250x250 pixels. And they are sorted by the names of humans.

As mentioned earlier the goal is to classify 133 types of breeds so it would be ideal to see the distribution of dog breed in the datasets.



Top 30 Dog Breeds Examples

What the this figure tells us is we have more than 80 examples -images- of dogs of Alaskan malamute. And so on. The next figure will show the lowest provided examples of dog breeds.



Lowest 30 of Dog Breeds

Algorithms

In performing multi-class classification, I will use Conventional neural Networks which is a deep learning algorithm, A Conventional is a is a mathematical operation that creates weights, is also referred as kernel or filter. The filter that is created is smaller in size than the input image. This filter pass through each subsection of the image and produced a value until it completed its pass.

As detailed in Problem statements section the solution is followed by 3 key steps.

1. Detecting if the image provided contains as Human in this task I will be using an OpenCv pre-trained model Haar cascade classifier. Which will be able to correctly detect Human faces in a given image.
2. Detecting if the image provided contains a dogs, in this task I will be using another pre-trained model, VGG16, That has been trained on ImageNet, a large dataset that is used for image classification.
3. After detecting and determine if the image contains either/both faces of humans and dogs the image is then passed to the model to predict the breed of the dog.

Benchmark

I have defined a CNN from scratch with the aim of at least performing on unseen data - Test dat - of accuracy of 10%. The benchmark model that I built and trained scored on test data with accuracy of 21%. What is this means is the model managed to learn some characteristics of the dogs. In other orders it didn’t randomly guessed the dog breed.

Data Preprocessing

To make the images in a format acceptable by the model. I have used Image Augmentation to add randomness to the images and improve the model performance by avoiding overfitting, these image augemtnaiton steps have been applied to all of the datasets, training, testing, validation. Through randomly resizing crop to 224x224, and randomly flip the image on horizontal axis. And to to transfer the image into a tensor which the datatypes that is required by PyTorch to train the model. And finally in a last step to normalize the pixels of the image.

Implementation

The CNN from scratch I defined had 3 Convolutional layer followed by 2 linear fully connected layers, and max pooling layer of size 2x2. With a dropout probability of 30% which would help in avoiding overfitting. And an activation function of Relu.

The Architect of CNN can be view in the following figure.

```
Net(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1
))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1))
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_
mode=False)
  (fc1): Linear(in_features=6272, out_features=500, bias=True)
  (fc2): Linear(in_features=500, out_features=133, bias=True)
  (dropout): Dropout(p=0.3)
)
```

CNN Architecture

Refinement

The above defined CNN model Accuracy is at 21% for test data, which is as mentioned earlier unseen data with goal of getting an approximation of how the model would perform in the real-world. The model however can improved through using transfer learning, but before moving on its worth noting that creating a deep learning network to classify dog breed is not an easy task, as the difference between some dog breeds is so small that it completely unnoticeable by non-experts.

To get better results I used one of PyTorch pre-trained model, ResNet50.

Which has 48 convolution layers, and 1 max pooling layer, and 1 average pooling layer. The ResNet50 won the AlexNet competition back in 2012. To make use of the pre-trained conventional layers I added a linear layer to map the output of conventional layers to 133 nodes. In our case dog breed classes. And then trained it on our dog training dataset, which managed to score an accuracy of 75% on the test set. Which is a significant improvement form 21% score of our CNN 3-conventional layers.

Model Evaluation

Here I will outline the scores of the algorithms used in this project. As mentioned in Alogrithims section there three key steps in this project so I will outline their score as follows:

A. *Human face detector*:

implemented by using OpenCv pre trained model, managed to detect 98% of human faces in the human files datasets. In other words it failed to capture 2% of human faces. And it misclassified 17% of dogs as human.

B. *Dog detector:*

implemented by using VGG16 pre-trained model, it managed to capture 100% of dogs in dog file image, and it didn't classify any human as dog.

C. *CNN from scratch:*

With three conventional layers and two linear layers. This model managed to correctly classify 180/836 of dog test images correctly, in other words it accuracy was 21%.

D. *CNN by using Transfer learning:*

ResNet50 pre-trained model with 48 convolution layers, and 1 max pooling layer, and 1 average pooling layer. Managed to correctly classify 633/836 of dog test images, with accuracy of 75%. It is a big improvement from 21%.

Justification

As mentioned earlier the improvement from using CNN from scratch to using a transfer learning is significant, ResNet transfer learning model can capture 75% of any dog breed correctly compared to CNN from scratch where it only capture 21% of dog breeds correctly a slightly better than guessing.

Improvements

Before improving the model I believe the images can be tried on different set of image augmentiaon which would help capturing the randomly and different scenarios of dog images in the real-world. And the model can be further optimized by using hyper-parameter tuning, and trying a diffident model architect such as inception_v3.

Reflection

To summarize the workflow I have done in this project:

1. Understanding the problem.
2. Importing required data.
3. Developing human face detector.
4. Developing dog detector.
5. Augmenting the image.
6. Define a CNN model and trained it.
7. Define a Transfer learning model and trained it.
8. Design the behaviour of the application.
    1. Takes input as image
    2. Detected human face
    3. Detect dog
    4. Predict dog breed
    5. Return prediction

For me the most challenging part for me was working with images and understanding different types of augmentation and how it affect the model and its performance. My first choice of image augmentation steps resulted in the model training never went lower than 8.9. and test accuracy never managed to score a result even close to 10% however after iterations I managed to fix my errors. It was great learning journey I spent a lot of time reading through amazing papers of the pre-trained models.