DS 4400: Machine Learning and Data Mining I

Spring 2022

Project Report

Project Title: Translating the ASL Alphabet through Computer Vision

TA: Jake Horban

Team Members: Tejas Sathyamurthi, Justin Lau

Link to code:
https://colab.research.google.com/drive/14I0r-CmC3EEP2CRqr0VwxPPQqlrsDZ9K?usp=sharing

Link to video: https://youtu.be/2-uW6NEJjkw

Problem Description:

The machine learning problem that this project seeks to solve is translating ASL signs into the English alphabet. In a world where virtual interactions are more prevalent, translators may not always be available to help deaf people communicate with non-ASL speakers. Furthermore, despite new technologies allowing for various treatments and methods for helping deaf people communicate, these tools are often very expensive and may not be a reliable resource for those coming from low-income families. According to the BBC, “just 5% of adults who could benefit from cochlear implants receive one.” Although cochlear implants would remove the need for ASL for deaf people completely, we cannot assume that every deaf speaker will receive one. On the other hand, using machine learning, we would be able to translate ASL signs into English words in real-time. This is very important because deaf people will have more autonomy if machine learning algorithms were implemented in society to translate their signs. Actions like ordering food and accessing medical care will become more seamless and accessible for signers. This methodology would also be more cost-effective and accessible for users than cochlear implants. This technology would also have a widespread impact; currently ASL is the third most commonly used language in the United States, after English and Spanish, used by over a half-million people.

In this project, we seek to classify images of hand signals within a set of classes; classes that represent the 26 letters in the English alphabet, or other representations, such as “space”, “nothing”, or “delete”. Therefore, this problem is a **multiclass classification problem**, since we are trying to classify hand signals into 1 of 29 classes.

As discussed above, this is an important problem to tackle and identify because it can help over a half-million people in the USA, and modern tools and technologies are often very expensive and may not be a reliable resource for those coming from low-income families. Machine learning would allow for real-time translation, providing deaf people with more autonomy, while also being more cost-effective and accessible for users.

References:

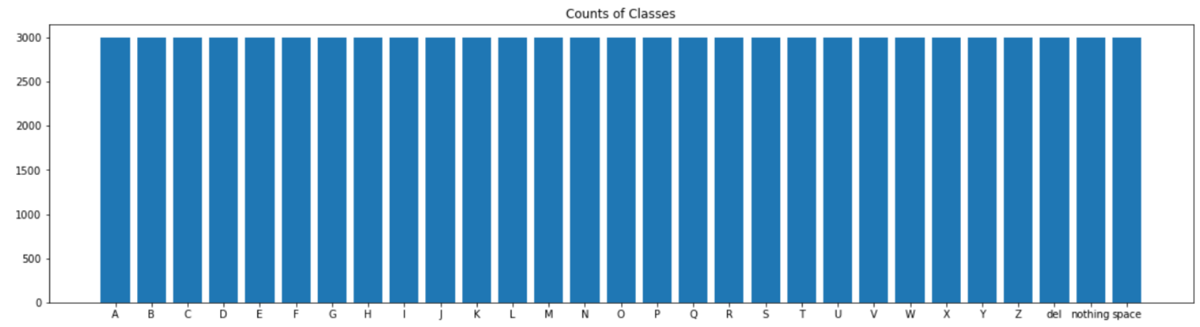
Listed below are some references used in the development of this project. Key insights were extracted from each of the above references, and are discussed below:

- https://www.kaggle.com/code/ryuodan/asl-detection-walkthrough
- https://www.kaggle.com/code/razinw/asl-alphabet-classification-with-cnn
- https://www.kaggle.com/grassknoted/asl-alphabet
    - The three links above are websites that were referenced to help build the neural network models and encode the data.
- https://www.bbc.com/news/health-47475036

