Implementing Facial Feature Segmentation to Aid American Sign Language Recognition

Helen Chen, Grace Cuenca and Nicole Meister

1. INTRODUCTION & MOTIVATION

According to the World Health Organization, around 5% of the world is Deaf or hard of hearing (HH) [1]. For the Deaf and HH communities within America, American Sign Language (ASL) is the main form of communication [2] whereas hearing people use spoken languages as their primary form of communication. As such, it is easy to see how there can be a difference and difficulty in comprehension between the hearing and the Deaf. Current well-known assistive technologies include products such as text-to-speech software and image captioning. However, even these technologies operate under the impression that the user can read and speak a spoken language such as English. In fact, a common misconception is that ASL is a signed version of English, when the two languages operate under entirely different grammars and vocabularies [3]. It has been noted that approximately 50% of the deaf students in the United States have a reading comprehension below that of the average American fourth grader upon graduating high school [4]. Because of this, written English cannot be a reliable bridge between the hearing and Deaf communities. Computer vision is a potential candidate in solving the problem of sign language recognition (SLR) and increasing accessibility for the Deaf and HH communities. There has been extensive research done on facial and gesture recognition, both of which can be used to develop algorithms and models for SLR. In particular, our paper focuses on how including facial expression features, rather than simply focusing on the handshape in a gesture, can improve or worsen a convolutional neural network's (CNN) ability to accurately detect signs. This is due to the fact that a signer's facial expressions can convey important grammatical information, intensify a sign and more [5]. The five main components of a sign in ASL are handshape, location, movement, palm orientation, and non-manual markers (including facial expression) [6]. As an example on the importance of facial expression in practice, observe the following ASL signs.

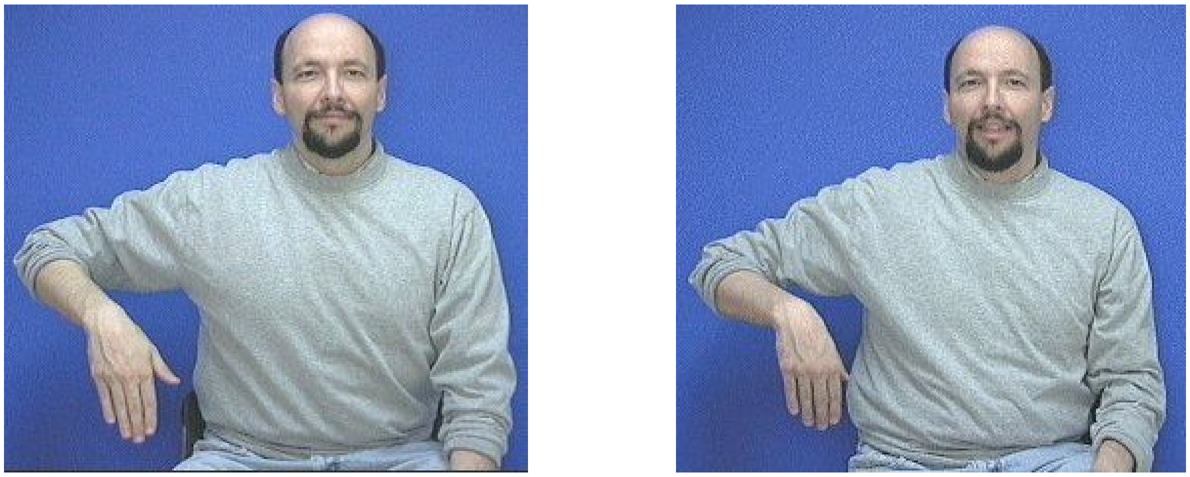


Figure 1. ASL signs for LATE (left) [7] and NOT-YET (right) [8]

The two signs LATE and NOT-YET are minimal pairs in ASL, meaning that they differ by a single parameter. In this case, it's the facial expression that sets these two signs apart.

2. RELATED WORK

A substantial amount of work using both deep learning and classical methods has been done in the SLR sphere. Nyaga et al in *Sign Language Gesture Recognition through Computer Vision* [9] introduces promising results through user-testing and post-experiment reviews. Their experiment involved testing 10 students in the 10th grade who had been attending a school for the deaf and signed in South African Sign Language. In their experiment, they captured videos of students fingerspelling[1] the numbers one, two, three and four. However, there were substantial limitations in the experiment because their video captioning setup required that the signers must sign within a region confined in a rectangular box in order to accurately track motion. In addition, signers generally use their entire body to communicate and so simply tracking the hand reduces this form of expression to the motions of a signer's hand. Ye et al in *Recognizing american Sign language Gestures from within Continuous videos* proposes a hybrid model to detect ASl through a combination of 3D convolutional neural networks (3DCNN) and a fully connected-recurrent neural network (FC-RNN) [10]. This hybrid model allowed them to extract both temporal (sequential) information as well as spatial information from videos and greatly improved ASL detection and accuracy. However, this model also does not take into account facial expressions when classifying signs. Finally, the paper that most heavily influenced our model's design was Bantupalli et al's *American Sign Language Recognition using Deep Learning and Computer Vision* [11]. The paper proposed a

