Dog Breed Classifier Capstone Project Report

October 24th, 2020

Ahmed Gharib

Machine Learning Engineer Nanodegree

Udacity

Table of content

I. Definition **2**
Project Overview 2
Problem Statement 2
Metrics 3
II. Analysis **4**
Data Exploration 4
Human Dataset: 4
Dog Dataset: 4
Exploratory Visualization 5
Algorithms and Techniques 7
Benchmark 9
III. Methodology **10**
Data Preprocessing 10
Implementation 10
IV. Results **12**
Model Evaluation and Validation 12
V. Conclusion **15**
Reflection 15
Improvement 15
References **16**

I. Definition

Project Overview

Recognizing dogs’ breeds according to specific physical characteristics is a challenging task for humans, since There are several breeds that most people have never seen before, or even heard of. Some are newly registered, and some are just less common in the United States.

To mention Some of these dogs, there are (Beauceron, Canaan Dog, Cesky Terrier, Chinook, Dandie Dinmont Terrier and Lagotto Romagnolo), and many more.

In the United States alone, the AKC's dog breed list currently includes 190 dog breeds. Worldwide, the FCI lists 360 officially recognized breeds. These don't include experimental breeds that have yet to achieve official status. Official lists also don't include mixed-breed dogs, not even "designer" cross breeds like the goldendoodle (a cross between a golden retriever and a poodle) or the puggle (a mix of beagle and pug).

But on the other hand some people in the world have no clue about any of the dog breeds and can’t even tell the difference between them.

This may give an indication that the task of degs breed classification is a complex one, since remembering all dog breeds and distinguishing between similar ones cannot be easily done by humans. From here comes the need for computers to do such a task, machine learning pleasures to assist.

The task includes image recognition and classification as machine learning techniques to aid computer vision. A Convolutional Neural Network (CNN) will be used for the task.

Problem Statement

It’s an Image Classification problem where we need to classify images of dogs and define their breeds and if the image is for a human we classify it as a human and give the closest dog breed for this human image.

Metrics

Since we have unbalanced data sets, we have to take this into the count when calculating the loss function so I decided to go with CrossEntropyLoss criterion this criterion companies LogSoftMax and NLLLoss in one class other approach would give the same result is to use LogSoftMax for the finally output layer and use NLLLoss to compute the loss.

The loss function can be described as:

$$loss(x, class) = weight[class]\left(-x[class] + log\left(\sum_i exp(x_i)\right)\right)$$

However, we consider accuracy for model's evaluation,

Where accuracy is comparing the predicted labels for our images with the true labels and compute the percentage of our model accuracy as follow:

$$accuracy(y, \widehat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples-1}} 1(\widehat{y}i = y_i)$$

II. Analysis

Data Exploration

1. Human Dataset:

LFW folder with total 13233 images of humans distributed in 5749 folders, but there is variation in the number of images for each folder.

Worth to mention that data is not balanced here, there's only one picture for some people, where other people have more than one. Images also have different backgrounds and different angles, but they all have the same size.

2. Dog Dataset:

Distributed in 3 main folders with a total of 8351 images. Each folder contains 133 folders corresponding to dog breeds.