    - Title: Cochlear implants to benefit more people with hearing loss
    - This paper describes how cochlear implants are slowly being more accessible to those who need it, however currently very few people can afford and receive these implants.
    - This article demonstrates the continuing need for ASL, as not everyone will be able to obtain a cochlear implant in the near future. Thus, using machine learning to translate ASL can be a cost-effective solution to help those who are deaf and do not have cochlear implants.
- Using Deep Convolutional Networks for Gesture Recognition in American Sign Language
    - Title: Using Deep Convolutional Networks for Gesture Recognition in American Sign Language (2017)
    - The paper describes a Deep Learning approach for developing a classification algorithm for the American Sign Language. The paper used deep convolutional neural networks for this task, using a training/testing set that consisted of images taken by the researchers themselves (researchers took 25 images from 5 people, after which they split the images into testing and training).
    - Because the dataset and pictures were not taken in a controlled setting, the researchers encountered problems with differences in lighting, skin color, and other environmental issues.
    - Ultimately, the researchers observed 82.5% accuracy on the alphabet gestures using a testing set that already existed (NZ-ASL dataset). The testing performance on the self-generated data set had an accuracy of 67%.
    - This article illustrates one study that used a similar methodology to use computer vision to identify and translate ASL. This paper provides insights and information regarding potential areas that could be problematic and cause challenges.
- A convolutional neural network to classify American Sign Language fingerspelling from depth and colour images (2017)
    - This paper develops a convolutional neural network "aimed at classifying fingerspelling images and depth data."
    - The input consists of an image of a finger sign in the form of three feature maps (YUV components), each with 32x32 pixels, and one feature map of 32x32 pixels for the depth.
    - The overall evaluation shows better performance than previous studies as discussed in the paper, with 82% precision and 80% recall.
    - However, the paper notes that there is an inherent challenge in using machine learning for prediction tasks.
    - Recognizing signs from images is challenging because of the variation in size, position, and shapes adopted by people.

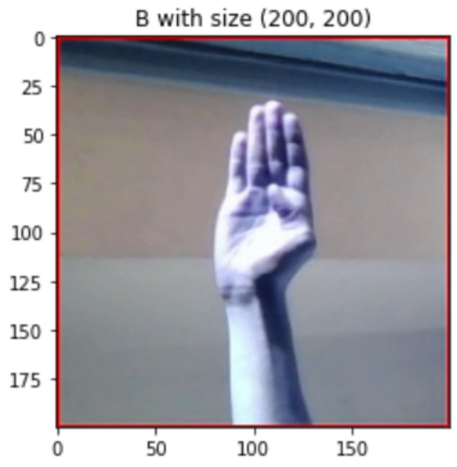
    - This paper is another study illustrating a similar methodology to classify the American Sign Language from images. It provides background information regarding the nature of the encoded images and how images were preprocessed for model development.
- Alphabet Sign Language Image Classification Using Deep Learning
    - Data and image inputs were collected from a wide variety of lighting conditions, background, and other environmental features.
    - Python was used to implement the data collection through a laptop camera with an image size of 120 pixels by 160 pixels.
    - Some pictures intentionally included images of people in the background holding up their fingers to introduce noise and improve robustness of the dataset.
    - Model used a deep neural network, with data input as RGB values.
    - Overall accuracy of the model was noted to be 90.3%.
    - This paper provided details of techniques that could be used to overcome the challenges with taking pictures. Including images with "noise" helps improve the robustness of the data set and model.

Dataset and Exploratory Data Analysis:

- The link to the original dataset is here: https://www.kaggle.com/grassknoted/asl-alphabet
- The bar graph below illustrates that there are an equal number of instances for each class in our dataset. Thus, a balanced dataset will be used for this multiclass classification task.
    - 

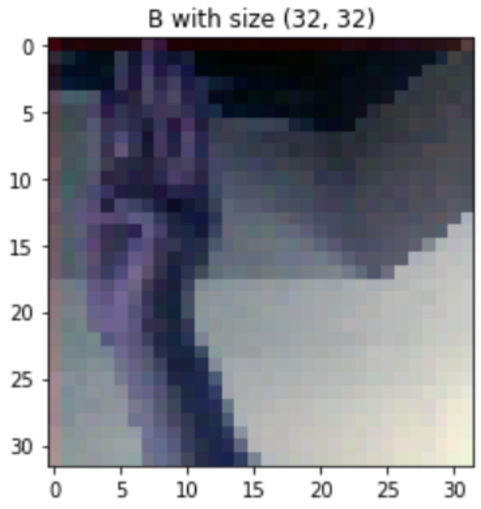
- Below we have various images of different sizes. Since we are going to reduce the size of the image, we want to see the impacts of this from a high-level overview.

```
original_size = 200, 200
show_image('B', signs, original_size)
```



