Machine Learning Engineer Nanodegree

Capstone Report - CNN Project: Dog Breed Classifier

Chukwuemeka Ezumezu

January 29th, 2020.

Project Overview

This project is among the options provided by Udacity as part of capstone project for Machine Learning Nanodegree Degree Program, with the aim of creating an application that can be able to easily identify dog breeds when a dog image is provided as an input and predicts the closest resembled dog when an image of a human is provided as an input (just for fun). It also indicates an error when neither dog nor human image is provided.

With advancement in deep learning technology, we now have applications that now out-perform humans in identifying objects. The problem of people finding it difficult to properly identify and understand their dogs is part of what prompted this project. The trained image classifier will be intergrated into a smartphone and web app, where users of the app can use it to easily identity dog breeds of the image of dog taken with their phone or personal computers, it can also predict the dog breed camera is looking at without snapping the picture.

Dataset provided by Udacity in a GPU-enabled workspace, which consist of 133 dog breed categories, 8351 dog images and 13233 human images, was used to train the model.

Problem Statement

In this project, we make use of pytorch to build a Convolutional Neural Network (CNN) that can make dog breed predictions. We first built a CNN from scratch and obtain an accuracy of 12% on the test dataset, which is reasonable being a simple architecture with fewer layers for feature extraction. That was done to enable us to understand how CNN works. We then restructure and trained a transfer learning, resnet152, (Transfer learning can save us a lot of time and computational resources being that we can not start developing the model structure and training from scratch, It gives a very high accurate prediction) model on our dataset, test the model on our dataset and get accuracy of 88%. With the accuracy of upto 88%, we expect to approximately to correctly classify 9 out of 10 dog breed correct which goes a long way of us achieving our purpose of developing an app that will solve the problem of identifying dog breeds.

Metrics

We used a balanced dataset of an average of 50 images per class of dog breed. Since it’s a classification problem and we also have a balance datasets across the categories, we used accuracy as our evaluation metric.

The accuracy was calculated with formula:

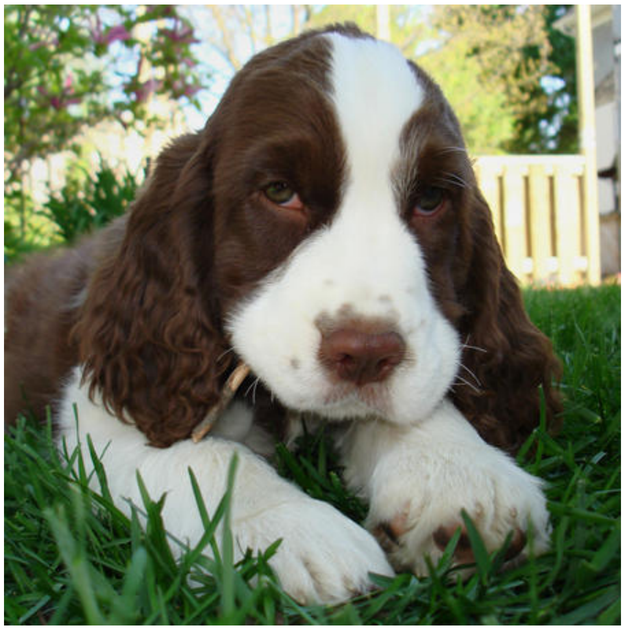
Test Accuracy = correct classified items / total classified items.

Data Exploration and Visualization

The datasets provided in Udacity Dog Project Workspace for both the dog and human dataset contained 8351 total dog images with 133 different breeds and 13233 total human images. Which the 8351 dog images, is split into training, validation and testing ratio of 80: 10: 10 respectively. That is 6680 dog images for training, 836 dog images for validation and 835 dog images for testing.

Since we have a balanced dataset with an average of about 50 images per class of dog breed and we are solving a classification problem, we used accuracy as our evaluation metric. Otherwise we would have also use precision or F1 Score, if we do not have a balanced dataset.

Let us take a look at some of the images:



The image sizes are different, likewise the orientation . To pre-process the images, we resized them to 224x224 pixels so it can fit our network architecture.

