Machine Learning Engineer Nanodegree Program

Capstone proposal

13 May 2020

By Kin Cheung

Background

This project is going to focus on dog breed recognitions using Convolutional Neural Networks (CNN). By the end of this project, there will be an application that users can apply their own images and the application will be able to recognize them as dogs or humans or neither. If they are recognized as dogs, this application will also classify their breeds. On the other hand, if recognized as humans, it will also try to resemble a dog breed to them. Just for fun.

This project is largely based on the Udacity dog classification project that can be found in the link below:

https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification#project-overview

Problem statement

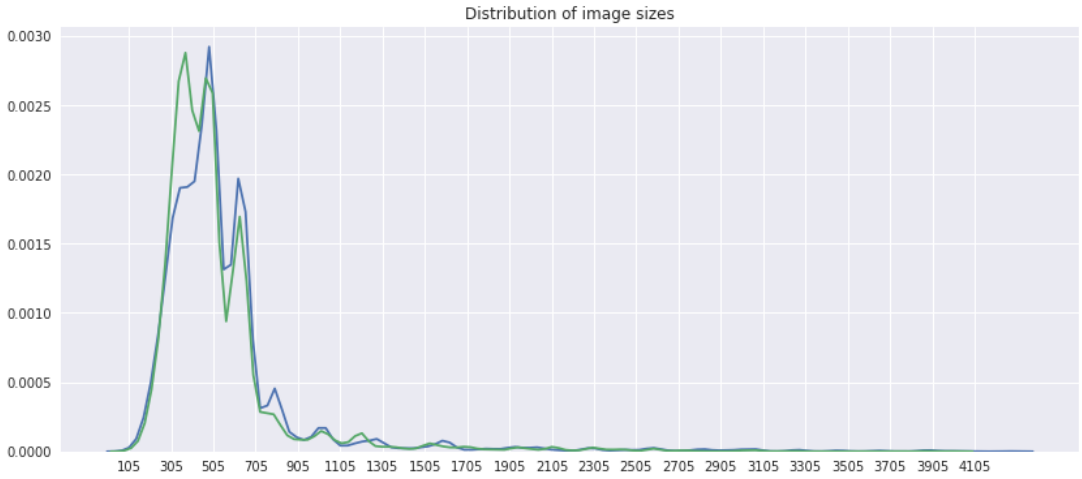
Image recognition has been a very popular field of research in computer vision. In recent years, many image recognitions models have been developed and significant performance improvements have been achieved over the years with promising results.

Although, these models are powerful and some of them even perform better than humans on average. Classifying dog breeds still appears to be very challenging considering many dog breeds do look very similar even to human eyes and to make things more complicated, a single dog breed might vary in colors and shapes.

Dataset and inputs

Datasets are provided by Udacity. They are in RGB JPEG format separated into different folders by dog breeds and further divided into training, validation and test sets. Arbitrary images will also be used at the end of the project to test out the final application. The images will include dogs, humans and neither of both.

From the graph below, we can see that the images provided are in different sizes. The green and blue lines represent width and height of the images in the datasets, respectively.



Also we can see that most of them images are between 320 to 480 H x W pixels. In order to maintain reasonably good performance and try not to consume too much memory and computer resources, I will resize the images to relatively smaller sizes around 32x32 to 56x56. They are around 1/10 of the original sizes.

If the training time and computer resources allow, I will try larger sizes. Otherwise I will try to keep the input sizes as small as possible.

Solution statement

There are a few functionalities that we will need to build the final application.

1. A human classifier
2. A dog classifier
3. A dog breed classifier

Human classifier

We will use an existing Haar feature-based human face classifier to detect human faces. It takes an image and can detect human faces in the image and also provide their x and y coordinates it.

Dog classifier

To detect dogs, we will use a pre-trained VGG-16 model. VGG-16 is one of the variations of VGG and it is the final best VGG network that contain 16 CONV/FC layers.