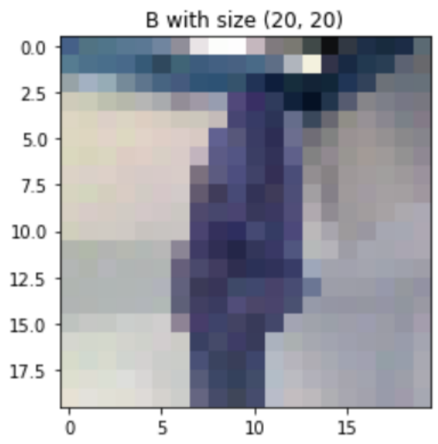
-

```
ideal_size = 32, 32
show_image('B', signs, ideal_size)
```



-

```
reduced_size = 20, 20
show_image('B', signs, reduced_size)
```



- In total, there are 87,000 records (images), and 29 classes (images consisting of every letter of the ASL alphabet, "space", nothing (no hand in the image), and delete). There is no missing data in our dataset.
- Since our images are colored, 1 encoded image is an np.array with shape (1, x, x, 3) where x, x is a manually defined size (we initially chose 32, 32). However, in order to feed this into a logistic regression model, our encoded image must be two-dimensional. Thus, we reshape the image to have dimensions (1, x^2 * 3). The type of data stored in this array is an integer ranging from 0 to 255, essentially describing the pixel of the image.
    - Based on this logic, 4500 encoded images (the size of a training set for two labels) has shape (4500, 32, 32, 3) → (4500, 3072).
- We do not perform any other feature engineering for these images.

Approach and Methodology:
Our general approach to solve this problem was:

1. Data Import
    a. Read in file paths from images and obtain labels from source data (linked above).
    b. Store in dataframe
2. Data Exploration
    a. Explore class distribution of labels in the dataset
    b. Generate random images of various sizes and signs to understand context of resized images
3. Data Processing
    a. Encode data into numpy arrays of integers from 0 to 255
        - Depending on parameters of function, the size of the numpy array will vary.
        - Allows us to feed data into our sklearn models and neural networks
4. Data Modeling + Evaluation

    a. Use baseline logistic regression model through sklearn to provide proof-of-concept and evaluate performance on a “simpler” model
        - Utilize accuracy, F1 score, precision and recall metrics, as well as a confusion matrix and ROC curve
        - Use two selected classes of ASL as binary classes as proof of concept for classification.
    b. Build 4 different neural network architectures and compare performances of each
        - Manipulate input size, number of hidden layers, number of filters
        - Utilize accuracy, F1 score, precision, recall, and confusion matrix to evaluate performance
    c. Use k-fold cross validation as well as train-test split to evaluate performance of neural networks on suitable dataset

5. Summary
    a. This will be described later in our report in the discussion and conclusion sections.

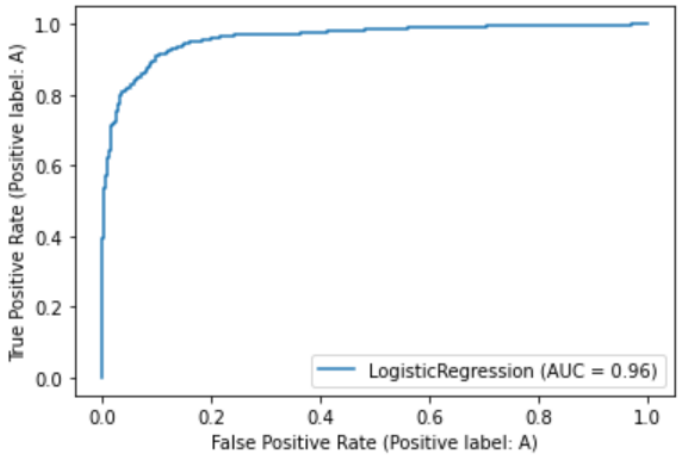
Discussion and Result Interpretation:

As discussed in the methodology section, a baseline logistic regression model was trained and tested as a proof of concept in order to analyze how the model performed with the data on a relatively “simpler” model. Because it is expected that neural networks will perform better than logistic regression on image classification, the development of the logistic regression model serves as a baseline for the accuracy metrics; it is expected that the neural network architectures will perform at least as well as this logistic regression model. Furthemore, logistic regression can only perform binary classification, and would not be effective in performing our full machine learning task.