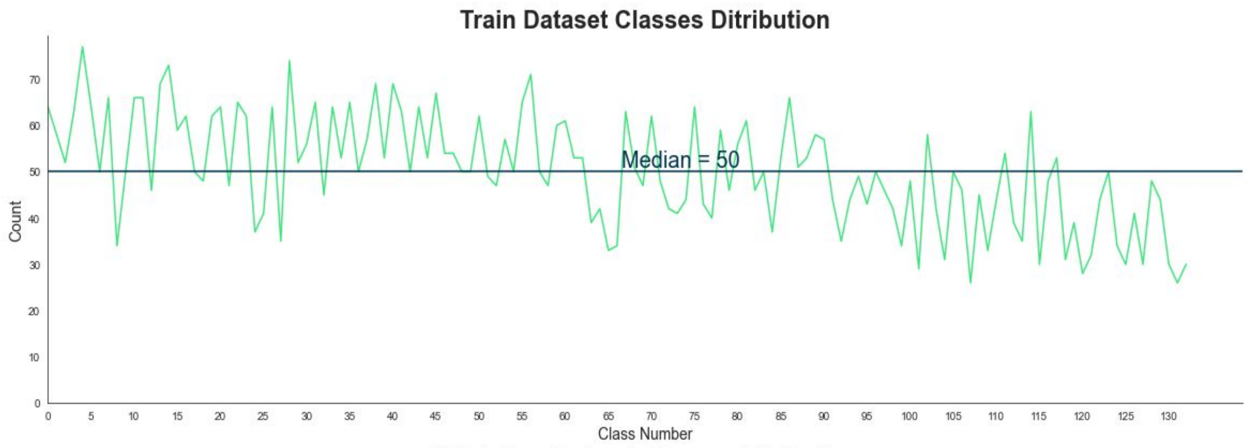
Noteworthy that the number of images provided for each breed varies, so the data is not balanced.

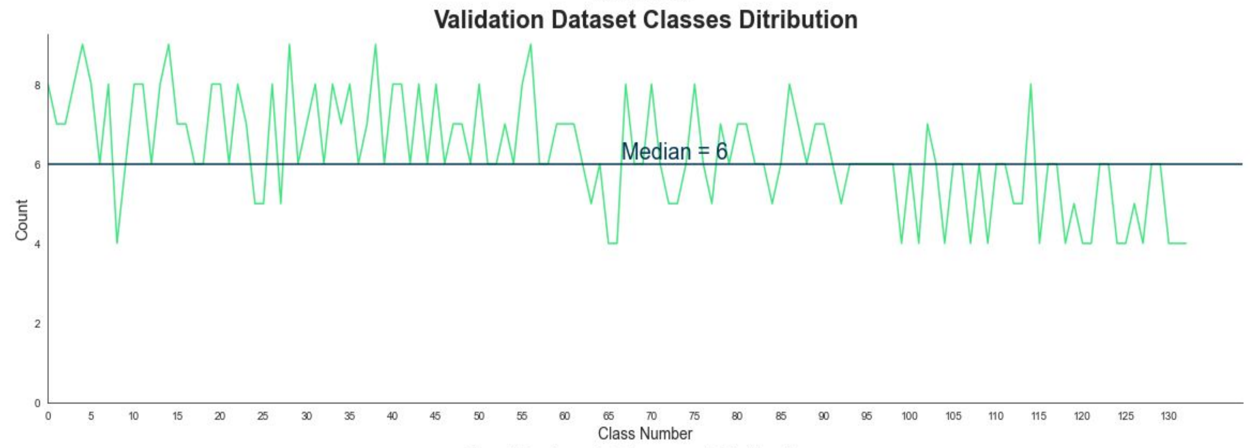
And we can see the distribution of number of images in each class in train, validation and test datasets as the following table:

