Machine Learning Engineer Nanodegree

Capstone Proposal

Darryl Ma
December 31st, 2017

Proposal

Domain Background

American Sign Language (ASL) is the predominant sign language used amongst the deaf communities in North America. It was created in the early 19th century at the American School for the Deaf in Hartford, Connecticut. For years, due to the stigmas and beliefs in the superiority of the oral language, sign language has been largely constrained to the deaf community with a few family members embracing the language in order to communicate with their loved ones. However, with the advent of machine learning and visual classification, this barrier can easily be overcome by introducing apps within our mobile devices to translate visual cues into spoken words.

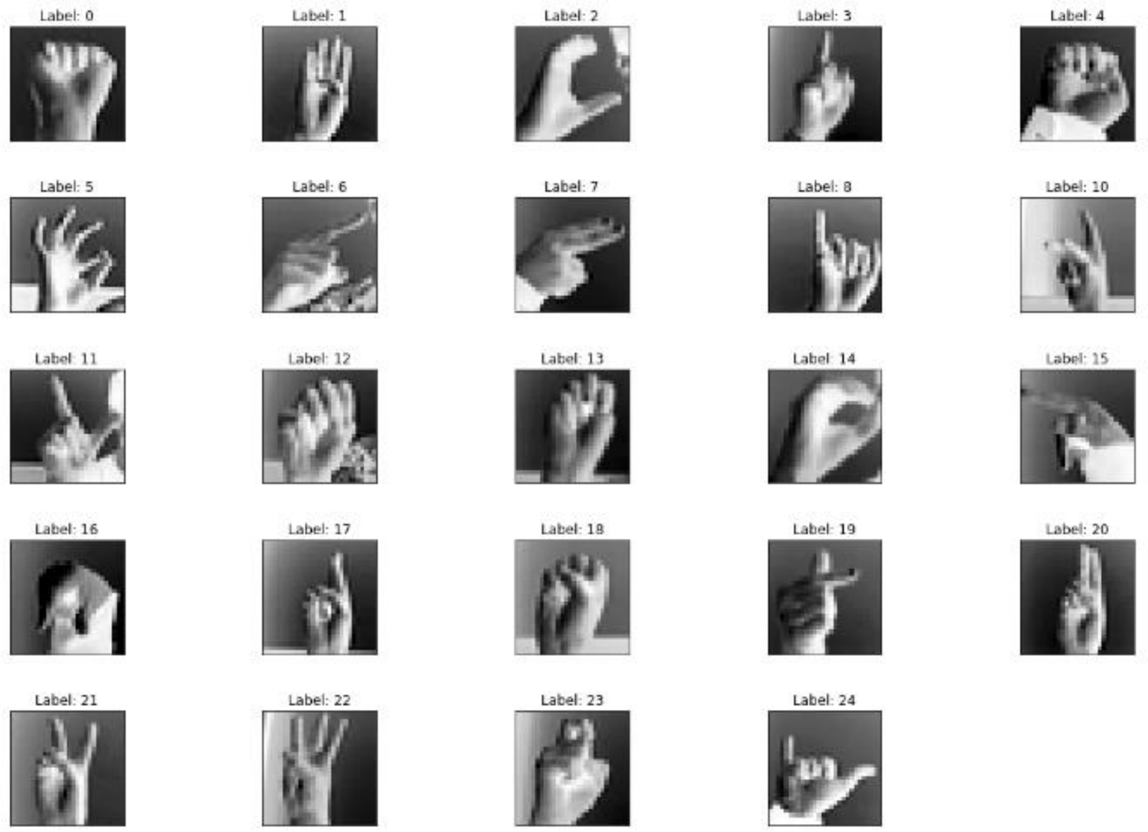
My grandfather was deaf and as such communication between family members, who did not learn sign language, was limited. Because of this handicap, he was isolated from a lot of family conversations and discussions. Enabling him to communicate with the rest of the family would have drastically altered his relationship with them and ultimately enhanced his quality of life. Therefore, I don’t see this endeavor as just a way to solve a communication problem rather it is more about re-integrating a typically estranged community of people (i.e. the deaf community) back into our society.