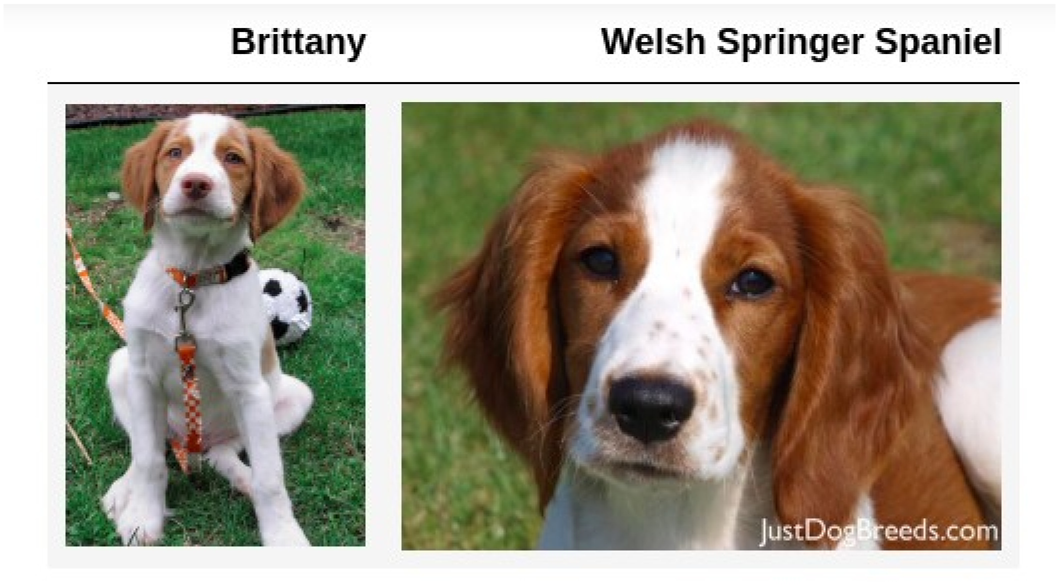
Algorithms and Techniques

We used OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images. First we converted BGR image to grayscale to be able to easily find faces in the image, print number of faces detected in the image then get bounding box for each detected face. We then converted the image back to color image and display it along with bounding box. Making it easily to detect human faces in a picture with higher accuracy.

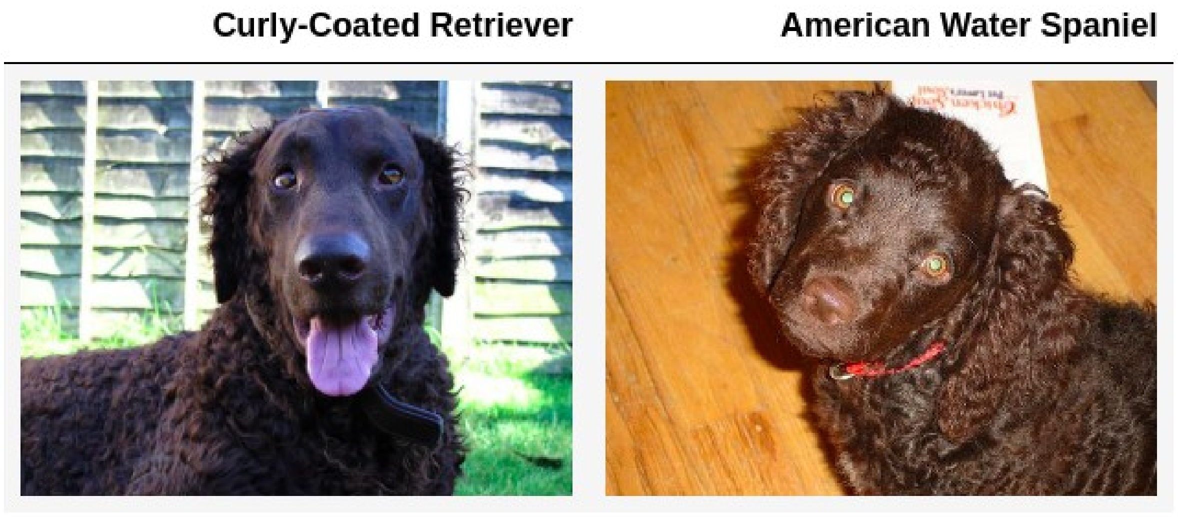
We then used this procedure to write a function that returns True if a human face is detected in an image and False otherwise. This function was tested on the first 100 dog and human images and got 98% of human faces detected from human files and 17% detected on dog files which is still an acceptable performance.

We used pretrained VGG-16 model to implement dog detector function which detected 99% of the first 100 dog images as dog and 0% of dog was detected on first 100 human images.