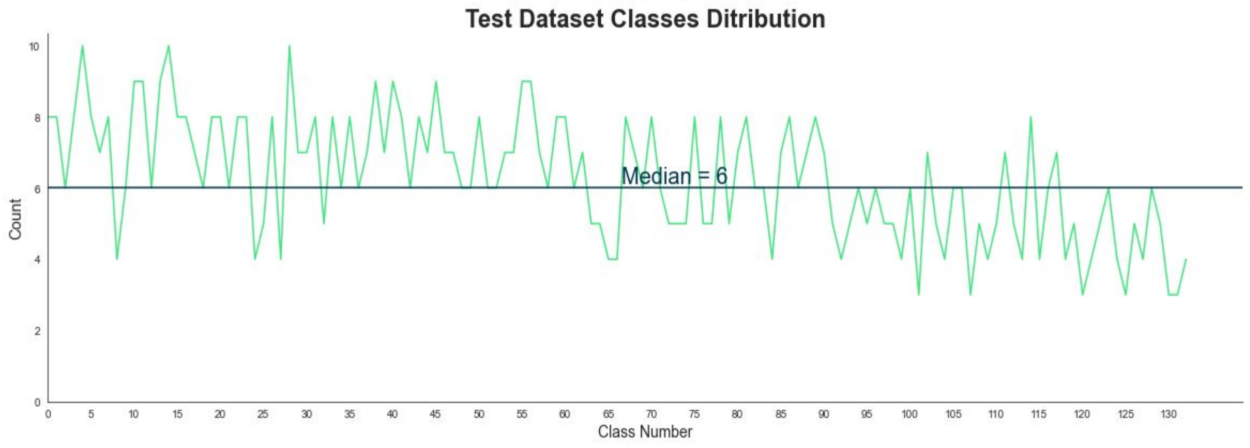
Statistic	**Train**	**Validation**	**Test**
Total images	6680	835	836
Percentage	80%	10%	10%
Total number of classes	133	133	133
Mean	50.23	6.28	6.29
STD	11.86	1.35	1.71
Min	26	4	3
25%	42	6	5
50%	50	6	6
75%	61	7	8
Max	77	9	10

Exploratory Visualization

- First let's display how the number of images in the dog datasets (train, valid, test) vary across classes.

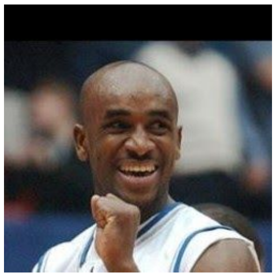






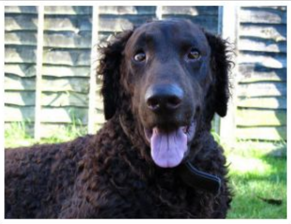
It's super clear now that our dataset is unbalanced.

- Samples from the human dataset

Andre Smith	Susie Castillo	Madonna	William Bulger
			

- Samples from the dog dataset

Brittany	Welsh Springer Spaniel

Curly-Coated Retriever	American Water Spaniel
	

Yellow Labrador	Chocolate Labrador	Black Labrador

Algorithms and Techniques

Our application should receive an image from the user and run the algorithm and then return 1 of 3 possabilities:

- if a dog is detected in the image, return the predicted breed.
- if a human is detected in the image, return the resembling dog breed.
- if neither is detected in the image, provide output that indicates an error.

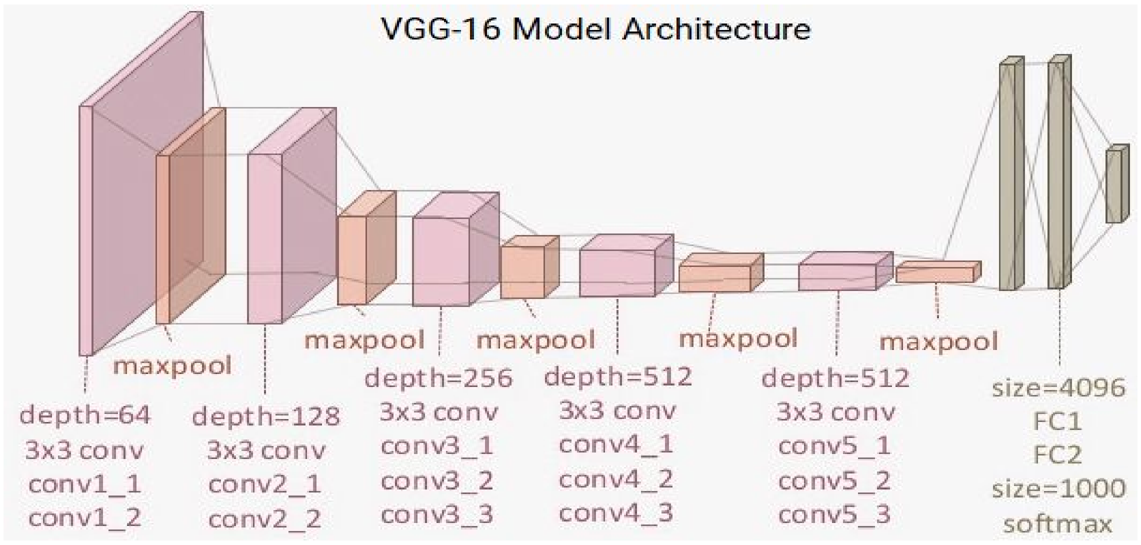
And in order to achieve this final output we will need to piece together a series of models to perform different tasks.