> **VGGNet**. The runner-up in ILSVRC 2014 was the network from Karen Simonyan and Andrew Zisserman that became known as the VGGNet. Its main contribution was in showing that the depth of the network is a critical component for good performance. Their final best network contains 16 CONV/FC layers and, appealingly, features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end. (https://cs231n.github.io/convolutional-networks/)

Dog breed classifier

For dog breed classifier, we will use a pre-trained ResNet model. ResNet is one of the best performed image recognition models out there that the public can make use of.

> **ResNet**. Residual Network developed by Kaiming He et al. was the winner of ILSVRC 2015. It features special *skip connections* and a heavy use of batch normalization.

> The architecture is also missing fully connected layers at the end of the network. The reader is also referred to Kaiming's presentation (video, slides), and some recent experiments that reproduce these networks in Torch. ResNets are currently by far state of the art Convolutional Neural Network models and are the default choice for using ConvNets in practice (as of May 10, 2016). (https://cs231n.github.io/convolutional-networks/)

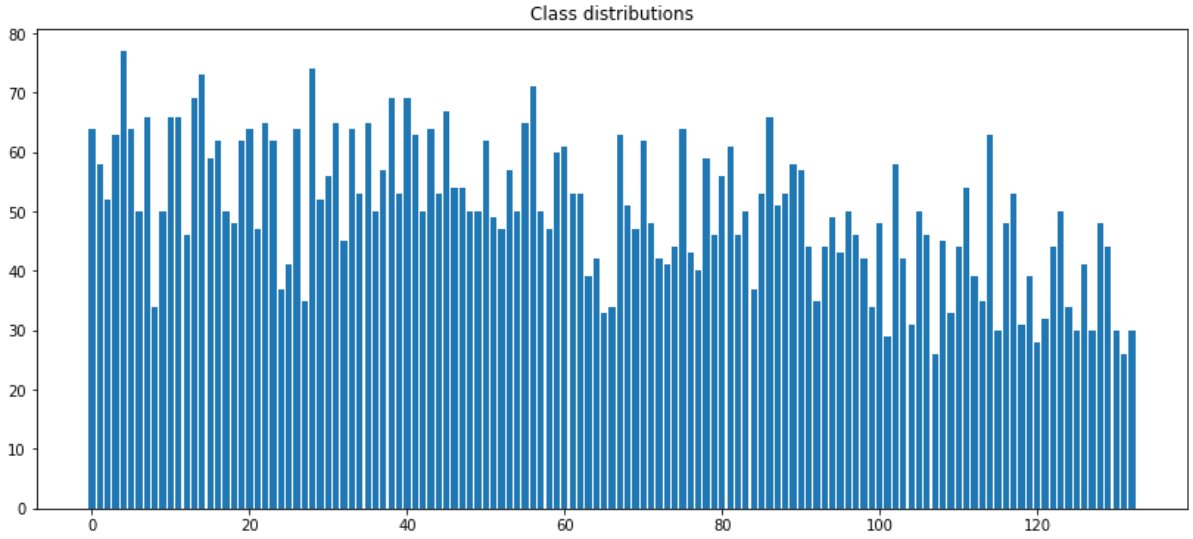
Benchmark model

I will use a simple CNN as a benchmark. This simple CNN model will be trained using the same set of training and validation data as the final solution. And then it will be tested against the same set of test data that I will also be using to test the final solution.

This simple CNN will take a relatively small feature set which is around 32x32 or 56x56 pixeled images as inputs with RGB channels. The architecture will consist of 3 to 4 CONV-ReLU-POOL layers to have reasonably good performance and will not take too much computer memory and time to train.

Evaluation metrics

There are 133 classes in total and the training dataset is slightly imbalanced as shown in the chart below.



Since the classes in the training dataset are imbalanced, a good option to handle class imbalances is to use log loss as an evaluation metric. In addition, the task we have here is a classification and we will use activation functions to model probabilities in the output layer of a CNN. It is an ideal case to use cross entropy as the loss function.

Project design

1. A human face detector using OpenCV implementation of Haar feature-based cascade classifiers.
2. A dog detector using a pre-trained VGG-16.
3. Build a dog breed classifier from scratch to see if it can achieve at least 10% accuracy

4. Use transfer learning to make use of a pre-trained ResNet model to make a dog breed classifier
5. Finally, build an application that takes user images as inputs and recognition results as outputs.