MAIS 202 Deliverable 2: ASL Alphabet Classifier

Cheng Lin

1. Problem Statement

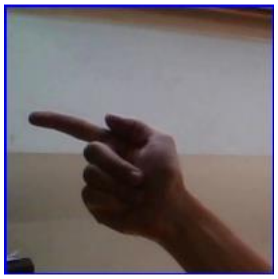
I want to build a model that classifies American Sign Language letters inputted by a user. I plan on integrating my model with a web app that interacts with the user via video camera.

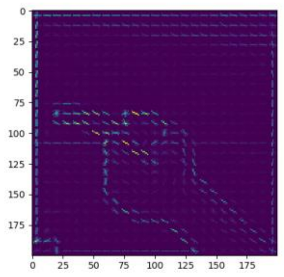
2. Data Preprocessing

As stated in deliverable 1, the dataset consists of American Sign Language images, each 200 x 200 RGB pixels. In total, there are 3000 labelled images per letter, resulting in 78 000 images in total. However, because the full dataset exceeded my computer memory and took too long to train, I limited training to 2 600 photos. Upon reading similar image classification posts, such as this one, which classified non-ASL signing and ASL signing, as well as advice from John (MAIS Exec), I decided to pre-process my data using Histogram of Oriented Gradients (HOG). I decided to deviate from using PCA and convolutional neural networks because the image classification problem is not complex enough.

The 26 classes involved in ASL sign language all look relatively similar: for example, the visual differences between an A and a B are not comparable to the visual differences between a dog and a boat. As a result, the model doesn't need to learn all the intricacies of each class via a neural network, it only needs to learn the silhouettes of each hand sign. This is why HOG is a much simpler, but still effective, alternative.

I began my pre-processing by reading in each image with the Pillow library, converting it to gray scale, and calculating the HOG with scikit-learn, as per this article. This converted each image from a 3-dimensional array representation of an image, to a 1-dimensional vector of gradients. Visualizations of the pre-processing results can be seen in Fig. 1.





(a) (b)

Fig. 1: (a) original image of G6.jpg from test set
(b) visualization of G6's HOG

MAIS 202 Deliverable 2: ASL Alphabet Classifier

Cheng Lin

To convert the labels into values an SVM could interpret, I used scikit-learn's LabelEncoder to transform them into categorical numerical values. Finally, I used scikit-learn to perform a 80-20 split on the pre-processed data and obtain my training and validation datasets.

3. Machine Learning Model

I used scikit-learn's multi-classification SVM package to train the datasets. I assumed that when all the images are properly centered (as they are in the training data), the gradient vectors of the ASL hand sign's outline are linearly separable. I based my assumption on the fact that the silhouettes for the ASL letters are all quite distinct from each other.

4. Preliminary results

The trained model has an accuracy of 98%. The confusion matrix can be found in Fig 2.

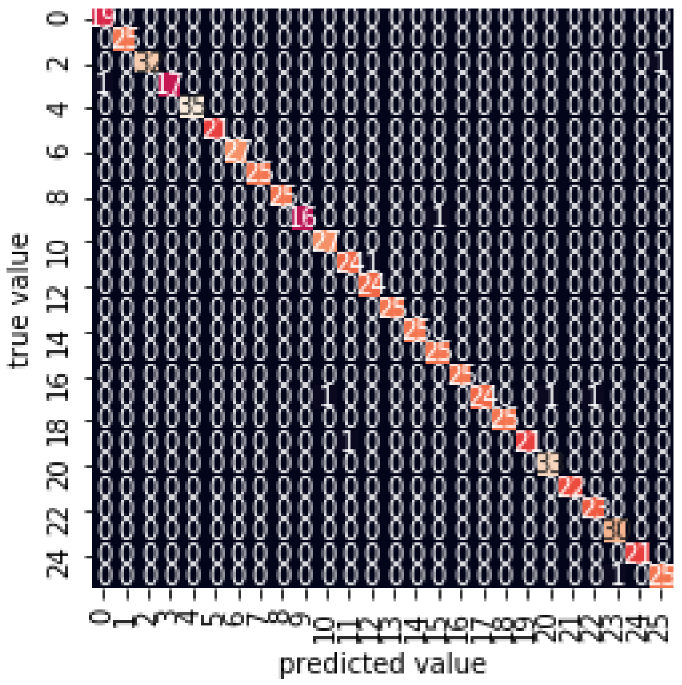


Fig. 2: confusion matrix for preliminary model

While the model isn't perfect, it manages to classify almost all letters correctly. However, because the dataset I trained and tested on is limited, the model may be biased towards the data its seen and not reflect all possible inputs. I need to train the model on more images to make it more versatile.

MAIS 202 Deliverable 2: ASL Alphabet Classifier

Cheng Lin

5. Next Steps

I will continue training the model on more images. I want to see if model performance decreases when I use more of the corpus, or if it stays relatively the same.

While the SVM model may not capture enough of the hand sign details when I add more data, so far it is simple to train and predicts fairly well. If the model performs worse when I increase the corpus, I will perform hyperparameter tuning for the C (penalty parameter). Once I obtain an optimized model, I will begin to integrate it with my web app. In doing the latter, I plan on using a sliding window approach to locate the ASL hand sign within the image received by the webcam.