CSCI 1430 Final Project Report:
ASL Recognition

Team Godhand: Will Chen, Robert Bush, Ikenna Ihenatu, Erick Lerena.
Brown University

April 23, 2021

Abstract

ASL (American Sign Language) is the primary language among individuals within the deaf community. Studying ASL promotes better awareness of and sensitivity to those who are deaf or hard of hearing. We wanted to use this project in order to explore how computer vision can be used to aid in the process of understanding ASL.

Specifically, we focused on **classification** for the letters of the ASL alphabet A-Z in tandem with **semantic hand segmentation** for the ASL signs.

1 Introduction

If you wanted to read a book in a different language, a quick Google translate of the text could provide an instant, albeit choppy, approximation. If you met someone on the street who was speaking a different language, there's audio based translation right in the palm of your hand. When it comes to a non spoken language, in our case ASL, these methodologies provide no aid and have no current popular parallel. Our goal is to use computer vision techniques to **better understand how they can be applied to ASL recognition**. Our plan for the project consisted of three parts.

1. Detect and classify all 26 ASL letter digits using a convolutional neural network (CNN) model
2. Semantic segmentation to separate the hands from the image also using a CNN
3. Web application live video interface to connect our two models

2 Classification

Being that our team is relatively new to deep learning techniques and computer vision in general, we planned to base a large amount of our project on what we learned during the deep learning sections of lecture, alongside with the application we practised in Project 4.

We decided to reuse the general format of Project 4 code, the setup with hyperparameters, dataset preprocessing, Tensorboard utilities, since it was given to us and we believed it wasn't necessary to reinvent the wheel. With this as our basis we then began to look at studies and papers to provide a tangible pathway for us to follow throughout the next few weeks.

The first part of our project consisted of finding datasets to use to classify all ASL alphabet digits. Immediately we noticed a potential issue: the letters J and Z required motion which was not easily captured through a static image. Much of the datasets that we encountered had little to no data on J or Z so we understood that these two letters may not be classified as well in our model. We settled on combining two ASL datasets: An ASL dataset from Kaggle and one from here . [1] For our validation sets, we used around a rather small 5% random sample of our training set but augmented it with our own images that we took where we signed ASL digits from various angles and zoom levels. We thought to supply this validation data as it would show how well our model would generalize to outside the training set.

[1] Pugeault, N., and Bowden, R. (2011). Spelling It Out: Real-Time ASL Fingerspelling Recognition In Proceedings of the 1st IEEE Workshop on Consumer Depth Cameras for Computer Vision, jointly with ICCV'2011.

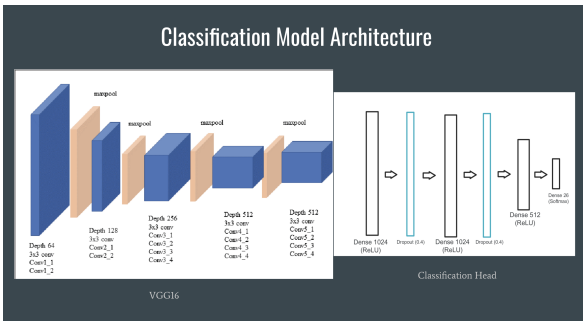


Figure 1: Classification model architecture

We initially experimented with many different model architectures. We tried the classical approach of having repeated blocks of convolutional layer followed by a max pooling layer, similar to AlexNet, but we also tried stacking convolutional layers on top of each other. We tried many of these combinations but our model never seemed to train. (More discussion in the Results section) When we switched to deep transfer learning with VGG16 weights, as we did in class, our model was actually able to train and we were able to obtain a working model with a test accuracy of around 87% (more in results section).

We also used our knowledge we learned in class of data augmentation by applying horizontal mirroring to simulate the possibility of left-hand signs as well as shearing and other transformations to capture various zoom levels. Our model also ended up using the Adam optimizer due to its capacity for adaptive learning rates and that it is pretty well hailed as the best optimizer for many scenarios.