[1] Fingerspelling is spelling out words using the designated signs for the alphabet rather than signing them. From https://www.lifeprint.com/asl101/fingerspelling/fingerspelling.htm

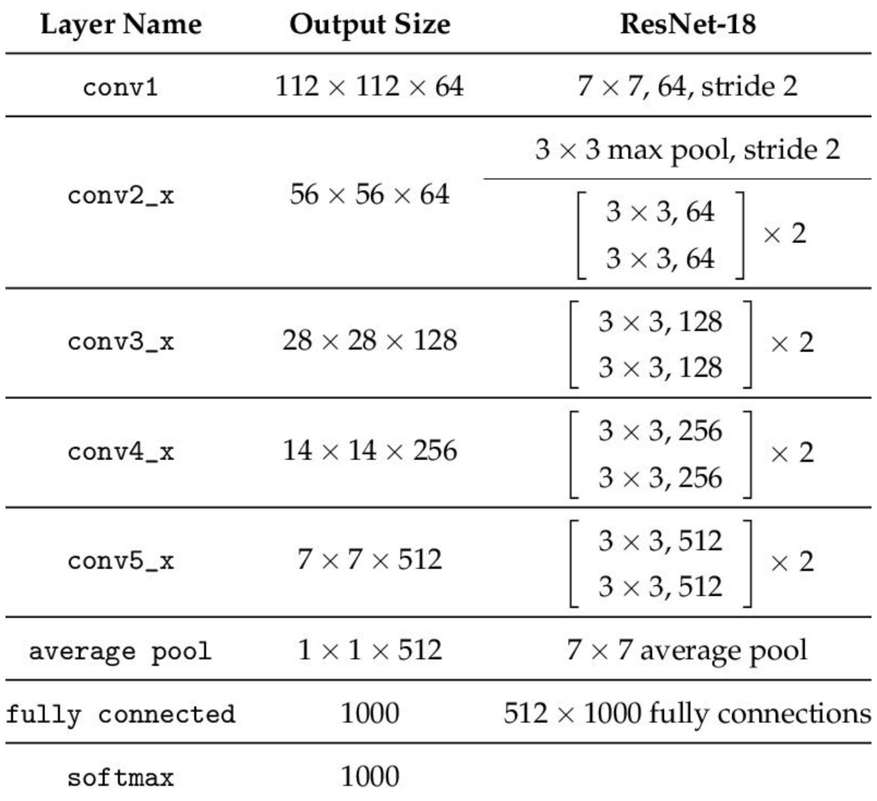
CNN and Long Short Term Memory (LSTM) architecture combination in which they use a CNN to extract the features from image frames and then feed the extracted features into an LSTM to classify the gesture. Their paper concludes by mentioning that their model degrades significantly when faces were included in their frames, meaning that for their dataset, they cropped the image such that the face was not included.

3. GENERAL APPROACH

A CNN is a Deep Learning Algorithm that takes in input images and through the training process, the model learns to not only make classifications, but also learn to extract features of importance from the input images. Unlike more classical methods (HoG, Viola Jones, etc.), CNNs are different in that we don't use a manual algorithm to derive image features and feed these features into a model for classification and the model simply trains on these selected features, but instead in a CNN the model learns to extract these features on its own through the learning process. This helps to improve classification in that we are no longer bound by the feature extraction process. For our project we decided to use a pre-trained CNN for the feature extraction part since we wanted our model to have some capability already for extracting features. We had to then finetune this model to our data since most of the pre-trained models are trained on ImageNet which is a very different dataset from ours. Moreover, many previous works that we came across also used CNNs which further motivated us to take this approach of using deep learning instead of more primitive feature extraction methods.

A Long Short Term Memory (LSTM) network is a special type of recurrent neural network (RNN) architecture that has the capability of learning long-term dependencies. In other words, it is able to "remember" information for a long period of time and use the information that came earlier in a sequence in conjunction with the later information to learn and make predictions. We used a LSTM for our model because signs often may be a combination of different gestures so we wanted our model to have the capability of using all components of the sign within a gesture sequence in order to make an accurate prediction for the class of the sign. This concept of time that is present in LSTMs is necessary when classifying things such as gestures [12].

