American Sign Language

Machine Learning

IST 718

Marc Bragdon

Joel Bravo

Stephanie Gardner

Vince Nazzaro

This report examines machine learning methods to interpret American Sign Language (ASL) alphabet letters. The dataset contains vectorized images of ASL letters in addition to images of the ASL letters. Four models were explored including XGBoost, Naïve Bayes, Random Forest and CNN. Extremely high training and testing accuracy was seen in three models, achieving over 98% accuracy. It is recommend to continue to train ASL alphabet letters using the Random Forest model as the model is exceptionally good with images.

**Specification**

**Data**

There were two different datasets leveraged; the first data set was composed of 78,000 color images of individual’s hands signing the letters A through Z, including the hand sign for space. The second dataset contained two CSV files, one for training and the second for testing. In the training CSV file there were 27,455 labeled color vectors of an image. The test CSV dataset contained 7,172 labeled vectors as well.

**Problem**

The problem identified after examining the dataset is: Can machine learning accurately interpret American Sign Letter letters using computer vision?

**Hypotheses**

The application of this project could be used to help provide fluid translations of ASL to non ASL-readers in order to facilitate easier communication

**Observation**

The initial dataset consisted of RGB images and depth data of ASL alphabet letters that correspond to the 24 letters of the English alphabet as a sample is seen in Figure 1 below. Letters J and Z were excluded due to the motion required in signing these letters. Given the large size of the data files (the initial compressed files totaled 2GB in size), vectorization and storage solutions had to be addressed by the team.



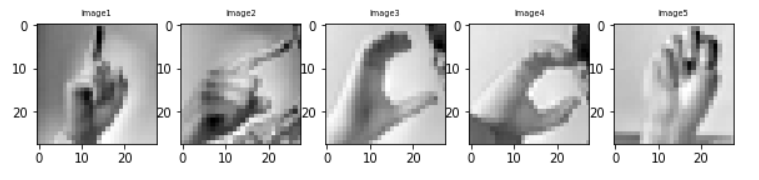
Figure 1

Testing on the vectorized images would result in a dataset that was still too large for some machines. Running a CNN on all images, for example, results in a matrix 4.7GB in size. As an alternative, a second dataset was leveraged that had already vectorized the ASL letters and was much more manageable in size of the machines. The images in question had relatively low pixel noise and clean backgrounds.

Many preprocessing steps were required including the following the steps:

* + Converting the dataset images from color to black and white for more manageable analysis
  + Normalizing the dataset
  + Standardizing size to 28x28 pixels
  + Reshaping the dataset

The final requirement was removing the labels from the testing dataset to avoid erroneous results.  Once all of the preprocessing steps were completed, the analysis models could be explored. Examples of the converted images are seen below in Figure 2.



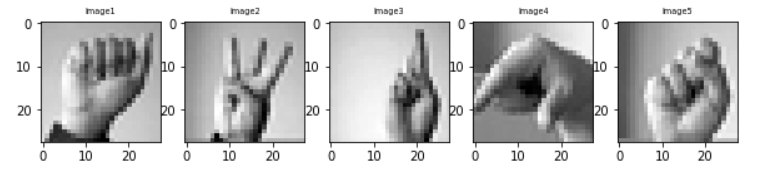


Figure 2

**Analysis**

Four models were applied to the dataset to achieve the highest accuracy in predicting the ASL characters. The following models were explored:

* + XGBoost
  + Convolutional Neural Network (CNN)
  + Random Forest
  + Naïve Bayes

Convolutional Neural Networks (CNN) are known for their high accuracy in image classification. Random Forest is known to be a strong classifier algorithm with high accuracy even if it is more general purpose. An XGBoost algorithm is also known to be a very strong classical machine learning model and finally, a Naive Bayes classifier which was used as a comparison for other models to beat.