Because our initial plans of creating our own architectures failed, we stuck with mostly using Project 4 code with the VGG16 deep transfer learning as it provided us with the best references to us who were new with deep learning. We trained our model on GCP and received pretty good results as we will demonstrate in the following sections.

3 Segmentation

Having not learned much about segmentation in class, this part of the project proved to be the most difficult. Our initial goal for segmentation was to combine classification with segmentation in that once the classification model classified an image, we would then segment out the hand and display these results back to the user. We used many outside resources including the following. [2] [3]

For hand segmentation, we learned that we would need to train a CNN model to classify every pixel in the input image (after resizing) as either hand or not hand, which would get returned as an image mask of the same size as the image. We settled on a tutorial from the following source to help us with proper methodology. We ended up using a new library, Pytorch, a much more low-level library than Tensorflow, to construct this model as it gave us more control over what was going on at each step of training, including allowing us to specify an IOU (intersection over union) score and pixel accuracy score (correctly identified pixels) as well as other features such as easy loading in of ResNet50 weights, which was what we used as our model to supply deep transfer learning as it was a larger model than VGG16 and we believed would be more powerful and better suited.

The intersection over union (IOU) score along with the pixel accuracy score were the evaluation metrics we used to determine the performance of our model. The IOU score represents the area of overlap over the area of union of the ground truth mask and the classified mask. A score of over 0.5 for IOU is considered pretty decent. The other score is pixel accuracy, which is the number of accurately classified hand pixels over all the hand pixels classified. This is more commensurate to a standard accuracy metric for classification.

An issue came however, again, with data. We ended up combining the only two datasets we were able to find and get to work successfully, the GTEA dataset [4] and the Hand Over Face dataset but together the data was pretty small with under 15K images. The images in the training data came in pairs of the image itself as well as the ground truth for the image mask, but these images were not as suitable for ASL, as we will come to discuss more in the results. To briefly describe the issue, the dataset mostly came in

[2] S. Bambach, S. Lee, D. J. Crandall and C. Yu, "Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions," 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1949-1957, doi: 10.1109/ICCV.2015.226.

[3] Urooj, Aisha, and Ali Borji. "Analysis of hand segmentation in the wild." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[4] Yin Li, Miao Liu, James M. Rehg, In the eye of beholder: Joint learning of gaze and actions in first person video, ECCV, 2018

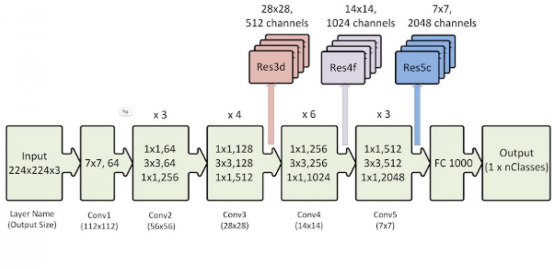


Figure 2: Segmentation model architecture (ResNet50)

first-person captured viewpoint of hands, which is not how our ASL testing data would be presented.

We had originally wanted to use the EgoHands dataset but we could not figure out how to extract the image masks from the images as the MATLAB code they gave us to accomplish that did not seem to do what it intended to do. If given more time and MATLAB expertise, we are sure that the inclusion of this dataset would have benefitted our results greatly, as the hand data from this dataset seemed to fall more in line with the hand data we would present as ASL data.

Overall, we found that training our segmentation model took very long (5-6 hours per epoch) due to how many classifications we must make per image (resizing to an image size of 384x288 yields around 110592 classifications per image). We were running out of GCP credits and were only able to train for 1 epoch, also due to fear of overfitting, as we will go into detail later.

4 Web Application Interface

We won't go too much in detail about this portion because it this is not a web app class. That being said, we used Flask as a backend to periodically classify the images sent to us from our React frontend. We encountered some annoyances loading in the models into the backend server and receiving the images from the frontend correctly but we were able to fix them pretty briefly. We created the ability to take snapshots and display the foremost confident result from classification as a card (but we were not able to connect this with the segmentation model due to other complications and its poor performance).

