Final Report: American Sign Language Recognizer

Wesley Penn, Mark Haghani, Brandon Reid, Ronil Synghal

December 19, 2022

1 Project Summary and Goals

The goal of our project was to create a system that would understand and interpret American Sign Language (ASL) gestures and hand shapes in order to translate them into written language.

We wanted our finished product to be able to interpret an image of a human gesturing ASL from a distance and return the letter that human is signing. Our model will detect if the subject is actually performing an ASL sign via CV2 Hand Feature Detection Model and then we will run three linear classification models to identify which letter has been signed.

Ultimately, this project's goal is to bridge the communication gap between the hearing and deaf community. We will achieve our goal in two ways. Firstly, by giving free access to our ASL Recognizer we will improve direct dialogue between deaf and non-deaf people. Secondly, we will improve ASL education for non-deaf people, who will be able to practice on their own and get immediate feedback on their hand gestures using this software.

2 Progress / Achievements

After an extensive search for an ASL dataset with object detection, we found Roboflow's ASL dataset. In this Google Collab notebook we load, clean, and augment that data using PyTorch. During our dataset search we had to ramp up on PyTorch, which we applied and structured our data with.

2.1 Data Collection and Pre-processing

For the sign detection and localization, we use anAmerican Sign Language Letters Dataset hosted on Roboflow, which provides a COCO formatted and YOLO formatted dataset with images and bounding box / classification labels. The dataset contains 1512 training images, 144 validation images, and 72 testing images.

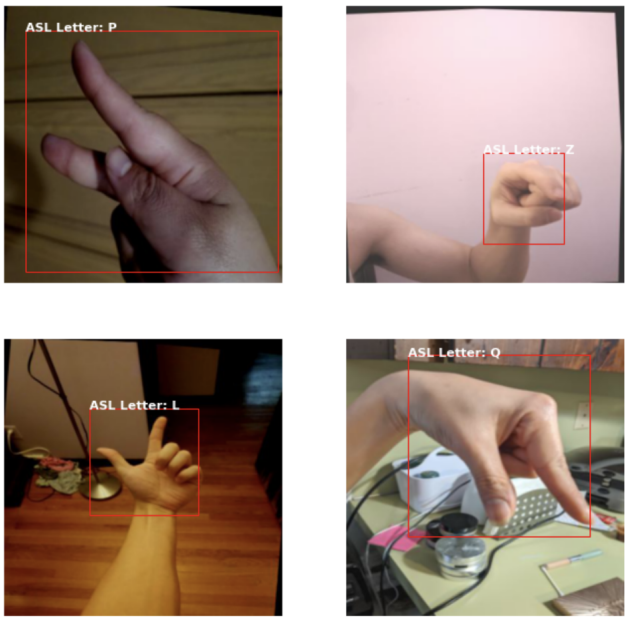


Figure 1: Images From Roboflow Dataset Post-Augmentation

One step in our data preprocessing was to account for gestures given by both left and right hands. We accounted for this by flipping the images horizontally. Since most of the dataset was right handed, left handed gestures were more neglected. We also made it possible to resize our images and bounding boxes to adjust for different image sizes. This step was quite technical as we had to proportionally change the bounding boxes to fit the new image size. We had to create a custom coco dataset to perform our data augmentations. The standard CocoDetection PyTorch dataset allows for transformations, but those are only applied on the image; the bounding box labels do not get scaled proportionally. Therefore, we had to create a custom dataset with pytorch where we define the functions to load in the image and labels, and perform our data augmentations to resize the image and labels. We used a bilinear interpolation method (similar to our "interp2()" function from class) in order to upsample and standardize the images. During this step we also were able to clean up the data to improve our runtime. We removed unnecessary data attributes such as 'area' and 'iscrowd', which we don't use and would be wasteful to load into the dataloader for the training.

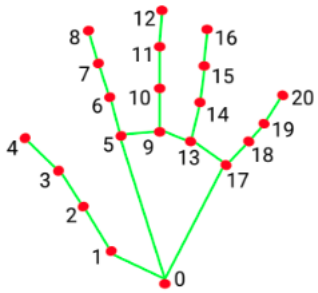
2.2 Models

We used two models to ultimately detect whether an ASL letter was being signed and if so which letter. We trained a CV2 Hand Feature Detection for the first model and built three Deep Linear Classification models (CNN, DLN, and a Composite model; a combination of CNN and DLN) to choose from for our second model.

