Machine Learning Engineer Nanodegree

Capstone Proposal - Dog Breed Classifier using CNN

Vedavyas Kamath
April 10th 2020

Proposal

Problem Statement

The problem statement here is to analyse any input image and check for the prescence of a dog in it, and when found return the breed name of the dog which resembles the one found in the image. This is an image classification problem (supervised learning) in which a model is built and then trained using images along with its labels. The model is made to learn the different images and remember their labels (dog breeds in our case), so that whenever a new image is passed, it is able to predict the breed for this new image based on what it has learned while training.

So having said this the model will take any image as the input and return the breed of the dog as output.

This problem has a practical potential solution and is measurable too, by all means. By tweaking the model a little and training it with an appropriate image dataset, the solution designed for this problem could also be used for identifying different fruits, cats, animals, vehicles etc.

Domain Background

This project belongs to the domain of Image classification which refers to a process in computer vision that can classify an image according to its visual content. For example, an image classification algorithm may be designed to tell if an image contains a human figure or not.

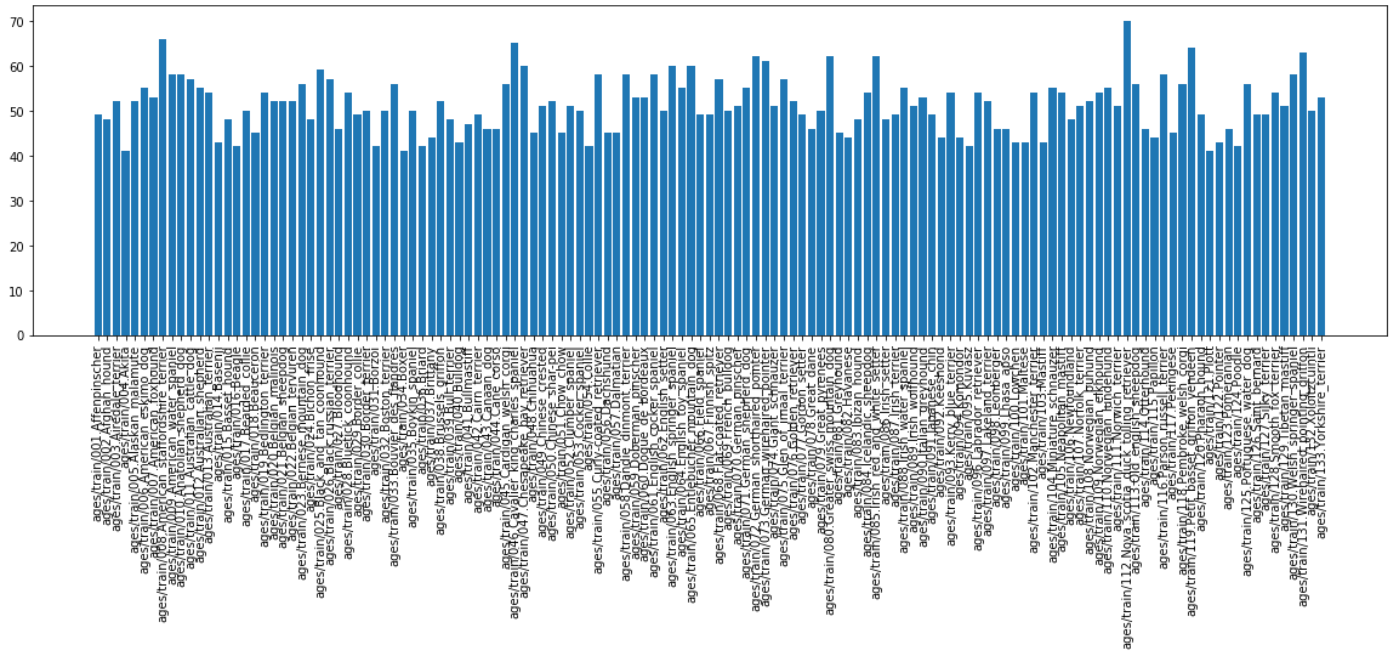
While detecting an object is trivial for humans, robust image classification is still a challenge in computer vision applications, which actually in fact has great scope in automation & implementing more robust feedback based Control Systems.

The main motivation for me behind choosing this project is mainly because of the fact that I find image classification very interesting and it has many practical applications that can be used to solve real life problems like crime, driving cars and finding tumors.

This project will help me explore the field of image classification, while giving me a solid idea and foundation of building an image classifier and preparing me to solve more complex real life problems.