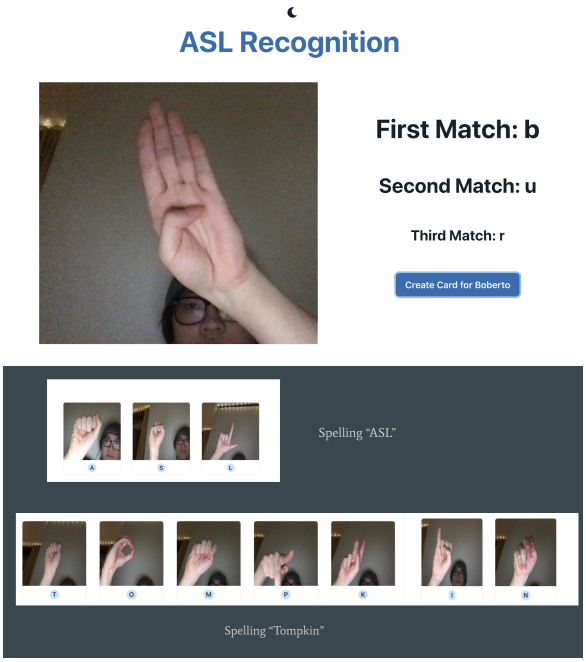


Figure 3: Web app interface and classification demo

5 Results

Github repo: https://github.com/wchen777/ASL-recog

5.1 Classification

When we originally tried to build our own architectures, our model never reached above 4% accuracy. This is no coincidence: randomly selecting 1 out of 26 alphabet letter is around 4%. We weren't exactly sure why our models failed, though. Our classification model, once we switched to deep transfer learning with VGG16, yielded pretty decent results. We were able to achieve around **87% on our validation/test set** and around **88% on our training set**.

In Figures 4 and 5, we can see the epoch loss and epoch accuracy respectively. One quirky thing from the Tensorboard logs showed a weird extra spike/line when first training, which may have been an artifact from a previous training session that was stopped. Regardless, we trained for around 27 epochs until we noticed that the model began

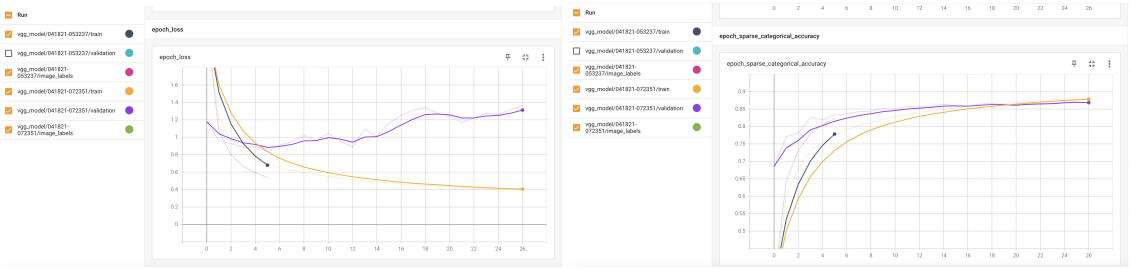


Figure 4: Classification training *Left:* Epoch loss *Right:* Epoch accuracy

overfitting, as we assumed. Examining the epoch accuracy graph, we notice that the validation accuracy (purple) begins to fall at the very end, below the training accuracy, while the loss continued to increase, signifying overfitting. We promptly stopped training our model and used our saved weights.

Our model was even able to detect very minute nuances between signs (albeit it would not detect these with perfect accuracy nor all the time). This is best exemplified in Figure 3. The signs for A and S are very similar, yet the model was still able to tell the difference. The same goes for T and M, where only the thumb placement is different. Even in examples where the face is showing, the model was able to ignore this. We are pretty pleased with these results for classification. Things we could potentially consider in the future could be training an SVM classifier instead of an MLP head and seeing how that changes our results, our using deep transfer learning with another model. We could also even looking at using depth images or detecting a bounding box around the hand to further improve our classification.

