Machine Learning Engineer Nanodegree Capstone Project Proposal

Author: Ray Phan

Date: May 17th, 2020

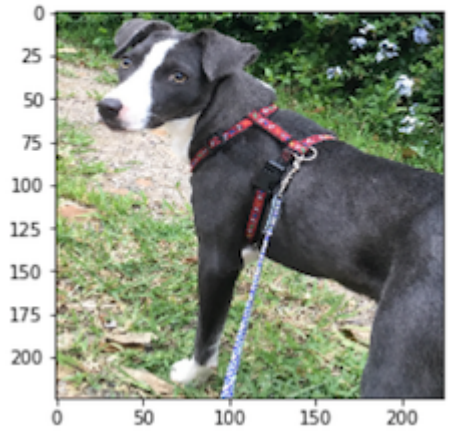
Introduction

For this capstone final project, I will lean on Udacity and simply work on one of the provided projects, which is classifying the dog breed given an image of a dog. The purpose of this project is to learn how to build a pipeline to process real-world, user-supplied images. Given an image of a dog, the proposed algorithm will identify an estimate of the canine's breed. If supplied an image of a human, the code will identify the resembling dog breed.

Along with exploring state-of-the-art CNN models for classification, important design decisions about the user experience for this application will be sought out. The end goal for this project is to understand the challenges involved in piecing together a series of models designed to perform various tasks in a data processing pipeline. Each model has its strengths and weaknesses, and engineering a real-world application often involves solving many problems without a perfect answer. Though this solution will obviously be imperfect, this will nonetheless create a fun user experience!

An example of what is to be achieved can be found below.

```
hello, dog!
your predicted breed is ...
American Staffordshire terrier
```



Domain Background

Classifying dog breeds just by image information is a very challenging problem. This problem is even hard for human beings without physically touching or assessing the dog in person. The end goal is to develop a model such that it infers what type of dog breed it is given a user-supplied image. After doing a cursory search through search engines, this topic is unfortunately not of academic interest but it has seen several surges in popularity on social media and on the Internet. For example, there is a renowned Kaggle competition that was held two years ago for this challenge. Other examples can be found in [2], [3], [4], [5]. Not only is this a very challenging problem, the original competition had 120 different classes of dog breeds. I found the discerning of

120 different classes to be intriguing, especially if there are certain nuances between the breeds that even humans would have trouble in discerning. I took on this challenge purely to see if machine learning and deep learning algorithms could automatically learn the right filters and weights to optimally decide this for us. Essentially, this is a relatively large multi-class classification problem where we fortunately can use supervised learning to solve this problem as we have the data available to us that is separated into the proper classes. Ultimately, the purpose of this is to explore the data available, design the right pre- and post-processing steps on the images and to decide which methodologies required to train a model and achieve a certain performance metric. This leads to our problem statement.

Problem Statement

The goal of this project is to build a machine learning - deep learning specifically - that can ultimately be used in a web application for example that can process real-world, user-supplied images. Specifically, this system will perform two tasks:

1. Given an image, if we detect that a dog is found the model will determine the canine's breed.
2. Given an image, if we detect that a human is found the model will specify that it's a human, then identify the closest dog breed that matches the human's appearance.

Datasets and Inputs

Fortunately this dataset is supplied by the Udacity Machine Learning Engineer Nanodegree Course. The dataset consists of already segregated training, validation and test groups. The input images must be colour images, though they can be of any resolution. Of course these must be images as the goal is to idenify what the canine breed would be in an image. In addition, the dataset consists of both dogs and humans.

For the dog images dataset, the images are of various resolutions but there are 8,351 images which are further segregated into 6,680 images for the training set, 836 images for the validation dataset and 835 images for the test dataset, approximately creating an 80%-10%-10% split. The images contain different backgrounds, and for each category the images are not balanced. Specifically, there are more images in one category than another. The number can vary between 4 and 8 images per category.

For the human images dataset, this contains 13233 total human images which are sorted by their corresponding names with each name in a directory. There are 5,750 unique names and each image is of size 250 x 250. The images also have different backgrounds and perspectives. The data is also not balanced as some directories have only one image when other directories have several images.