1. A pre-trained VGG-16 model to detect dogs in images.

   VGG-16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

   VGG-16 architecture:

   Inputs : RGB image of size (224,224)



   VGG-16 constructs of 2 main groups of blocks.

   - Feature Extraction used to extract the main features of the input image.

     Contains 5 convolutional blocks, the 1st 2 blocks have 2 Conv2D layers and each of the other 3 blocks have 3 Conv2D layers. All blocks end with a MaxPool2D with a kernel size of (2,2) layer to reduce the input shape for the next layer. Each Conv2D layer has a kernel size of (3,3) and uses the same padding.

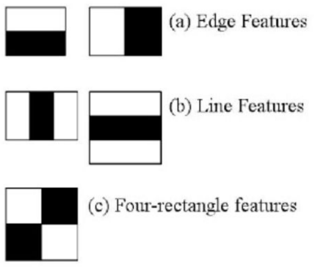
- Classifiers used to return the finally output (the probability of our 1000 classes)

  Three Fully-Connected (FC) layers follow a stack of convolutional layers: the first two have 4096 channels each, and the third (output layer) has 1000 channels corresponding to the number of classes.The final layer is the soft-max layer.

  All hidden layers are equipped with the rectification (ReLU) non-linearity.

2. Haar feature-based cascade classifiers to detect human faces in images

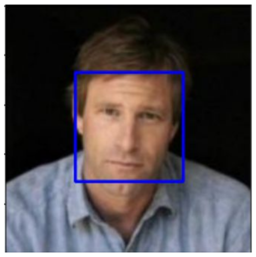
   Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

   image

   Haar features shown in the next image are used. They are just like convolutional kernels. Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle.

   Now, all possible sizes and locations of each kernel are used to calculate lots of features. But even for a 24x24 window results over 160000 features for only 200 of them are capable of detecting faces with 95% accuracy. Their final setup had around 6000 features. This will significantly reduce the computational power needed but still not so efficient. For this they introduced the concept of **Cascade of Classifiers**. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by-one. (Normally the first few stages will contain very many fewer features). If a window fails the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region.

3. A Convolutional Neural Network using transfer learning technique.

   For our image classification task we will use convolutional neural network and for this task in order to achieve the desired results we will need a large dataset as it's a case for all Computer Vision tasks and our dataset is small for such complicated task so we will need to use a technique like Transfer Learning.

Transfer Learning : The general idea of transfer learning is to use knowledge learned from tasks for which a lot of labelled data is available in settings where only little labelled data is available.