**XGBoost**

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. XGBoost was engineered to use memory and compute efficiently. It can run in parallel and distributed computing. The model was heavily parameterized the first attempt, whereas the second attempt used more default values and was more accurate. The model was run on the CSV version of the data using the following XGBoost parameters:

• eta=.3

• max\_depth=5

• objective = multi:soft

Using the parameters, XGboost was instructed to iterate 600 times using new weights and analyzing the results. Additionally, to preclude additional compute, a stop value of 50 was used to automatically stop the execution if there was no improvement of the model. Leveraging Google Colab and changing the environment to GPU, the training model ran for 39.86 minutes and stopped at the 156th iteration when there was no improvement over the 106th model. The training model achieved 99% accuracy. Executing the test data against the model resulted in a compute time of 0.981 seconds with an accuracy of 98.3%. Figure 3 represents the best model XGBoost tree for the best-identified model (106th).

A close up of a map

Description automatically generated

Figure 3

**CNN**

The CNN was run on a collection of 78,000 jpeg images. The images were resized from 120x120 to 28x28 so they would fit in the memory constraints on the GPU. The images required lazy loading due to memory storage size resulting in higher compute performance, approximately two hours for both the training and testing models.

The testing accuracy was far higher far sooner than the training accuracy with a lot of different images in the subset. There were two convolutional layers, two pooling layers and a dense layer with 1,024 neurons. Others attempts with more convolutional layers did not increase accuracy. After a 100 epoch run with 5,000 steps per epoch training and 500 steps per epoch, testing the model had an accuracy of 99.9997%. A 50% single dropout layer was employed to reduce overfit. The model accuracy and model loss can be seen below in Figures 4 and 5 respectively.

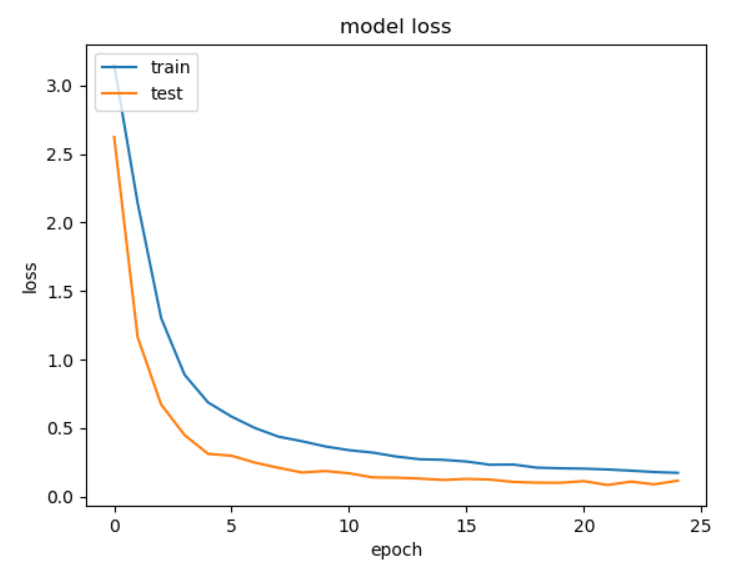


Figure 4

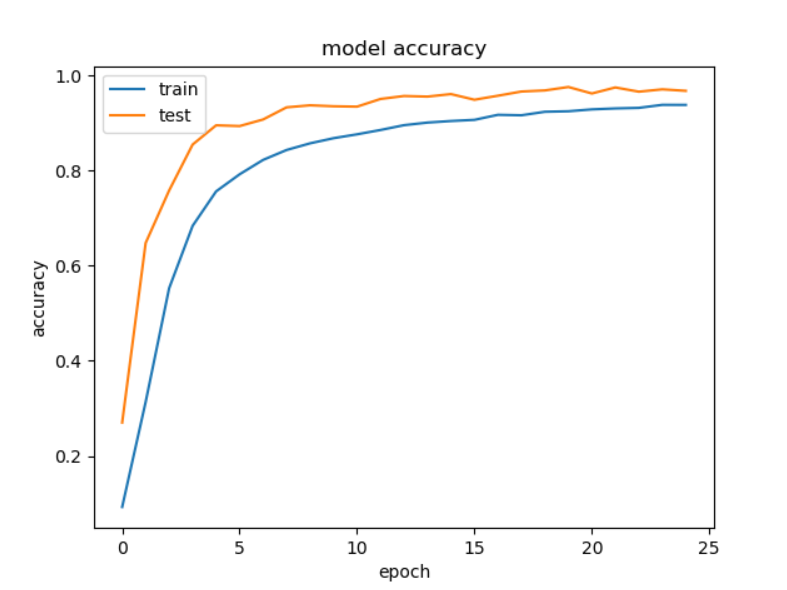


Figure 5

**Random Forest**

The next model explored was Random Forest. The initial assumption was that the accuracy would be high since Random Forests are known for their excellent classification and resistance to overfit. A high accuracy was also seen previously in the fashion MNIST dataset exploration. The sklearn implementation was used of the Random Forest classifier on the CSV version of the data with the n\_estimators set to 50.