In order to build our CNN + LSTM architecture, we used PyTorch because of the abundance of documentation on deep learning models as well as its usability with Python. Our approach can be split into two main parts. First, we fine-tuned all the layers of the Resnet18 model that was pre-trained on ImageNet with our dataset. We chose Resnet18 as the base CNN model because we felt that it was not too deep or complex of a model as the other models we had considered, therefore it would've been less likely to overfit on our small amount of data. In addition, Resnet was also used in one of the prior works that we found for classification of Chinese Sign Language and the results they had appeared promising so we decided to use Resnet as our baseline CNN architecture [13].



Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

Figure 2. Resnet-18 Model Architecture used in implementation [14]

We extracted frames from the videos that we selected from ASLLVD and then a single frame was manually chosen to “represent” the sign that was signed in the video. Then, these single frame representations of the videos were used to finetune the pretrained Resnet model. For ease of future description, this part of our approach will be referred to as the static sign CNN training phase. The next part of our approach was training a CNN + LSTM architecture on a sequence of images. We used the weights of our Resnet model trained on static images as the weights of the CNN + LSTM architecture (for the CNN component) and then trained this second model on sequences of three images. For this second component of our approach we focused on changing the dataset in a couple of different ways to see how our model would perform in comparison with our baseline model of the pretrained CNN + LSTM on regular image frames.

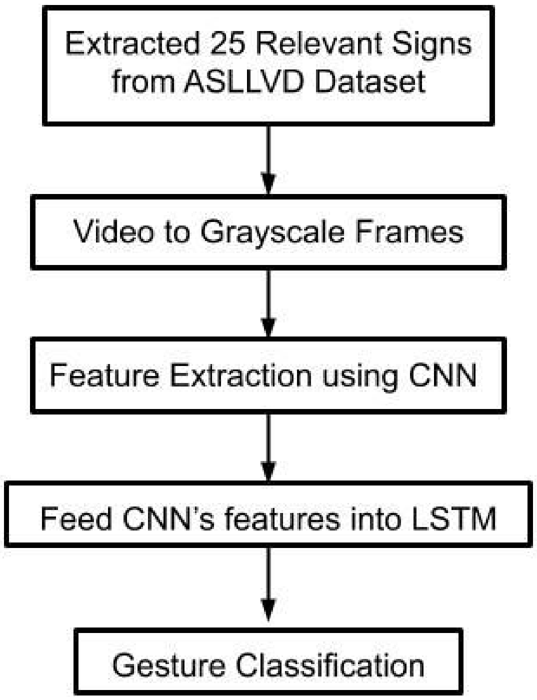

Extracted 25 Relevant Signs from ASLLVD Dataset
Video to Grayscale Frames
Feature Extraction using CNN
Feed CNN's features into LSTM
Gesture Classification