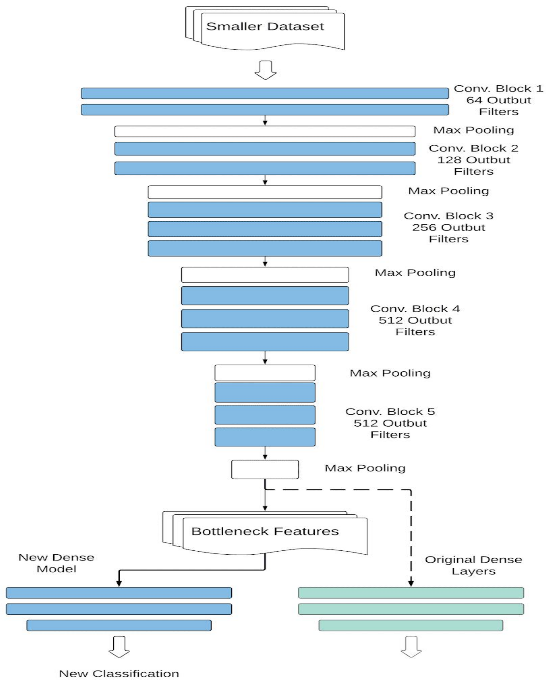
Again I will use A pre-trained VGG-16 but this time as a feature extractor only. VGG-16 has been trained using a large dataset and available as an open source pre trained model in many frameworks such as PyTorch and TensorFlow along with the trained weight.

We will use the same architecture.

Freeze the weights of the convolutional layers as they have already been trained in a huge dataset and can be used as a feature extractor.

Replace the last with another layer with the same output as the number of classes we have 133 dog breeds.

Train only the last fully connected layers.


Smaller Dataset
Conv. Block 1 64 Outbut Filters
Max Pooling
Conv. Block 2 128 Outbut Filters
Max Pooling
Conv. Block 3 256 Outbut Filters
Max Pooling
Conv. Block 4 512 Outbut Filters
Max Pooling
Conv. Block 5 512 Outbut Filters
Max Pooling
Bottleneck Features
New Dense Model
Original Dense Layers
New Classification

Benchmark

- A Model created from scratch with a minimum accuracy of 10% that confirms that the model is working better than a random selection as we have 133 classes and a random model will perform an accuracy of less than 1%. And use it as a start point to compare it with our final model using transfer learning.
- A Model created using transfer learning with a minimum accuracy of 60%. For such a hard task even for humans I think if we managed to achieve above 60% accuracy would be great.

III. Methodology

Data Preprocessing

1. Scaling and normalization

   Scale the image to the size of 224x224x3 as the VGG-16 model expect an input RGB image of size 224x224

   Normalize the values for each color channel to be between 0 and 1 for computational efficiency.

2. Data Augmentation

   In order to reduce overfitting data augmentation is used to provide more images to the model by slightly changing the original images and for this step I choose:

   - Random horizontal flip: to flip the image horizontally with a probability of 0.5.
   - Random rotation between than range of 30 degrees

   As we can see a batch of preprocessed images in the figure below.

Implementation

1. Dog detection using VGG-16 pre-trained mode.
   - Downloaded the model from PyTorch models and set the pretrained parameter to True as in PyTorch documentation to download the pre-trained parameters with the model architecture.

   ```
   VGG16 = models.vgg16(pretrained=True)
   ```

- Wrote a function to predict whether the picture has a dog or not.

  This function accepts an image path and loads the picture and runs preprocessing steps before passing it to the model.

  The VGG model takes the preprocessed image and returns an index from 0 to 999 inclusive corresponding to the 1000 classes of ImageNet.

- Finally another function was created that takes an image path and pass it to the VGG-16 prediction function above and takes it's output and checks if the output between 151 and 268 inclusive as if you check the Classes Dictionary, you will notice that 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'.

2. Detect Humans using OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images.
    - OpenCV provides many pre-trained face detectors, stored as XML files on github. I used one of these models to detect human faces.
    - Wrote a function that takes an image path and runs preprocess steps already implemented in OpenCV library to transform the image to grayscale .
    - Loaded the model using `cv2.CascadeClassifier('file path').`
    - Use `detectMultiScale` method that returns a list of the coordinates for the anchor box top left point and width and height for each face found in the image.
    - Check the length of this list if more than 0 returns True.

