Machine Learning Engineer Nanodegree

Capstone Proposal

Sitansh Rajput

January 10th, 2019

Proposal

(approx. 2-3 pages)

Domain Background

(approx. 1-2 paragraphs)

Increasing accessibility and education is an important goal within the STEM fields, and the applications for a machine learning powered ASL alphabet detector are enorous. Not only could such a model be fitted to other languages, but could be fitted to whole words and symbols. This could lead to educational services and apps focused on ASL to any variety of users.

I tookon this project because it seemed like a very interesting, and applicable topic with pragmatic, real-world uses.

https://pdfs.semanticscholar.org/eb2d/466f0c44509dde2893460473e1d6725e9e92.pdf

Problem Statement

(approx. 1 paragraph)

Each image is a grayscale (0-255) with a size of 28x28 and each set is contained within an independent .csv file. With a training set of around 20 thousand images, and a test set of around 5 thousand images, we need an efficient solution to detect the letter formed by a certain hand shape (as is done by speakers of ASL). The problem here is that each image needs to be recognized by the algorithm to a letter. Using a metric of accuracy, we can gauge how well the algorithm fits the training data and how well it can extrapolate to the testing set.

Datasets and Inputs

(approx. 2-3 paragraphs)

The dataset was provided by Kaggle as an ASL drop-in for the classic MNIST dataset. It contains a training and testing .csv. The training .csv associates a hand sign with its respective label. Each row within the .csv contains a unique image. The end result would be to create a model with 95%+ greater accuracy. The algorithm will be tested on the test.csv set provided for this express purpose. This dataset relates directly to the problem because it provides a testing and training set to use against the machine learning algorithm.

After doing some brief analysis on the data, the classes (signs) are balanced on both the training and test set so I will not need to preprocess the data during split to balance them out.

https://www.kaggle.com/datamunge/sign-language-mnist

Solution Statement

(approx. 1 paragraph)

In this section, clearly describe a solution to the problem. The solution should be applicable to the project domain and appropriate for the dataset(s) or input(s) given. Additionally, describe the solution thoroughly such that it is clear that the solution is quantifiable (the solution can be expressed in mathematical or logical terms) , measurable (the solution can be measured by some metric and clearly observed), and replicable (the solution can be reproduced and occurs more than once).

For this problem, I will be using a CNN similar to the dog labeling project that Udacity provides.

Benchmark Model

(approximately 1-2 paragraphs)

The benchmark model will be a publicly available kernel hosted on Kaggle. It provides the basis for a very simple CNN model that attains an ~85% accuracy, the goal will ultimately be to improve on this model and achieve an accuracy of over 95%.

https://www.kaggle.com/ranjeetjain3/deep-learning-using-sign-langugage

Evaluation Metrics

(approx. 1-2 paragraphs)

For this project, I will simply be using an evaluation metric of accuracy. It is a very simple metric, and relatively good gauge to test the performance of the model, since the classes are balanced accuracy will not be negatively impaired for its evaluation.

Project Design

(approx. 1 page)

The project will utilize a PlaidML backend for Keras (this can be changed by a comment of a line). This will allow me to compile the model using my AMD GPU, as TensorFlow currently only supports CUDA (Nvidia) cards.

I will then possibly augment the images to prevent overfitting depending on the initial results of the model by changing the blur, bloom, and rotation of the images. Rotation specifically, I will have to be careful with, because certain signs once rotated will have actually be another and negatively affect the performance of the model.

From the benchmark model, it seems as if it suffers from overfitting, because the accuracy on the training and validation set reaches 1.0 however, it does not translate to the testing set. So I will be adding more Dropout layers and increasing the rate of the current one.

Activation layers also seem to be important here, the benchmark uses Relu, but I'll also be looking into ELU and PRelu and seeing how they affect the performance of the modle.

There are pre-trained models for image classification such as Inception V3 and VGG16 that I might try to incorporate, this will probably give a big boost to the performance of the model, but I will implement this later on if my initital attempts to not give good results.