Datasets and Inputs

As this project is to create a Dog Breed Image Classifier, a large set of 8351 dog images and 13233 human images was provided by Udacity. From the data folders, could see there are a total of 133 different dog breeds in the training data set. So as a first step I will split these images into training, validation and test sets to be used through out the project accordingly to first train the model, then validate and eventually test the performance of my model. To understand the data better and gather some descriptive facts/statistics, I have plotted a bar chart which shows the number of images included in the data for every distinct breed. Checked and could see on an average there are 50 images per breed which can be seen in the chart below:



This is how a sample image included in the dataset looks like:

As the data is in the form of images, the main data preparation task required would be to convert the jpg images into proper formats and sizes and then transform it into a tensor so that it can be fed to the model. I feel that some data augmentation would be required which involves re-sizing, center-cropping and random flipping and rotation which is usually done when working with image data.

Solution Statement

The algorithm developed as part of the project will accept any user supplied image as input but will identify and classify only the dog image. For any image uploaded having a human face, an estimate of the most resembling dog breed will be given and if neither is detected in the image, it provides error output. There are multiple ways of approaching this problem however I will choose to use convolutional neural network (CNN) to solve this problem and predict the breed for dog images.

Benchmark Model

A benchmark model for this project would be one making use of a CNN (Convolutional Neural Network) which would be trained using our training image data of dogs & humans.

First to detect whether an image has a dog's face or a human's face we need to build a model. We could use the OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images. And simillarly for detecting the prescence of dog face, we could make use of the pre-trained VGG16 model which is a very Deep Convolutional Networks for Large-Scale Image Recognition that has been trained on ImageNet, a very large & popular dataset used for image classification and other vision tasks.

Once we have detected a dog face in the image, the next step would be to predict the breed of the dog for which we would have to build another model and train it using the data we have at hand. This can be achieved by either building a new network right from scratch in which we would have to define each of the input, output and hidden layers all by ourself. Or by using Transfer Learning, in which we would use another already built model such as a resnet50 which is again trained on general images data (NOT dog specific data) and so replace the classifier to give output as one out of 133 classes.

By putting these different models together, I plan to build an app that will check the given image for dogs and if found, will display the breed its likely to be. If the image input has a human face, will display the dog breed to which the human's face resembles the most.

Evaluation Metrics

The models can be evaluated using an accuracy metric utilizing a test set. The human and dog detectors could be tested on 100 images of each to check the accuracy of the models. Further as stated in the dataset & inputs section, I would split the total image dataset into training, validation and testing sets to be able to train, validate and finally test my models. Usually depending on the dataset, accuracy may not be an adequate measure for a classification model. Like in a medical diagnostic model for example, if the occurrence of a positive result is 1%, then a model that predicts negative 100% of the time is 99% accurate but also a completely worthless model. Other metrics, such as precision, recall or F1 would be more useful for that type of problem.

However, after checking the dog breed dataset, the classes (breeds) seem to be relatively balanced, and hence a simple accuracy score should be sufficient. Simply the accuracy in this case can be viewed as:

Accuracy = $\frac{C}{T} \times 100$

where,
C -> Number of correctly classified images
T -> Total number of images to be classified

Project Design

I have planned to follow the below design/workflow to be able to solve this problem of dog breed classification in a step-by-step manner.

1. Data setup:

a. Download : This step is not applicable as the data is already present on Udacity workspace.
b. Loading : Will load this data by creating a custom loader to load training, validation and test sets.
c. Pre-processing: To convert the image file into a tensor that can be used by CNN for classification.

2. Human face detection:

a. OpenCV for Human Face Detection : Will use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images.

3. Dog's detection:

a. VGG16 : Use Pre-trained network VGG16 to detect presence of any dog in the image.

4. Dog's breed classifier:

1. Create a CNN to Classify Dog Breeds (from Scratch):

- Here I plan to define a CNN from scratch by deciding the different types and number of layers that will be used in my network to classify the breed.
- I then plan to make it feed forward by defining the overall path in which data will be propagted through the network to finally arrive to any one of the 133 possible dog breed classes.

2. Use transfer learning to use the already trained models (resnet50) as a base for our model:

- Pre-trained networks are efficient and well suitable to solve challenging problems because once trained, these models work as feature detectors even for images they werent trained on.
- Here I will use transfer learning to train a pre-trained network (resnet50) which is trained on ImageNet and readily available for use in torchvision.models so that it is able to classify our dog images. I have used the "resnet50" model

5. Write an algorithm, that does the following:

1. applies pre-processing to the input image,
2. checks for a dog's face and returns its breed if found
3. checks for human face, and if found returns the dog breed which human's face resembles to.
4. Handles the error in an interactive manner if neither dog or human face is found in the image.