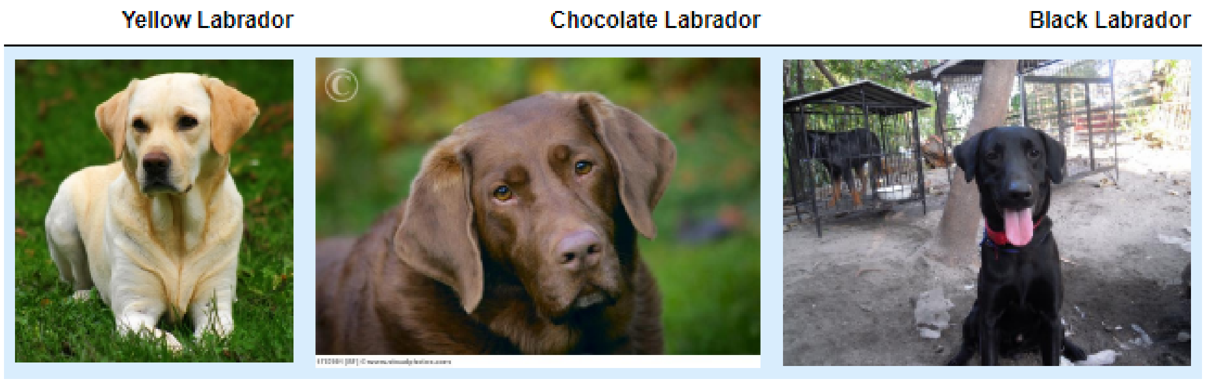
It worth noting that task of assigning dogs breed from images is considered exceptionally challenging, even to a human. Example would be distinguishing between a Brittany and a Welsh Springer Spaniel.



Finding other dog breed pairs with minimal inter-class variation was not difficult for the algorithm we used(for instance, Curly-Coated Retrievers and American Water Spaniels).



Our algorithm was able to also cover high intra-class variation of labradors coming in different colors (yellow, chocolate and black), was able to determine how to classify all of these different shades as the same breed.



Benchmark

Our benchmark of fine-grained classification which was an attempt to identify plant species by Belhumeur, identifying a bird species by finding key points along the beak, eyes, wings, feet, and tail, and building features around them and a 2012 paper by Liu attempted dog breed identification using a similar approach. All has a balance training dataset across its categories and are all evaluated with accuracy metric, our obtained solution will also be evaluated with accuracy metric and will be aiming to greatly surpass the 57% average by the previous projects.

Data Preprocessing

The preprocessing of data, like resizing and cropping, have been state on Data Exploration and Visualization section above. Data preprocessing in pytorch also involves normalizing mean/std values.

Implementation

We first create a CNN dog breed classify from scratch and attain a test accuracy of 12%.

- We used 5 convolutional layers to classify every input image. Each layer have an input and output channels kernel size of 3, default stride of 1 and padding of 1 (3 X 3 filter center itself but misses the border pixels by 1, that's why the padding is 1).

- The first input channel of 3 represent a colored image (RGB) with an out channel of 16 different filtered images. Then the subsequent channels take the output of previous layer as an input and produce an output (in base 2 e.g 2^4, 2^5).

- Relu activation was applied after each convolutional layer.

- We used max-pooling layers to shrink the x-y dimensions of every input (input to every convolutional layers after the first, convolutional, layer).

- We used Batch Normalization (put data to a standard scale) to normalize the output of activation function.

- 2 fully connected layers was used to classify the image into the final class of 133 classes (different dog classes)

But this implementation of 12% accuracy on test dataset will not give us a good result, especially in production. So we decided to refine our model by using transfer learning.

Refinement

We will then Create a CNN classify using transfer learning which is going to serve as the main classifier for the project.

Having used resnet152 before for Flower Image Classifier Project during Pytorch scholarship challenge getting a good accuracy of 94.74% on test dataset. After testing different other pre-training model, we found it appropriate for this project.

- We first defined a pretrained resnet152 model.

- Printed out the structure of the model to understand its fully connected layer before restructuring it into 133 needed classes.

- We ten restructured the fully connected layer into 133 needed classes by adding 2 fully connected layer using 10% dropout.

- We froze parameters so we don't backprop through them, So that only the fully connected layers will be trained.

- Replacing with new classifier(fully connected layers).

- Checking the model again to make sure We have the right structure.

- After training our model for about 20 epochs, we have any accuracy of 88%.

With an accuracy of 88% on test data, that is an approximate correct prediction of 9 out of 10. We adapted that to be good for production.

That being said, due to time constraints and due to the time it takes to train a neural network, we were not able to perform many iterations on our technique. We can still further train networks with a different architecture and do more batch iteration to see the approaches that will give us a better result and adapt to it.

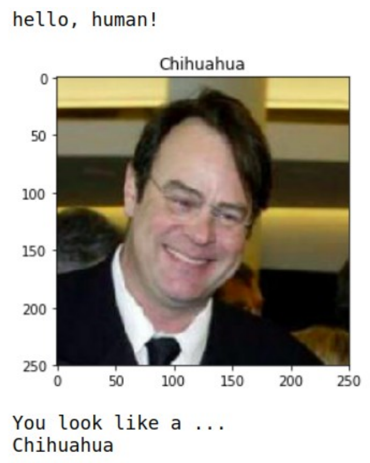
Model Evaluation and Validation

After training our model for about 20 epochs, we have a test accuracy of 88% and our model performed as expected when tested on a new image, though not perfect.

