ASL Gesture Detection with Computer Vision

Varun Mohan
February 8, 2023

1. Executive Summary

The objective of this project was to build a machine learning model capable of taking in images and/or video data of a human performing American Sign Language symbols and correctly classifying them. This required a combination of two tasks and therefore two models: an object detector to detect hands present in an image, and a classifier to decide which ASL symbol is being performed.

Image data was sourced from the Roboflow ASL Image dataset, which came with some light augmentation including a horizontal flip for each image and one variation in brightness. The data was further augmented with rotation variation and more variation in brightness.

The trained model was a MobileNet CNN with a custom head layer, and achieved an accuracy of about 70% and precision of about 85% on the testing data, which was satisfactory but definitely showed room for improvement. Improvements to make the model more accurate include expanding the dataset to make it more generalizable and pursuing other types of models to compare performance with MobileNet.

This model has the potential for meaningful practical use, namely, bridging the gap for communication between hearing impaired people and those who do not know ASL. The model could also be used in pedagogical settings to aid with learning ASL.

2. The Problem

Verbal communication is something almost all of us do every single day. It is practically our only way of communicating with others, and as a result we generally take it for granted. But what if you could not hear, and could not effectively verbalize the way others do? This is something the deaf community faces any time they have to communicate with others who are used to speaking; and the challenge occurs the other way as well, making it difficult for those with hearing to communicate with those without it. But what if we could help bridge that gap?

Much of the deaf community relies on sign language to communicate. In the United States specifically, American Sign Language, or ASL is most often used. It's a language as effective and expressive as any verbal one, but this also means learning it presents a similarly significant challenge.

The goal of this project is to create a model and application capable of receiving image data of American Sign Language hand gestures, and both detecting them and classifying them into one of 26 classes corresponding to each ASL letter. Deployed to an application, this model can be used to help people who do not speak sign language begin to understand it on a basic level without the need of a translator.

While this project is limited to classifying ASL letters, one could extend the library of training data to include more words and phrases to scale an application into a true ASL to English translator. At the very least, I hope to create a proof of concept that this type of translator is very much attainable with the power of computer vision.

3. Related Work

3.1 David Lee's YOLOv5 Model

The primary inspiration for this project is a dataset and model built by David Lee, a data scientist and engineer at Roboflow, a company with a suite of applications and models supporting computer vision tasks. David created his own dataset by taking pictures of his hands performing the 26 ASL letters in a few different environments each, with augmentations done to each image as well to improve generalizability after training [1].

Augmentations include variations in flipping, cropping, rotation, shear, grayscale, brightness, and blur. The result is a total of 1728 images, with 1512 in the training split, 144 in validation, and 72 in testing. This dataset is planned as the primary source of training data for this project as well.

For modeling, David used the vision model YOLOv5 made by Ultralytics. This model is made for object detection, and transfer learning was used from YOLO's pre-trained weights to apply to the ASL dataset for training. The model was trained for 300 epochs.

For evaluation, mean average precision was used. This metric calculates the overlap between the actual (ground truth) bounding box and the predicted bounding box for each class, and takes the average over all of the classes. This is a common metric used to evaluate the performance of object detection models.

Overall, David's project is one of the best examples of an ASL gesture detector. However, David did note that his model does have some trouble generalizing detection

to other people. When other people used the model to detect their ASL gestures in their own environments/lighting, the model had less confidence classifying each letter. Still, it was usually correct which is impressive.

3.2 Real-time Object Detection and Classification for ASL alphabet

A similar project was done in a research context at Stanford University by Rain Juhl and Eric Feng. Using the same dataset, this team trained a few different vision models on the images and evaluated/compared their relative performance. Data preprocessing was largely the same; augmentations were done on each image to create several slightly different duplicates of each letter to improve generalizability [2].

The first model used here was the Faster RCNN ResNet-50. This model first uses a CNN to create a Region Proposal Network and a second model to make the final region detection based on the network. ResNet50 is trained on ImageNet data detecting ~1000 object classes; the output layer was changed to 27 nodes to classify the ASL letters.

Rain and Eric also trained a Faster R-CNN MobileNet model, which based on performance and effectiveness, is planned for use in this project as well. It uses the same Region Proposal method as ResNet-50, but MobileNet is a lighter model designed to be used with CPUs and even mobile devices while retaining a solid level of accuracy. The important part of MobileNet is that it uses depth-wise separable convolutions.

Mean average precision was used again for evaluation of the models used; the authors noted that this is the best way to evaluate the model's performance as both an object detector and a classifier simultaneously.

Performance was strong for both models which lends greater confidence to MobileNet being a worthwhile tradeoff for better computational cost. The authors noted some specific difficulty with similar looking letters, like M and N, which were more likely to be misclassified by both models.