A train test split approach was used to split the data between training and test sets based on a given set of two labels (two inputs for binary classification: as discussed this was performed as a proof of concept); a logistic regression model was fitted on the training sets, and was evaluated on the test sets based on the provided labels. The sample metrics below represent the performance of the logistic regression model:

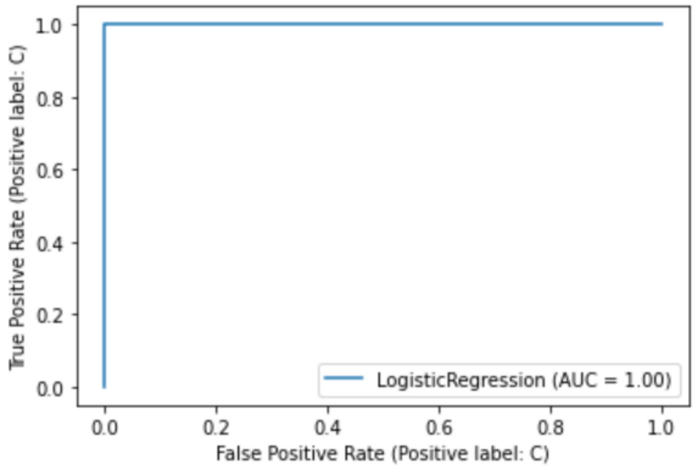
```
Logistic Regression Metrics: A and B (Similar Labels):
-------------------------------------------------
Testing Accuracy: 0.9013333333333333
Testing F1 Score: 0.896358543417367
Testing Recall: 0.898876404494382
Testing Precision: 0.8938547486033519
```

```
Confusion Matrix:
[[640  72]
 [ 76 712]]
```



```
Logistic Regression Metrics: C and G (Different Signs):
---------------------------------------------------
Testing Accuracy: 1.0
Testing F1 Score: 1.0
Testing Recall: 1.0
Testing Precision: 1.0

Confusion Matrix:
[[712   0]
 [  0 788]]
```



Based on the initial results for the baseline, we can see that logistic regression performs relatively well, depending on the form of the input. When the signs are similar, such as with the example of images representing “A” and “B”, the model performs relatively well, with an accuracy of around 0.901; we can see that the precision, recall, and F1 score metrics are also high, with all values around 0.89.

In the case that the hand image signals are very different (in the case of “C” versus “G”), the model performs extremely well, with a 100% accuracy. This shows that the logistic regression can separate and classify hand signals especially well when the two labels are visually different.

However, as discussed, the inclusion of the logistic regression model was to analyze and establish a baseline to which the main neural network architectures can be compared to. Because the accuracies established in this initial study were between 0.90 for similar labels and 1.0 for different labels, we expect the neural networks to perform better overall and produce better metrics.

It is expected that the evaluation accuracy of the logistic regression model will be lower than that of neural networks due to the relatively “simple” nature of the model; it is a linear classifier that predicts the class probability of an input. Therefore, using a non-linear classifier such as neural networks, is expected to give more accurate results.

4 Neural Network models, were developed as discussed in the methodology section; their architectures, results and metrics are illustrated below (after training and testing using cross-validation and train test split)

Summary Testing Set Metrics (Table):

NN	Architecture	Params	Precision	Recall	F1	Accuracy
NN 1	3 hidden layers (20, 40, 40 filters), input size (20, 20)	25,069	0.817	0.805	0.805	0.805
NN 2	3 hidden layers (32, 64, 64 filters), input size (32, 32)	123,805	0.971	0.969	0.969	0.970
NN 3	3 hidden layers (45, 90, 90 filters), input size (32, 32)	243,119	0.974	0.971	0.972	0.972

NN 4	1 hidden layer (32 filters), input size (32, 32)	232,285	0.491	0.495	0.487	0.495

Interpretation of Results:
Based on the above results, we can see that **NN 3** – the architecture of 3 hidden layers (45, 90, 90 filters), input size (32, 32) → 243,119 parameters – performs the best, because it has the highest accuracy value and F1 score of .972.