5.2 Segmentation

On the other hand, our work in segmentation did not go as smoothly.

The first issue as we mentioned previously came in the fact that our data was not commensurate with the problem we are trying to solve. The GTEA hands mostly had data that came in first-person perspective hands rather than a third-person viewpoint that our data required. This meant that the data from the hands were mostly the backs of the hands or contained both hands, which did not fall in line with what we were trying to segment.

Another challenge we faced was during training. Because we did not have that much data and we were dealing with ResNet50, a considerably large network, we were deathly scared of the looming existential threat of overfitting. It did not help that each epoch took around 5-6 hours, so we were not able to save checkpoints until after a long epoch was finished. Unfortunately, due to lack of GCP credits and time constraints, we were only able to train for 1 epoch due to some issues with Pytorch failing to save our checkpoints after an epoch. This ended up being more than enough.

Although it was only for 1 epoch, the deep transfer learning was able to achieve around a **57% IOU score and a 98% pixel accuracy score**. These results were misleading, however. Although the model performed well on our training set, it did not perform well to when we tried to use it to segment ASL hands.

The combination of the above issues resulted in the following in Figure 5. In the right, using a training example from our dataset, the model was able to segment the hands perfectly. This is unlike in the left, where an example of signing a letter created faulty results. There are many reasons for this. As mentioned previously, the examples from the training set, like the example on the right, contained back of the hands which were smooth in texture, unlike the ASL signs which exposed the palms and created wrinkles in the hands which most likely threw the model off.

It also did not include many images of different skintones, and images of ”darker” hands only were due to differences in lighting. This meant that often times the model would only be able to find the lighter or smoother parts of the data we provided, exacerbated by the fact that our team was all POC. We did not see this however as an end-all-be-all issue however. We were confident that

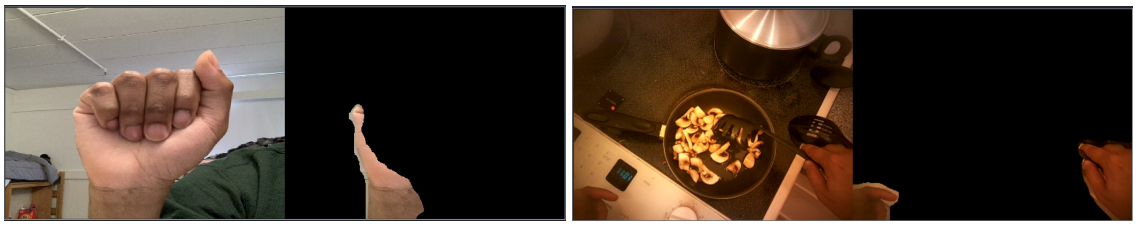


Figure 5: Segmentation overfitting *Left:* Robert signing an A *Right:* Training example from GTEA dataset

with more and better data these issues would be resolved. Unfortunately we simply did not have the GCP credits nor the time to invest any further in the project than we would have liked to, especially due to the shortened semester.

We would've liked to capture a segmented hand instead of the full image as shown in the cards in Figure 3, but due to the performance of the segmentation model, this would've detracted from our application. Overall, we are definitely confident that with more time, resources, effort, and coffee, we would be able to produce meaningful segmentation results and acquire our Sea Legs as seasoned computer vision students.

5.3 Societal Discussion

1. **Social-historical context**
   ASL has been around for the better part of 200 years. Originally stemming from LSF (or French Sign Language); it's developed into a language all its own. ASL is imperative in deaf communities across America, with America currently boasting over 500,000 people who are deaf or exceedingly hard of hearing.

