Capstone Proposal

Shrey Samdani

December 2019

1 Abstract

This project aims to create a deep learning model that accurately classifies the breed of a dog given the picture of the dog. The solution will test a convolutional neural network made from scratch and a transfer learned network to predict the dog breed. The goal is to get an accurate model that could be implemented into a phone app or web app.

2 Domain Background

This project falls under the domain of deep learning, a category of machine learning that uses methods based on neural networks. Deep learning models can be used for natural language processing, speech regognition, computer vision, etc. This project explores image classification through deep learning. Generally, image classification is approached with a convolutional neural network, a model type that will be used in this project.

3 Problem Statement

This project aims to make the first steps towards developing an algorithm that could be used as part of a mobile or web app. At the end of this project, the code will take an image as input. If a dog is detected in the image, it will provide an estimate of the dog's breed. If a human is detected, it will provide an estimate of the dog breed that is most resembling.

4 Datasets and Inputs

The datasets for this project are provided by Udacity. The dataset with images of humans contains 13233 total images of humans. The other dataset contains 8351 dog images along with their breed. There is some imbalance in our dataset, as there are more human photos than dog photos, but this is not an issue as we will only be training on the dog photos. Within the dog photos, there is also some imbalance between the 133 different classes, which may pose an issue

during training. The images are also in different shapes and sizes, with varying locations of where in the image the dog actually stands.

5 Solution Statement

We first plan to identify if an image corresponds to a human. OpenCV provides Haar feature-based cascade classifiers to help do so. Next, we work to identify if an image corresponds to a dog. We can do so by obtaining a pretrained VGG 16 model provided by PyTorch and running our image through the model. Classifying the actual breed of the dog is trickier; we will attempt to do so using two different methods. The first method consists of building a CNN from scratch that will identify the breed. The second will use a well known neural network transfer learned into our dataset. Finally, we put together the best performing models into a function that takes in an image and attains the functionality desired by the problem statement.

6 Benchmark Model

For the CNN created from scratch, we hope to attain an accuracy of at least 10%. Given that a random guess has a 1/133 probability of being correct, 10% accuracy would be a good benchmark to show that our model does better than random guessing. For the transfer learned model, we hope to attain an accuracy of at least 60%. Since this model is based on a more well tested neural net (90%+ accuracy on other image sets), we should hope to achieve at least 60% (lower due to the high number of classes we have).

7 Evaluation Metrics

There are two evaluation metrics we will use to evaluate the performance of our model. The first is the prediction accuracy of our model. This is a relevant metric as it is easy to visualize and interpret - we will have a good understanding of how our model is doing. The second is the cross-entropy loss (log loss) of our model. This metric does not provide an obvious understanding of our performance, but rather a relative understanding. It is a metric we can use to compare across models to understand which model is performing better.

8 Project Design

The solution will be developed using python. We will use OpenCV to obtain the Haar feature-based classifier. Since this works on a raw image, we do not have to do any image pre-processing. For the classification of the dog breeds, we will have to first split our data into train, test, and validation sets. To train our model, we will have to augment our images to add transformations (rotations,

flips, brightness, random cropping, etc.) Once we preprocess our data, we will build two neural nets: one will be a CNN built from scratch and the other will be a transfer learned neural net. We will use PyTorch to accomplish the transformations and the model building. Once we tweak the parameters of our model to achieve the expected accuracy, we will put everything together to create a classifier of an image given just the filename (final product). The project will be executed in a python Jupyter notebook.