NN 2 and NN 3 had similar performances, however the runtime of the two neural networks were different. The difference between NN 2 and NN 3 is the filter size. Increasing the filter dimensions also appeared to increase the model accuracy slightly; the filter in NN 3 was of size (45, 45, 90) and yielded 0.972 accuracy, as opposed to the size (32, 32, 64) of NN 2 which yielded a slightly lower 0.970 accuracy. Even though we reduce the size of the image from (200, 200) to (32, 32), we still maintain an accuracy of .97, which is exceptional. Furthermore, each epoch for NN 2 took about 80 seconds, which led to a total runtime of 400 seconds. Each epoch for NN 3 took about 120 seconds, which led to a total runtime of 600 seconds. Thus, based on our findings, we can assume that as our input size grows, the difference of runtimes between the two architectures may increase exponentially, with only a marginal increase of performance.

NN 1 performs worse than both NN 2 and NN 3, with an accuracy of 0.805. NN 1 used encoded images of size (20, 20), which contributed to the lower accuracy, since we lose pixels (and thus information) from the image. On the other hand, the runtime of this neural network was much faster compared to NN 2 and NN 3, with epochs taking approximately 20 seconds per step. This experiment illustrates that even if we had very limited computational power, our resulting model still performs somewhat well. Considering how important accurate ASL translation is, however, we cannot afford to make these mistakes (similar to a false negative with detecting cancer) because there is no translator in place to rectify them.

Lastly, NN 4 performs the worst by far with an accuracy of 0.495. We created this neural network to evaluate the impact of hidden layers in a neural network on its performance. Unfortunately, having only one hidden layer does not lead to optimal performance for this task. It can be observed that having more hidden layers increases the complexity of the model, and creates a neural network that generalizes to new data more effectively. Ultimately, NN 4 was likely not complex enough to make the decisions to classify the images based on the training data.

In addition, when fitting the neural networks, we felt that setting the number of epochs to 5 balances the runtime of fitting the NN model itself while also allowing the model to improve after each epoch. Ending on one epoch would lead to poor validation accuracies (since we don't have an abundance of training examples), but using too many epochs would lead to diminishing returns with added runtime. Tweaking the batch size and epoch hyperparameters may lead to slightly better results, but we felt the NN architecture itself would be more influential on the outcome of our models, so we spent most of our effort devising those instead.

Conclusion:

In conclusion, the models were very successful in approaching this machine learning task, and overall, NN 3 with three convolutional hidden layers of input size (32, 32) with a (45, 45, 90) filter size performed the best, with an accuracy of 0.972. Although previous work has been done on this topic, a lot was learned about building neural networks to tackle practical tasks. Furthermore, we gained experience working with image data, which none of us had experience with. Although we had some knowledge implementing neural networks for NLP tasks, we had never used them for computer vision. Nonetheless, this topic has furthered our interest in the computer vision field, and we have thought about future applications of machine learning for ASL translation. Although the ASL alphabet provides a basis for forming words, generating longer sentences in ASL are prone to more errors in machine learning models and take more time to compute. For instance, there are ASL signs for certain words, and these signs can differ depending on the context the word is being used in. In addition, ASL also contains gestures that represent words, which cannot be precisely translated through images. Ultimately, we wonder how machine learning can be adapted (or if at all) to capture these nuances in an extremely important language in today's society.

Additionally, it can be concluded that computer vision can be used in translating the ASL alphabet with a high degree of accuracy; however, it can also be concluded that translating the ASL alphabet through computer vision is a relatively challenging task. The nature of the images and how they are taken pose potential problems in model training and analysis, and as discussed in some papers listed in the references section, these variations in images (such as lighting, background noise, skin tone) can affect the model's performance and the model training.

The results from this project do show that machine learning can be used to translate the ASL with a high degree of accuracy; however, expanding and scaling this approach to using more sentences, words, and images needs to be tested and analyzed. These are some ideas that should be explored in the future/in future work.

Team Member Contribution:

We pair-programmed for a majority of this project, and performed most tasks together.

- Justin Lau
    - Worked on processing the input data, exploring the data, and obtaining readable representations of the images for use (import, exploration, encoding).

    - Developed models and evaluated models for performance (data modeling and evaluation).
    - Helped develop final report and project video
- Tejas Sathyamurthi
    - Worked on reading papers and getting background information about the topic, existing methods, and neural network architectures, among other details (topic research)
    - Worked on processing the input data (data import)
    - Developed models and evaluated models for performance (data modeling and evaluation).
    - Helped develop final report and project video