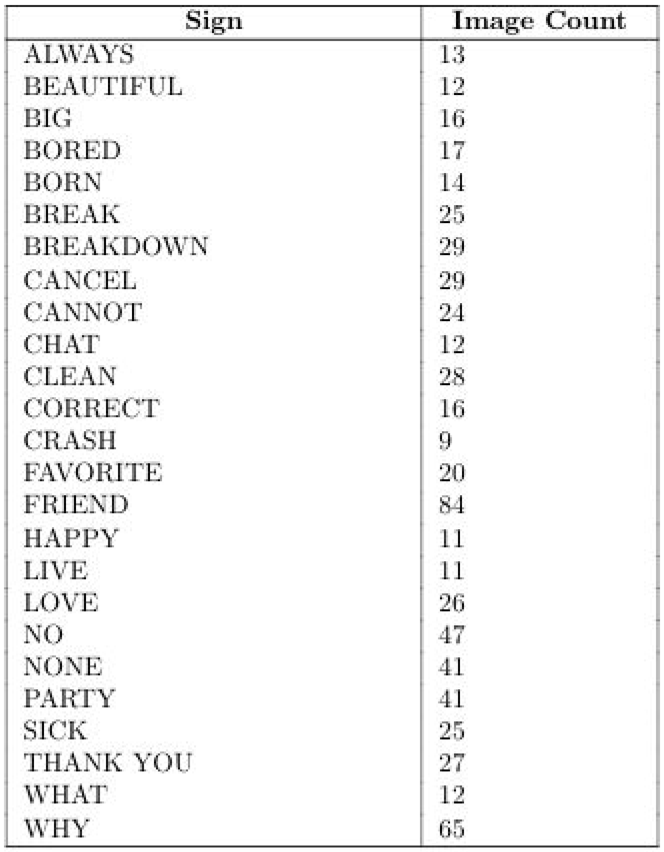
Figure 3. Summary of Implementation

4. DATASET

The dataset we used comes largely from the Boston American Sign Language Lexicon Video Dataset (ASLLVD) [1]. The original dataset consists of more than 3,300 ASL signs, each produced by 1-6 native ASL signers. From this dataset, we picked out 25 signs that either contain more explicit usage of facial expressions in the gesture performance or have a significant amount of data for training purposes. The data retrieved were in the form of short video clips that contained only the sign of interest segmented out from an original longer clip where signers are signing an entire sentence containing the target sign. Each class of signs had varying numbers of videos as well as a varying number of signers.

We also took videos of the same classes from the WLASL, which is the largest video dataset for Word-Level American Sign Language (ASL) recognition [2]. It contains 2,000 common different words in ASL and is also a combination of videos from varying datasets. We combined the videos from the two datasets because in both datasets, the signers are in the center of the video and also perform the single sign. However, the WLASL contains videos that are slightly longer than the ASLLVD since there are some static frames where the signers are not moving in the beginning and also because these signs are not performed within the context of sentences. In addition, since WLASL itself is a combination of different datasets, we felt that it would be appropriate to merge the relevant words of the two datasets for additional data. In order to account for these differences with the ASLLVD, since WLASL had static shots in the beginning and end of the videos, we removed a majority of the corresponding frames from the beginning and end that were not related to the sign performance.

We preprocessed all the data collected by splitting all of the videos into individual gray frames. Upon loading the data into our models, the frames were resized and center cropped to the size 224 x 224 to be passed into our model.



Sign	Image Count
ALWAYS	13
BEAUTIFUL	12
BIG	16
BORED	17
BORN	14
BREAK	25
BREAKDOWN	29
CANCEL	29
CANNOT	24
CHAT	12
CLEAN	28
CORRECT	16
CRASH	9
FAVORITE	20
FRIEND	84
HAPPY	11
LIVE	11
LOVE	26
NO	47
NONE	41
PARTY	41
SICK	25
THANK YOU	27
WHAT	12
WHY	65

Figure 4. The table of signs (classes) and their respective image counts

DATA AUGMENTATION

To prevent our model from overfitting due to a lack of data, we added on data augmentation during the training process. Data augmentation was added via an image augmentation package called imgaug [15]. This package has over 60 image augmenters and augmentation techniques including affine transformations, contrast changes, gaussian noise, dropout of regions, hue/saturation changes, and so forth. For simplicity and consistency across our models, we focused on 3 main augmenters: coarse dropout, gamma contrast, and image flipping. When a sequence of images is read in for training, we calculate a probability that would determine whether or not this sequence of images would get augmented. If it does, then we apply coarse dropout (dropout amount varies each time, but is consistent across one input sequence), gamma contrast (contrast amount also varies each time but remains consistent across one input sequence), and flip (with probability 1). By adding this data augmentation, we hoped to prevent our model from overfitting as it would now be able to see variations of the original images and hopefully be able to learn something different from these altered images that would aid in sign classification.

DATA FOR STATIC SIGN CNN MODEL

For the static sign CNN training phase, **we filtered through all the image frames for each video and selected one image per video sequence that represented the most distinct aspect of the sign**. These images were converted to grayscale.

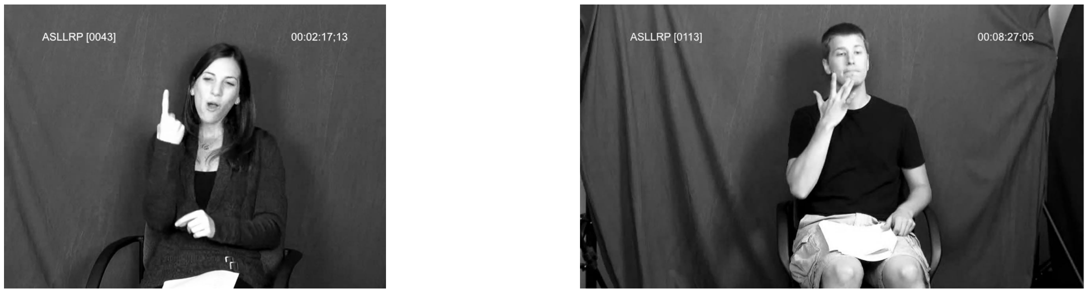


Figure 5. Sample signs from training set ALWAYS (left) and FAVORITE (right)