4. Data

To train a classification model, as well as to provide scaffolding code for handling bounding boxes, David Lee’s American Sign Language Dataset was used. This dataset consists of 1728 images of sign language hand poses: 1512 in the training set, 144 in the validation set, and 72 in the testing set. There are 26 classes corresponding to the 26 letters of the ASL alphabet. For each letter, there are multiple image versions which

vary the photo angle and ambient lighting/background, and each photo within a photo is triplicated with manual augmentations to brightness and a horizontal flip.

Along with each image file is a corresponding .xml file with annotations in PASCAL VOC format; these annotations specify the dimensions of the image and, most notably, the coordinates of a bounding bound isolating the hand in that image.

It is worth noting that this is a relatively sparse dataset for a multiclass image classification problem. This was taken into account during the preprocessing and modeling stage, but, as will be elaborated in a later section, further iterations on this project would certainly benefit from an expansion of the dataset to more images per class. On a similar note, since the training images were all photos of David Lee's hands performing the ASL gestures, one could make a model more generalizable by introducing ASL images with greater variation in attributes like hand size/shape, skin tone, ambient lighting, etc.

With all of this considered, this was a very well-crafted dataset given its size and restrictions, and modeling performed relatively well with unseen data.

5. Modeling

In line with my initial proposal as well as with the previous work done by David Lee and the Stanford research team, I developed a two-part model for object detection and classification. The ultimate objective for prediction was as follows: the object detector would detect and bound a hand within an image it was passed. Then, additional code would isolate the portion of the image inside the bounding box. And finally, a neural network classifier would take as input the isolated hand image and classify it as one of 26 ASL letters.

5.1 Hand Detection with MediaPipe

For the first task of detecting and bounding a hand within an image, I used Mediapipe: a suite of ML models and solutions built by Google. Among its functionalities is hand and finger tracking, for which it combines a palm detection model as well as a hand landmark model [3]. The MediaPipe Hands model is trained on over 30,000 data points, and provides landmark coordinates for palms and fingers, and is able to differentiate between left and right hands.

So, while the actual hand detection model was trained by Google on a very large dataset, the challenge was to transform the output of this model into a bounding box, where it normally provides more detailed landmark coordinates. This was done with

some manipulation of the coordinate data and testing in opencv: I wrote a python script that runs a detection with MediaPipe on an image, checks all landmarks currently being detected and checks every x,y-coordinate. It then takes the top-left x,y coordinate and the bottom-right x-coordinate as the preliminary points to draw a bounding box. It adds a small additional buffer space (23 pixels was found to be optimal) in each direction, before drawing and recording the bounding box coordinates with opencv.

5.2 Preprocessing for Classification

Since a neural network classifier has relatively strict shape requirements, I needed to take the resultant data from the object detector and create a pipeline of preprocessing steps to prepare it to be passed in. For each image, once a bounding box was recorded based on the data from MediaPipe and the appropriate adjustment, the image was "cropped" to include only the pixels that were within the bounding box. In accordance with the requirements of MobileNet V2, described in the next section, each image was then resized to 224 x 224 pixels, and transformed into a numpy array representing the RGB value of each pixel. The training set was thus transformed into a (1512 x 224 x 224 x 3) array.

Labels for each image were one of 26 letters and were extracted from the .xml files. These labels were encoded using sci-kit-learn's LabelEncoder for compatibility with TensorFlow, making the training label array 1512 x 26.

Once both the images and labels were processed into arrays as specified, the data was ready for training.

5.3 Classification with MobileNet V2

Due primarily to time constraints, for the purpose of this project, the only model used for classification was MobileNetV2, a convolutional neural network that is specialized as a particularly lightweight and efficient CNN compared to most others. This model in particular was chosen based on its documented balance between strong performance and efficiency for prediction. It also proved effective for live predictions on a video stream with relatively high frame rate, which would have been more difficult to achieve with larger/less efficient models

MobileNet achieves its remarkable efficiency through its use of depth-wise separable convolutions. An ordinary convolution would take a kernel of size:

$$D_k \times D_k \times M$$

and convolve over the feature map of size $D_F \times D_F \times N$. This would result in a total number of multiplications equal to:

$$D_k \times D_k \times M \times D_F \times D_F \times N.$$

With depthwise separable convolution, we have one $D_k \times D_k$ kernel per input channel running separately. Because these three convolutions don't inherently interact with each other, a second step is required: pointwise convolution. In this step, we have a single $1 \times 1 \times M$ kernel which convolves over the feature map again and creates a linear combination of each depthwise layer. The total cost of the calculation, with depthwise convolution followed by pointwise convolution, is significant improved over a traditional one-step convolution:

$$D_k \times D_k \times M \times D_F \times D_F + N \times M \times D_F \times D_F.$$