The model completed in just seconds, with the training model at 17 seconds with 100% accuracy. The testing model completed at a mere 0.341 seconds. Outputting the testing accuracy would result in an accuracy of 99.71% This proved to be far higher than initially anticipated.

**Naïve Bayes**

A Naïve Bayes model was also explored as a comparison to the other models. A low accuracy was assumed with a low compute performance. For this model, a Gaussian (normal distribution) was used. Both the training and testing models ran extremely fast, at 1.179 seconds and 2.19 seconds respectively. However, the training model only reached a 46% accuracy rate and the testing model only reached a 39% accuracy.

Figure 6 below shows the precision by each letter where it can be noted that is very easy to classify letters B (2) and H(7), however the majority have low precision.

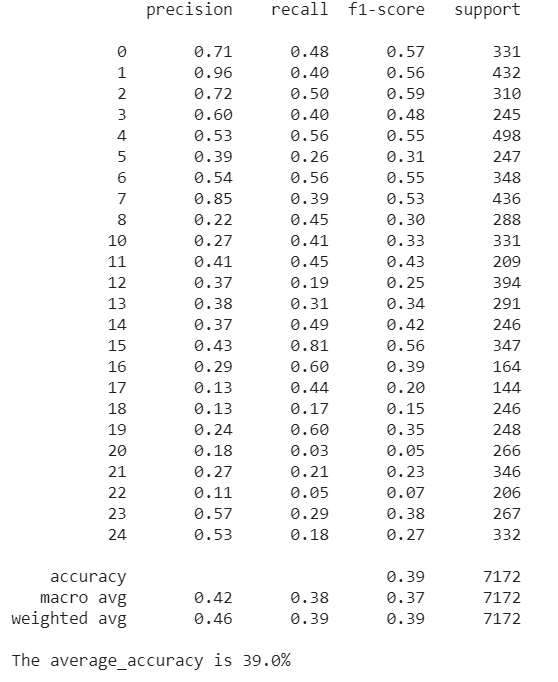


Figure 6

The collective results of all models can be seen below in Table 1. Whereas it was assumed the CNN would be the best model, given how purpose driven an algorithm it is, the Random Forest model was a surprisingly strong and accurate classifier for this data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Training Accuracy** | **Training Compute performance** | **Testing Accuracy** | **Testing Compute performance** |
| XGBoost | 99% | 39.8 minutes | 98.3% | 0.951 seconds |
| Naïve Bayes | 46% | 1.179 seconds | 39% | 2.19  seconds |
| Random Forest | 100% | 17 seconds | 99.71% | 0.341 seconds |
| CNN | 99.9997% | 2 hours | 99.9997% | 2 hours |

Table 1

**Recommendation**

After reviewing the performance of the three models with the highest accuracy, the model that performs the best, considering accuracy and compute performance, is Random Forest. The next steps would be for individuals to sign the ASL alphabet letters to continue to train and test the model.

Since Random Forest is exceptionally accurate with classification of images, there is an opportunity to explore real time classification of the ASL alphabet letters. A real time classification model was developed and was successfully run. It can continue to be trained and also explore background noise as this would be a factor during real time classification.

Finally, the ultimate goal would be to expand into the audio translation of ASL alphabet letters. This would be very beneficial in communication with the hearing impaired.

**References**

<https://www.kaggle.com/mrgeislinger/asl-rgb-depth-fingerspelling-spelling-it-out>

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

[https://colab.research.google.com](https://colab.research.google.com/)

<https://wecapable.com/tools/text-to-sign-language-converter/>