3. Dog breed classification model (Using Transfer Learning)
    - I used the same pre-trained VGG-16 model architecture and fine tuned it to suit the problem.
    - Freezed the feature extractors layers parameters.
    - Replace the final output layer number of output features to 133 instead of 1000.
    - And restrained the model.
    - Used `CrossEntropyLoss` function with $weights = \frac{1}{number\ of\ images\ per\ class}$ to take the classes imbalance into account when calculating the loss.
    - Used SGD optimizer to update only the classifier layers parameters.
    - Saved the best model parameters to use them later.
    - Wrote a function that takes an image path, runs preprocessing steps and path it to the best model and returns the class name of the top 1 predicted class along with it's index.

4. The final algorithm
    - Wrote a function that takes an image path and loads it.

- Pass it to the `dog_detector` function and if a dog is detected it passes the image path to `predict_breed_transfer` function and gets the class name and index and Display the image with it's predicted class name.
- If there was no dog detected it passes the image to `face_detecor` function and checks if a human face detected it passes the image path again to our `predict_breed_transfer` function and gets the class name and it's index and Displays the image as long as an image for the predicted dog breed.
- If neither were predicted the image displayed with the title "Sorry neither dog nor human were found here".

Finally worth to mention that while trying to train the final model I ran into "CUDA out of memory" error several times and tried to solve the problem by reducing the batch size with no hope but finally managed to pass this error by just closing visual studio code and tried again and this time was fine.

IV. Results

Model Evaluation and Validation

1. Regarding the Human Detector

   Was tested on a subset of 100 images from the human datasets and another 100 from the dog dataset and correctly detected humans in 96 of the human images and incorrectly detected humans in 18 of the dog images.

2. Regarding the Dog Detector

   Was tested as the same as the Human Detector and was able to correctly detest dogs in 95 of the dog images and didn't detect any dogs in the human images.

3. Regarding the Dog Breed Classifier:

   In order to get to the final model I tried several models first with different parameters.

   And here is a table summarizing the models optimization and loss functions and final results.

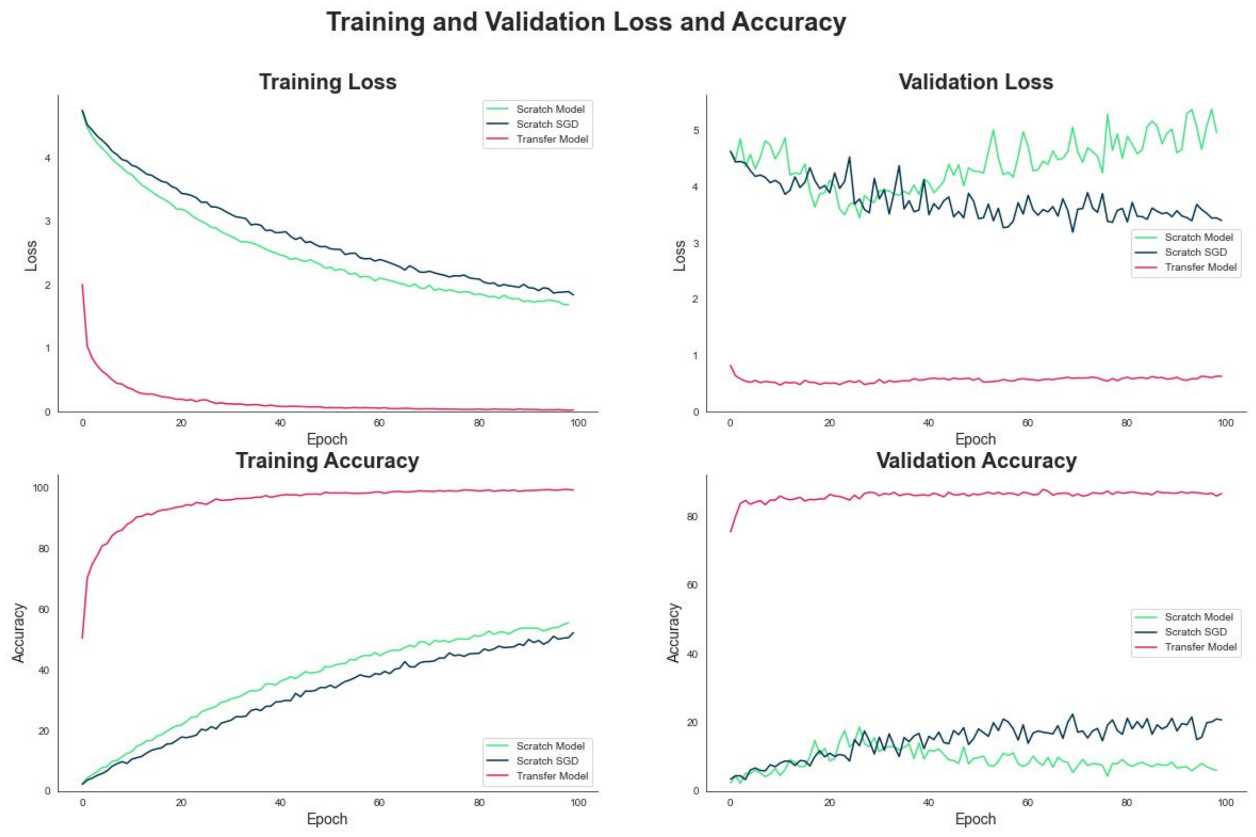
Model	**From Scratch with Adam**	**From Scratch with SGD**	**Transfer Learning VGG-16**
Optimizer	Adam	SGD	SGD
Learning Rate	0.0001	0.01	0.01
Weight_decay	0.01	0.01	0
Loss Function	CrossEntropyLoss	CrossEntropyLoss	CrossEntropyLoss