MobileNet is trained on ImageNet, a very large image database with over 20,000 classes and millions of datapoints. Clearly, we cannot effectively work with MobileNet as it works by default. I therefore took advantage of transfer learning and redefined MobileNet's head layer to work with the task of classifying ASL. Using TensorFlow, I retrained the head layer on the ASL training data, and used the Adam optimizer which seemed to perform the best for the model. Some additional data augmentation was also performed as the model was trained using ImageDataGenerator: the image rotation, brightness, and zoom were all varied slightly.

6. Results

Several head layer architectures and hyperparameter combinations were tried in order to center in on the best model. The highest performing model used the Adam optimizer with a weight decay of 0.1, two dense layers with 512 and 256 nodes, and a dropout layer of 0.5 for the latter dense layer. Accuracy for the training set peaked around 87%, with precision at around 90%. On the validation set, accuracy was a bit lower at 75% indicating some overfit, but precision was a bit higher at 83%. These results were fair if a bit short of ideal; however considering the number of classes and the limits of the source data, this was a relatively good performance.

Some insight into the performance limits were discovered during more manual testing of the model on video stream; namely letters which the model seemed to

“confuse” with each other. The model performed best on letters that were very distinctive from others, such as Q and L. However, several letters are quite similar in their gestures, such as A,M,N, and T. These were confused particularly often, and my hypothesis is that the fact that they are all “closed-fist” letters made it more difficult for the CNN to pick up salient differences. Requiring a resize for each image to a strictly 224 x 224 image likely exacerbated this issue.

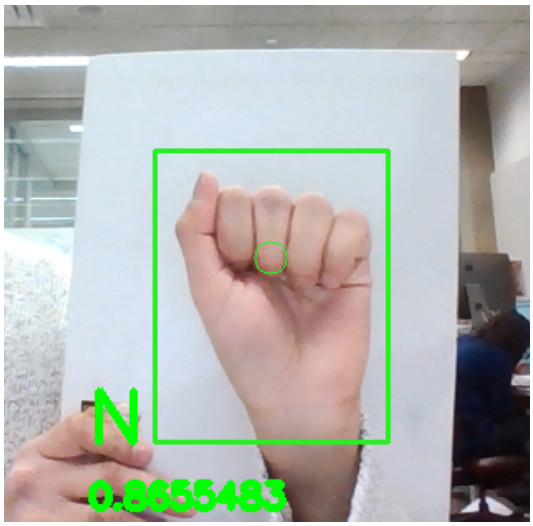

Fig. 1: An example of a misclassification of the Letter A as the Letter N; both are closed-fisted letters.

The trained model itself, as well as a Python script to run a live detection on video stream is included in the repository in the App directory.

7. Conclusions and Future Work

Considering the limitations in the data and the use of a more lightweight model, the results and performance of the MobileNet model was impressive. The primary pain points resulted from a lack of breadth and variation in the data, as well as classes, i.e. letters that presented very similarly during modeling.

These types of performance shortcomings can potentially be resolved within the scope of this personal project by acquiring more data and fine-tuning more models, and so they are on the list of goals for the near future. However, the scope of this project does not represent the true potential for this space of computer vision; the alphabet is far from a complete use case for a sign language translation technology.

The future of this space lies in training models using ASL words and phrases, extending the scope to encapsulate not just an alphabet, but an entire language.

This obviously represents a massive undertaking for the data science community, requiring entire databases of image data relating to an entire sight-based language. However, given the size and growth of large language models like GPT3 as well as image models trained on ImageNet like MobileNet itself, this is not at all outside of the scope of possibility. For sign language detection to reach its potential, this is the direction we need to move toward.

Despite the fact that this project is limited to the ASL alphabet, it nonetheless represents an important proof of concept for how much we can already do with ASL using machine learning. Moreover, it is a showcase for the potential that computer vision has to create a truly positive impact for those who rely on sign language to communicate with others. As long as the majority of us default to verbal communication, there will exist a gap in communication for the hearing-impaired; but with machine learning, we may be able to bridge that gap.

8. References

1. Roboflow: American Sign Language Letters Dataset: https://public.roboflow.com/object-detection/american-sign-language-letters
2. Rain Juhl, Eric Feng. Real-time Object Detection and Classification for ASL alphabet, 2022. http://cs231n.stanford.edu/reports/2022/pdfs/147.pdf
3. MediaPipe Hands: https://google.github.io/mediapipe/solutions/hands.html