There are an abundance of dog and human images which should be enough for a data-heavy machine learning algorithm to properly bridge the gap between image to dog breed.

Solution Statement

Because there is an abundance of images, a Convolutional Neural Network (CNN) would be the most optimal (and obvious) choice here as neural networks are data-hungry, but because we have a large amount of data available, we would be able to create a set of filters, weights and biases to help us automatically solve this problem for us. A CNN is designed to examine an image and extract an optimal feature representation that would be suitable for a classification algorithm to optimally choose the right class. The classification algorithm would naturally be a multi-layer perceptron (MLP). Therefore, the solution first involves training a CNN with the dog image dataset. The choice of architecture for the CNN will require some experimentation and study but this

will naturally be part of the exploratory analysis of the problem. Secondly, we can use an already pre-trained solution to detect whether we can find a human face. We can use OpenCV's Haar Cascades which have been carefully engineered to find human faces. Thirdly, in order to distinguish between whether we see a human face or a dog face, we have to determine whether we can find a dog in the image first. Creating a dog breed classifier only tells us the type of dog, but not if there's an actual dog appearing in the image. Thankfully, we can use already pretrained networks that have been trained on the ImageNet database that do detect the existence of dogs but not to the granularity of what we expect from the proposed application. A pretrained network like VGG16 can work here. Therefore, the pipeline will be as follows:

1. Use OpenCV's Haar Cascades to determine if we can find a human. If yes, run the dog breed classifier to see what the closest dog breed is given the human image.
2. Use a pretrained VGG16 network to determine if we see an actual dog in the image. If yes, run the dog breed classifier to determine what the dog breed is.
3. If neither (1) or (2) execute, produce an error.

Benchmark Model

Naturally since the solution to this would be CNNs, a benchmark model would be to try and create a CNN architecture from scratch to see if this solution is viable. If we can create a CNN model "from scratch" which can achieve a test accuracy of say 10%, we can confirm that we would be able to train a network to achieve the desired result as this would be better than random guess, which is 1 / 133 or roughly less than 1%. Once we confirm that we can create a CNN architecture from scratch that achieves this desired test accuracy, we would naturally transition to using a deeper network and transfer learning such that the deeper network has already been pretrained on the ImageNet database, but we change the final fully-connected layer that handles the classification of the image by fine-tuning the weights to account for the dog image dataset that we provide to it. In this case, we hope to achieve a test accuracy of 60% or more.

Evaluation metrics

Because there is a relatively good distribution of data for each dog class, we can simply use accuracy which is the fraction of the dog breeds the algorithm classifies correctly in proportion to the entire dataset. We do this for the training, validation and test datasets. However, because there are only a small number of images per class we may also opt to look at the magnitude of the loss function we impose on the CNN when optimally finding the parameters that can most accurately classify the dog breeds. In this case, we use the standard multi-class cross entropy loss function to help us do this.

Project Design

Using the above, there are six steps we need to complete:

1. Download the dog dataset, have a cursory look at the dataset to see what we're looking at and propose the proper pre- and post-processing steps that best suit the CNN we desire to train.
2. Explore using OpenCV's face detection algorithm and convince ourselves that this works as expected.
3. Create the dog detector using the pretrained VGG16 network with ImageNet weights.
4. Create the CNN "from scratch" to assure ourselves that this problem can be solved with the dog dataset.
5. Train a CNN through transfer learning using the same dog dataset. Here I will choose ResNet50 [6] as it's relatively deep but the parameters can fit comfortably on a K80 GPU which I will be using to train the network.

6. Create the application as specified in the problem statement that combines the human detector and dog detector.

The deep learning library we will be using for this task will by PyTorch.

References

1. Kaggle Competition - Dog Breed Classification
2. How to easily build a dog breed image classification model - James Le - Medium
3. Dog Breed Classification using CNNS - Deniz Doruk Nuhoglu - Towards Data Science
4. Dog Breed Classification - mc.ai
5. Dog Breed Classification using CNN and Transfer Learning - Li Liping - Gitconnected
6. PyTorch
7. Deep Residual Learning for Image Recognition - Kaiming He et al. - arXiv