Num Epochs	100	100	8
Loss	3.265	3.793	0.636
Accuracy	33%	13%	83%

I Started by a simple CNN network with 6 Conv2D layers for feature extraction followed by 2 fully-connected layers and Relu activation function and used MaxPool2D layers after each of the Conv layers to reduce the output size and dropouts after each 2 layers with a probability of 0.5 and used batch normalization for the last conv layer and the first fully-connected layer.

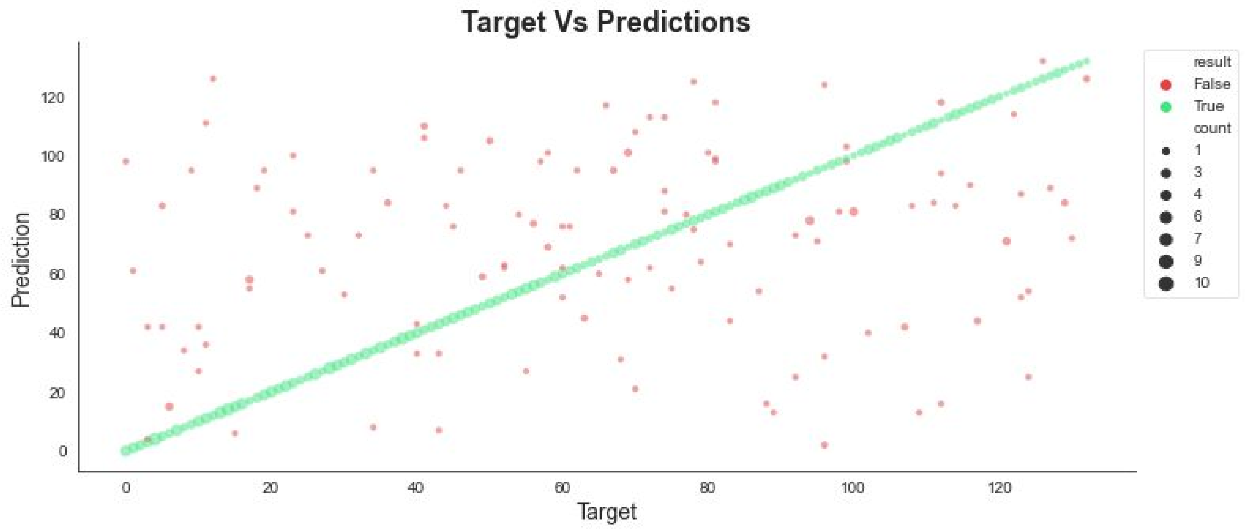
The first model achieved 33% then when I tried SGD the accuracy dropped to 13% then I tried Transfer Learning and achieved a way better 83%.