Problem Statement

The ASL lexicon consists of over 50,000 signs and there are there are many sign language dialects (e.g. French sign language, Ecuadorian sign language, etc.) used around the world, however, for this project, we will be focusing strictly on ASL and the static handshapes used to communicate the English alphabet. Furthermore, because the letters “J” and “Z” require motion, these two particular alphabets have been removed from this classification project. My goal is to use machine learning, specifically

Convoluted Neural Networks (CNN), to achieve 80% accuracy in identifying 24 of the 26 English alphabets from a given dataset of sign language cues.

Below are samples of the 24 handshapes I will be attempting to classify:



Datasets and Inputs

The training and testing datasets for this project were obtained from Kaggle.com (https://www.kaggle.com/datamunge/sign-language-mnist).

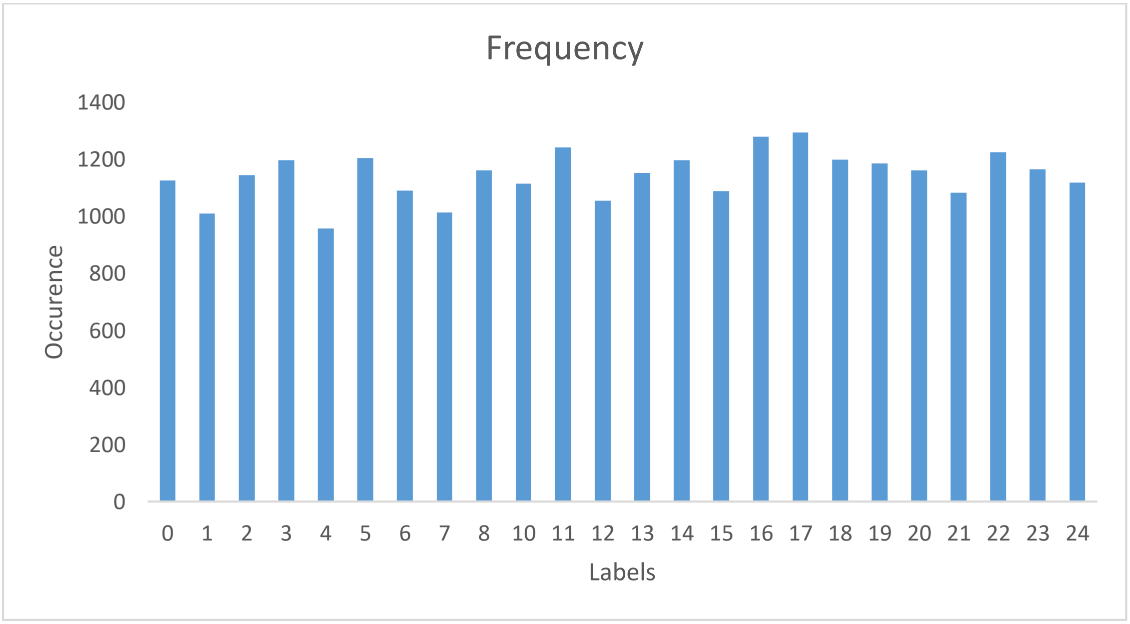
The training dataset consists of 27,455 labelled samples, each a 28x28 monochrome image of a handshape.

The testing dataset consists of 7,172 labelled samples, each a 28x28 monochrome image of a handshape.

The following table below is a mapping of the labels to their corresponding alphabets. Do note that label 9 is missing as it corresponds to the alphabet, J, which, as highlighted before, is not included in this classification exercise as it is a hand gesture that requires motion:

Label	Alphabet		Label	Alphabet
0	A		13	N
1	B		14	O
2	C		15	P
3	D		16	Q
4	E		17	R
5	F		18	S
6	G		19	T
7	H		20	U
8	I		21	V
10	K		22	W
11	L		23	X
12	M		24	Y

The plot below shows the frequency of the different labels in the training dataset, which verifies that no particular handshape is biased, all handshapes are relatively equally represented:



Solution Statement

As stated above, I will be using Convoluted Neural Networks to solve this classification problem. To start off, I will attempt to build my own model architecture from scratch. Later, I will utilize transfer learning on pre-trained models such as VGG16

VGG19, Resnet50, InceptionV3, or Xception to see if I can improve accuracy. At the end, the goal is to achieve at least 80% accuracy in identifying the 24 classes of handshapes.

