AI for ASL

Avnish Jadhav, Belton He, Urvi Patel

There are more than 70 million deaf people around the world who use sign languages to communicate. Although there are various resources to learn sign languages such as with courses or videos, these methods take extensive amounts of time to learn and become conversational in the language. Spoken languages have electronic translators which are easily accessible, simple to use, and allow for quick and immediate translations between numerous languages. This technology, however, does not accommodate users of sign languages and those wishing to communicate with someone using sign language. Our team wanted to help bridge this gap and create a more inclusive environment by creating an American Sign Language Translator.

We started off by choosing a neural network. A Convolutional Neural Network (CNN) is a Deep Learning algorithm that takes an input image and through its layers of convolution, ReLu, and pooling, accurately classifies the image based on the classification label. We chose CNN because CNN's are specifically designed to process pixel data and are best for 2D images. The traditional Neural Network known as Multilayer Perceptron (MLP) is not great at image processing because it reacts poorly to shifting versions of an input image — it assumes the object in the image will be in the same area at all times. CNN on the other hand, does not pose this problem because it breaks down an image into multiple groups and it only focuses on groups of pixels closest to each other.

Next, we chose the 3 different models we wanted to test: Simple Classifier, DenseNet, and GoogLeNet. We hypothesized that having a pretrained and more complex model would be better than a simpler model because It would have a higher accuracy and lower loss. For the Simple Classifier model, we used the Pytorch Blitz tutorial for making a classifier since we all didn't have any previous experience. Our data, which consisted of images of the ASL alphabet hand signs, was passed through this model, and after training the Simple Classifier model it reached an accuracy of 7% and the loss decreased overtime, but the slope of this function was not very steep and could definitely be improved. The second model we used was DenseNet. We chose DenseNe because during our initial search to identify which models would be best suited for image classification, DenseNet was a popular choice. Diving deeper we found that some of the key traits of DenseNet are the emphasis on strengthening feature propagation, and encouraging feature reuse — which is great since CNN emphasises on manipulating features. However, things don't always go as planned, and when we ran DenseNet the results reflected poorly on our hypothesis that this model would yield a high accuracy. The last model we used was GoogLeNet. We chose this model because of its 22 layers deep network, specifically designed for classification and object detection. This model also won a Visual Recognition Competition in 2014, so we thought it would be interesting to test this model with our dataset. GoogLeNet surpassed both previous models and had the highest accuracy of approximately 14.85% with 11 epochs.

Lastly, we played around with different hyperparameters and data augmentation to see if the accuracy of our model could be improved. We first modified the learning rates of all three of the models to try and find the most optimal for each. Learning rates are a hyper-parameter that affects how fast the algorithm learns, and whether or not the cost function is minimized. Testing different learning rates turned out to be very useful and we were able to achieve ~14% accuracy on GoogLeNet. The next hyperparameter we modified was the number of epochs, or in other words, the number of times we iterate through all of our data during training. Increasing the number of Epochs gives more insight into how the model learns over time. Increasing epochs also shows when the accuracy plateaued signifying when the model reached its optimal performance. Finally, we worked on some data augmentation to see if it affected our results. We got all our data from Kaggle, but we realized that the images were all very similar, and our accuracy was not very high. We decided to apply multiple different augmentation techniques to the images in the dataset to see if it helped our model learn better and improved our accuracy. Our results found that there was not a big difference in our results when training our models with augmented vs non-augmented data.

Out of all three models we chose, Simple Classifier, DenseNet, and GoogLeNet, we found GoogLeNet to be the best. Although we were able to get such great results, this was not without some issues. When we started this program back in October, all three of us had very limited knowledge of AI and ML. We spent a few weeks getting a background on this topic before we started coding and creating our project. Early on, we also had some problems with Skynet and Jupyter notebooks which hindered our progress and reduced the time we had to work on the project. Next, since our initial test case contained only one image per letter, our accuracy was skewed, making it either 100% correct or 100% incorrect; we realized we needed to add more data to our test case to achieve a more accurate accuracy. Lastly, due to the long run times (approx 9 hrs-12hrs) of each model, getting a good understanding of how we can improve our model was delayed. Nevertheless, even with all these challenges, we were able to achieve such great results and we learned a great deal.

There are some future applications and improvements we would like to mention. First, we would like to use more fine-tuning to improve the accuracy of the predictions. By running more tests on our data we would be able to understand how each specific model learns and what its capabilities are. Next, we would like to increase the number of hand signs our model can identify. Currently, we focused on static hand signs for the alphabet, but ASL is a very dynamic language and most signs rely on movement. By using this starting point of static hand signs, the model can be modified and improved to recognize dynamic signs. Lastly, we hope our project can be improved to have a higher accuracy and be turned into an app or website for use as a translator with a live camera.