And the next figure compares the training and validation loss and accuracy for each of the 3 models for a 100 epochs.



The transfer model with a few epochs to train was able to achieve amazing results so I went with it in my final algorithm.

To further look in the predictions of the final model here is a plot for Target VS Predicted classes with the green color indicates True and red for wrong and the size of the bubble for the count of predictions.

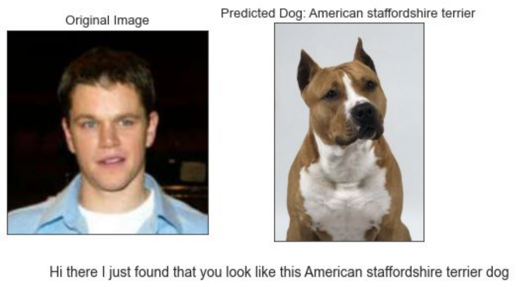


The predictions seem normal but it performs better for some classes than others.

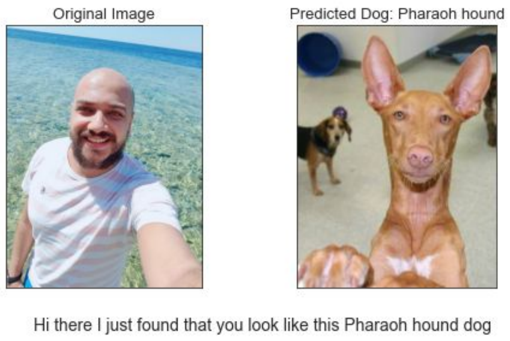
4. Finally the final algorithm

   I have tested it with several images from our datasets and some from my computer to compare the results and here is a sample from it’s output.

   From the dataset:


Original Image
Predicted Dog: American staffordshire terrier
Hi there I just found that you look like this American staffordshire terrier dog

From My Computer:


Original Image
Predicted Dog: Pharaoh hound
Hi there I just found that you look like this Pharaoh hound dog

V. Conclusion

Reflection

We had an image classification problem were we wanted to predict dog breeds and here is where computer vision and convolutional neural networks really shines this kind of problems wasn't easy to solve with MLP neural network as it doesn't take the patterns in the images into account and it's very computationally expensive but with convolutional neural network we managed to solve even more complicated problems.

And I want to mention also the role of Transfer Learning and how we can use a pre-trained model that was trained with another dataset to solve similar problems and how it helped to achieve significantly better results with just a little effort.

Finally I want to mention that I really enjoyed working in this project and practicing different techniques of deep learning and it helped me to better understand how neural networks work.

Improvement

1. First to enhance the closest dog breed to a human picture we can do:
    - Use OpenCV to find the human face.
    - Crop the face image before pass it to the model.
    - Train a separate model in dog faces and their breeds.
    - Use this model to predict the breed.
2. Second to enhance the dog breed prediction.
    - We can collect more images with equal numbers as possible for each class.
    - We can try several other pretrained model.
    - If we can get enough images for each class we can use the model architecture and train more model parameters, not just the last few layers.

References

1. Deep Learning Specialization at Coursera
2. PyTorch documentation
3. Intro to Deep Learning with PyTorch Free Course at Udacity
4. 10 dog breeds probably never hear of (Article)
5. How many dog breeds are there (Article)
6. Deep Neural Network for Dog Breeds Identification (Research Paper)
7. Rapid Object DetectRapid Object Detection Using a Boosted Cascade of Simple Features (Research Paper)