2. **Stakeholders**
   There are a variety of different stakeholders for this project; The primary ones being those that utilize ASL in their daily lives. Our project would provide an avenue for which they could communicate with someone who doesn't understand ASL: an auto-translator. Beyond that it could be a tool ASL professors could use to help teach. Though in both cases, the caveat remains that the scope of our project is primarily letters, leaving a whole realm of the language currently untouched. A major benefit of the project would be convenience. Providing the basic capabilities of a translator for the everyday situations where acquiring an in person one could be a hassle.

3. **Outside research**
   The research helped to ease our worries when it came to the project. This is due to the fact that so many researchers are having difficulty and consistently pushing forward into ASL recognition with deep learning currently. Thus knowing the fact that it's such a recent topic of interested eased our nerves when our model wasn't completely perfect. Alongside that it was from a research paper that we initially obtained the idea of looking into segmentation as an additional avenue of our project.

4. **Civil rights**
   As far as I can currently tell, I'm unsure on how our project's civil rights or liberties be affected. The influence of the project much more changes things as a cultural or ease level, but due to us not really keeping any data on and we don't have major sway to change anything at a civil rights level.

5. **Biases**
   As mentioned throughout this report, our segmentation dataset does run into the eventual problem of being too white, which is the tradeoff we have to endure for achieving a larger dataset. On the other hand due to our entire group being POC, the data that we created as a group highlighted the biases that were present in the dataset. It was also not able to detect the different textures in the palm or front of the hands as the hands in the dataset mostly included the back of the hands which are much smoother.

6 Conclusion

The purpose of our project was to provide more apparatus through which ASL can be translated. This is especially important as translating spoken language is becoming increasingly easy in the world in which we live. Even with our limited knowledge of state-of-the-art computer vision techniques, we were able to create a proof-of-concept tool that can help two different groups of people.

We were able to create and train a pretty effective model that can classify images of signed ASL letters into their language counterpart in a live feed. Although improvements could be made to this part of the model, we are satisfied with the result. The cherry on top is a working front end to help us cleanly display the fruits of our efforts.

For our experimentation with the segmentation aspect of our project, although we weren't able to get a deployable or even fully working model due to the constraints as we outlined above, we learned quite a bit of the difficulty in deep learning especially in its sensitive reliance to data.

Overall, the project was a great learning experience for us who had very little deep learning and computer vision experience out in the wild. The impact going forward, realistically, will not be abundant. We are simply a few students trying to learn the dark arts of deep learning and computer vision, and our work, the little bit that we were able to accomplish, here is nothing that hasn't been done before to a much better degree. Hopefully though, the greater impact is continued effort into live translation of sign language to continue lowering the language barrier.

Appendix

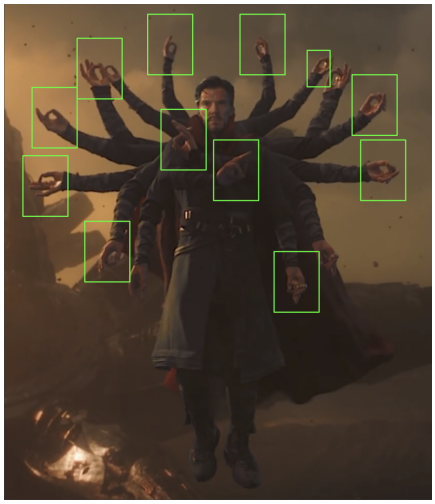
Team contributions

Will Chen Deciding overall project goals and direction, researching outside sources, creating architecture of classification and segmentation model, training both models on GCP, creating backend and frontend

Erick Lerena Forgot about the project entirely. Was watching WandaVision instead of ComputerVision. Just kidding, he ensured the accuracy of all ASL data and created validation data. Helped user test model and researched segmentation methods.

Robert Bush Wrote the progress report, alongside this lovely final project report. Got a crash course in front end, and was taught the absolute basics of React as we went along.

Kenny Ihenatu I helped in gathering the ASL alphabet datasets for the program and assisted in training the model.



Team Godhand signing out.