CV2 Hand Feature Detection Model

For our Hand Feature Detection Model we have implemented the Hand Landmark Model used by Mediapipe. The model detects and returns 3D coordinates for hand-knuckle landmarks (the image below shows those points on a hand). When we trained our model on our dataset, we extracted these twenty-one hand-knuckles landmarks, normalized their 3D coordinates and set their positions to features. We then used these twenty-one features to predict the ASL letter that was shown. The

Mediapipie model we used underwent its own extensive training, which combined real-world images with manually annotated coordinates and hand gestures over various backgrounds. A benefit of using this model is its robustness in identifying these hand regions even when the hand is partially visible or occluded. In doing so, we are able to learn consistent internal hand pose representation and predict which ASL letter was being signed.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

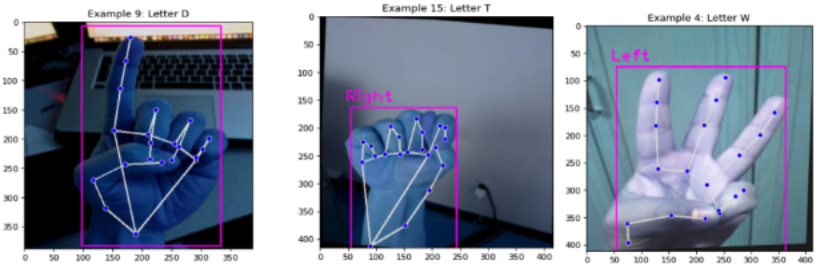
Normalization step

After identifying the 3D position of each hand landmark we normalized these coordinates in the 3D using PyTorch's InstanceNorm1d function. Since the person's hand could be in a different part of each image and at different distances away from the camera, the same Hand Landmark's across two images would be different and hence be incomparable. Additionally, by normalizing the coordinates of the twenty-one Hand Landmarks we identified the relative position of each Hand Landmark to one another.

Considering alternative models to use

When building our model pipeline for our project we considered using several alternative models for the first phase in our pipeline. We were most serious about using a Human Pose Estimation model instead of our CV2 Hand Feature Detection model. Using Human Pose Estimation, we would have first detected whether a person was signing, then cropped the image to the area around the hand, and finally used interpolation to then pass the focused image of the hand through to the second phase of our pipeline. However, the challenge of using Human Pose Estimation was in the cropping stage. Locating the hand from only data on arm landmarks (wrist, elbow, and shoulder) made it difficult to get the correct cropping and sizing. As a result, interpolation appeared challenging for the size of the task and we pivoted to the more appropriate Hand Feature Detection model after discussing our issue with our project advisor Neha.

Output of our Hand Feature Detection model:



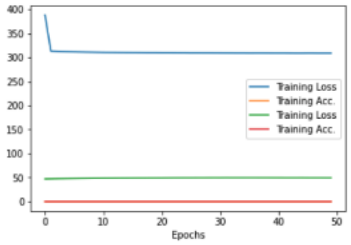
Deep Linear Classification on Hand Features

Using the hand feature data from the previous step, we decided to use Linear Classification to determine the ASL letter being signed. We experimented by building three different models for the second stage of our project pipeline. The first model we built was a CNN (Convolutional Neural Network), our second model was using a DLN (Deep Linear Network) and our third model was using a combination of a CNN and DLN.

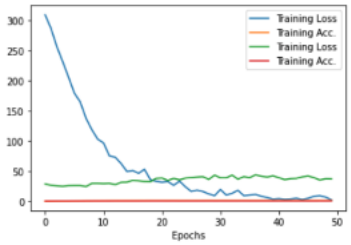
We designed our first model to be a baseline model. We built a CNN with several convolutional blocks and batch normalization in between those layers. We wanted to use this to compare our performance to the other models. Our goal in using the DLN and composite model (CNN and DLN) was to extract additional performance using CV2's ML pipeline for estimating hand landmarks. After that we wanted to see in our third composite model if increasing the number of features would help.

Results

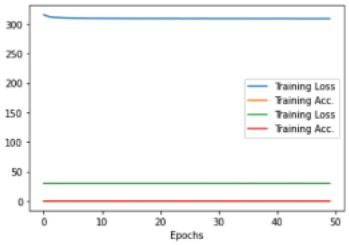
Overall, only one of three models performed well in testing, with the other two underperforming. However, each model outperformed how a random model would have done (a random model would have a 3.7% = 1/27 accuracy rate). Our CNN model, which we used as our baseline model, produced the best results. Our DLN and Composite (DLN + CNN) models performed worse than we expected, especially on the test set.



(a) DLN Model



(b) CNN Model



(c) Composite Model

The figures above show our training loss, training accuracy, validation loss and validation accuracy for each of the models. As you can see, each model has a training accuracy of less than 50% after training. This is subsequently shown in our accuracy metric from our test set as well. The DLN model had a test accuracy of 1.39%, the CNN model had a test accuracy of 22.2% and the composite model had a test accuracy of 2.77%.

We can see from the results that when a model involved DLN, it significantly reduced the ac-

curacy which contradicted our initial hypothesis that DLNs would allow our models to learn more features and perform better. Most likely, having too many features caused our models to not be able to generalize to the test cases.

Overall, we concluded that our CNN model was the best fit for ASL recognition and that our DLN and Composite models were not best suited for ASL recognition.

3 Related Works

1. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields: https://arxiv.org/pdf/1611.08050.pdf
2. Human Pose Estimation via Deep Neural Networks: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42237.pdf
3. American Sign Language Alphabet Recognition by Extracting Feature from Hand Pose Estimation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8434249/