Detecting dog breed with minimal inter-class variation was not difficult for our model. It(model) was also able to detect high intra-class variation of breeds coming in different colors.

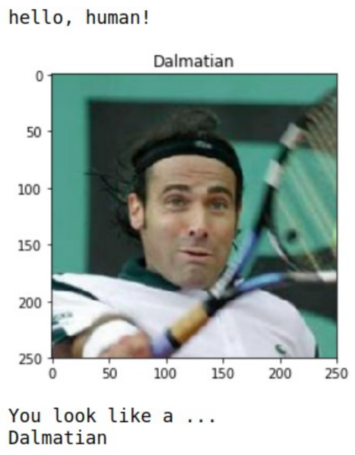
When six image of 3 dogs and 3 human was tested, Our model was able to correctly predict the dog breed and the breed which the human image Is closely resembled(the fun part of it) was also predicted.

Human-dog resemblance predictions:



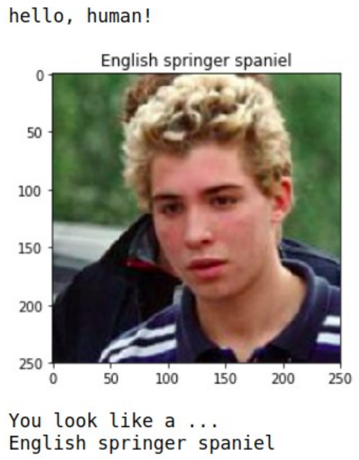
```
hello, human!
```

```
You look like a ...
Chihuahua
```



```
hello, human!
```

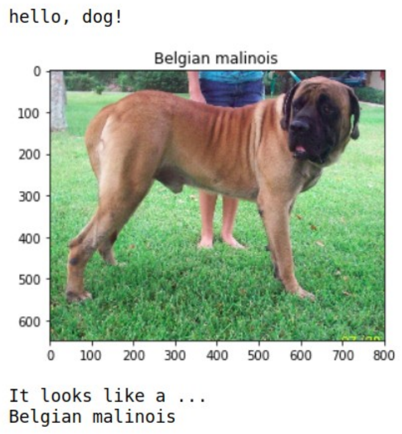
```
You look like a ...
Dalmatian
```



```
hello, human!
```

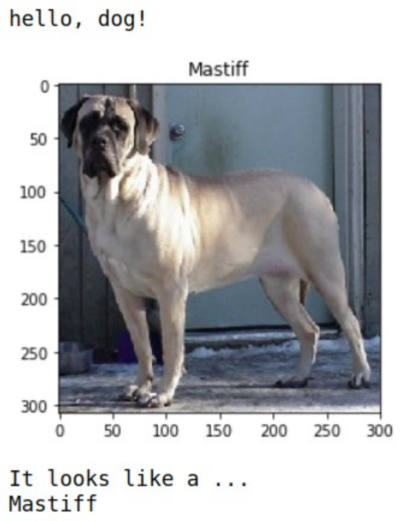
```
You look like a ...
English springer spaniel
```

Dog Breed predictions:



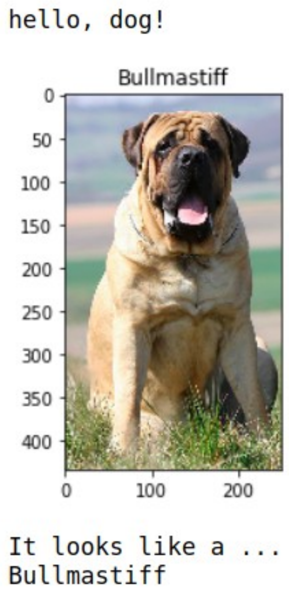
```
hello, dog!
```

```
It looks like a ...
Belgian malinois
```



```
hello, dog!
```

```
It looks like a ...
Mastiff
```



```
hello, dog!
```

```
It looks like a ...
Bullmastiff
```

Justification

Due to the capacity of our model to effectively predict the correct breed for more than 80% of the time, which surpasses the average predict of our benchmark (57%), given the high number of breeds in our classification problem and also given the high variability both between and within the 133 different breeds contained in the dataset. We considered our outcome to be a successful one and can adequately solve the dog breed classification problem.

Though there is still room for improvement. Due to time constraints and enormous time it takes to train a neural network, we were not able to perform some iterations on our technique (like batch iterator, trying out different network architecture). We can also improve the amount of our training dataset to obtain more refined and fine tuned result.

Neural networks still remain formidable classifier for the future that gives a high accuracy prediction over other traditional classifying techniques.