Benchmark Model

As there are 24 classes, the probability of guessing the correct label randomly is 1 out of 24 or 4.2% chance. Anything that achieves results above this percentage is an indication that there is more than just chance at work.

For the benchmark model, I will be using a vanilla convolution neural network (CNN), which uses 3 convolution layers, max pooling layers between each convolution layer to reduce dimensionality, a flatten layer, a dropout layer to reduce overfitting and lastly, a fully connected layer with softmax activation to obtain probabilities for each handshape prediction. See below for vanilla model architecture:

```
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)           (None, 28, 28, 16)        80        
_________________________________________________________________
max_pooling2d_1 (MaxPooling (None, 14, 14, 16)        0         
_________________________________________________________________
conv2d_2 (Conv2D)           (None, 14, 14, 32)        2080      
_________________________________________________________________
max_pooling2d_2 (MaxPooling (None, 7, 7, 32)          0         
_________________________________________________________________
conv2d_3 (Conv2D)           (None, 7, 7, 64)          8256      
_________________________________________________________________
max_pooling2d_3 (MaxPooling (None, 3, 3, 64)          0         
_________________________________________________________________
flatten_4 (Flatten)          (None, 576)               0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 576)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 25)                14425     
=================================================================
Total params: 24,841.0
Trainable params: 24,841.0
Non-trainable params: 0.0
_________________________________________________________________
```

The accuracy achieved by the benchmark model above will be used as the standard to evaluate the effectiveness of other supervised learning methods used solve this classification project.

As a secondary benchmark, I will be using Luis A. Estrada Jiménez and Marco E. Benalcázar’s paper on “Gesture Recognition and Machine Learning Applied to

Sign Language Translation", which they presented at the VII Latin American Congress of Biomedical Engineering conference in October 2016 (https://link.springer.com/chapter/10.1007/978-981-10-4086-3_59). They claimed to achieve classification accuracy of 91.55% on 61 gestures from the Ecuadorian sign language (30 letters, 10 numbers, and 21 expressions). Their solution utilized k-nearest neighbors, decision trees, and the dynamic time warping algorithms to achieve their results and unlike this project, instead of image processing, they used flex, contact, and inertial sensors mounted on a polyester-nylon glove to build their input features. Given that I don't have the investment to build my own glove, we will see if I can achieve comparable results given the limited resources available at my disposal.

Evaluation Metrics

As stated before, I will be using classification accuracy on the test dataset as the main evaluation metric. Similar to previous Udacity projects, we will using the code below:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('\n', 'Test accuracy:', score[1])
```

Along with the accuracy metric, I will also be plotting a confusion matrix to see if there are any particular handshapes that tend to get misclassified more often than others. Theoretically, this should help in identifying handshapes which require more training samples. The confusion matrix should look something like this:

Predicted Vs True Label

	Predicted Labels					
True Labels	■	0	1	2	3	4
	0	50	0	4	2	0
	1	1	40	0	0	3
	2	0	2	45	1	0
	3	1	0	0	60	0
	4	2	4	0	0	30

Project Design

The project flow will mirror the flow of the dog_app project. The major sections of the project are outlined below:

1. Import datasets and explore data
    a. Plot set of images to get a sense of the data

2. Pre-process data
    a. Rescale images by dividing each pixel by 255
    b. One-hot encode testing and training labels
    c. Split training dataset into training and validation sets
    d. Reshape image vectors into 4D matrices/tensors that can be fit into the model
3. Build, train and test a vanilla Convolution Neural Network as described in the "Benchmark Model" section
4. Build, train and test on other supervised learning models:
    a. CNN model utilizing transfer learning
    b. Choose either Naïve Bayes, Support Vector Machines, or AdaBoost classifier
5. Test other images on most accurate model
6. Discuss findings and limitations
7. Propose improvements

As of now, this is the proposed flow of the project, which is subject to change as soon as I start to actually implement the project.

References

[1] https://www.kaggle.com/datamunge/sign-language-mnist "Kaggle.com"

[2] https://link.springer.com/chapter/10.1007/978-981-10-4086-3_59 "Gesture Recognition and Machine Learning Applied to Sign Language Translation"

[3] https://en.wikipedia.org/wiki/American_Sign_Language "Wikipedia – American Sign Language"