DATA FOR CNN-LSTM MODEL

For our CNN-LSTM model, we split the videos from our raw data (WLASL + Boston Dataset) into gray image frames to be passed into the model as an image sequence of gestures. Since each video had a varying number of frames **we created a data loader that would choose three consecutive frames each time as input for our model**. These 3 frames were then fed into the model as a single image sequence input to be trained on the CNN-LSTM model. While our final CNN-LSTM model was trained on all 25 classes, we also had an intermediary dataset of just 6 classes trained on the classes: BREAK, BUY, CLEAN, NO, NOT, WHY.

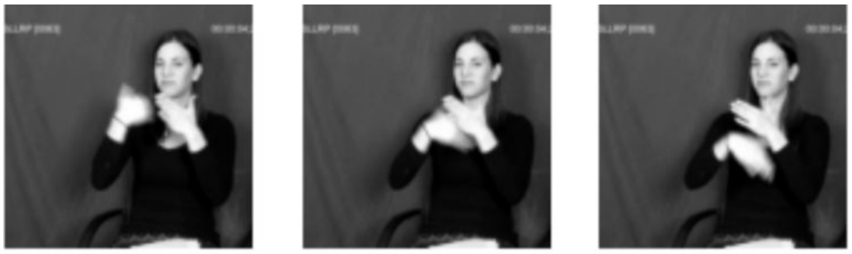


Figure 6. Sign frame sequence from training set- BORN

DATA FOR SEGMENTED FACES + ORIGINAL IMAGES

In order to add particular emphasis on facial expressions, we created a new dataset from our original data that **concatenated the image and the segmented face within the image together as the input**. We first used a face-extractor script that utilized the Face Detection Neural Network from OpenCV [16] to extract the faces from each frame. Then the original image was center cropped and concatenated side by side with the corresponding face of the image. Three consecutive frames were passed in as one input.

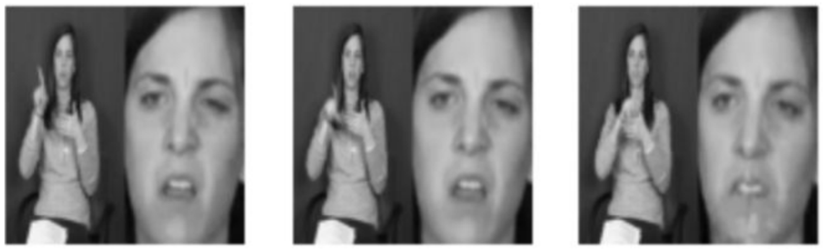


Figure 7. Sample sign image sequence FACE+IMAGE from training set- CANCEL

DATA FOR SEGMENTED HANDS

The ability to perceive the shape of hands is a vital component in sign language understanding as hands gestures and shapes communicate words. To make use of this information and to aid our model in focusing on the hand positions, we **extracted images from our dataset that only contained the left and right hand**. To accomplish this, we leveraged Google Mediapipe's hand pose representation [17].

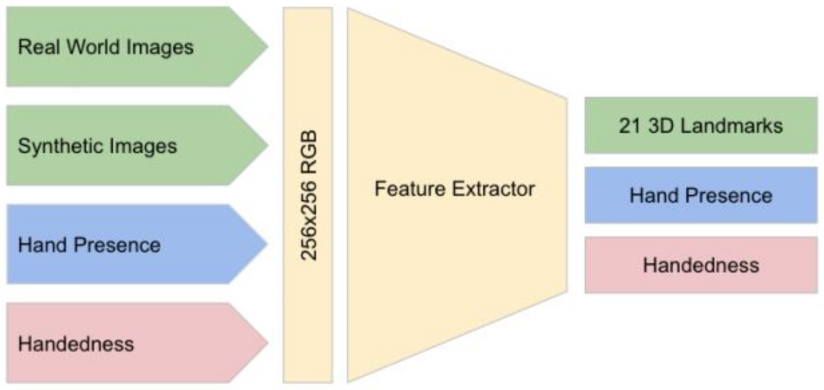


Figure 8. Mediapipe's hand pose identification architecture [17]

Mediapipe hand's pose identification works by running a palm detection model over the entire image and then applying a hand landmark model to produce 21 coordinates corresponding to the hand pose. We use the coordinates of these 21 key points to crop the image (with 50 px padding) and store an annotated hand image and unannotated hand image of the left and right hand. Later, these images were concatenated with the extracted face images.

a)

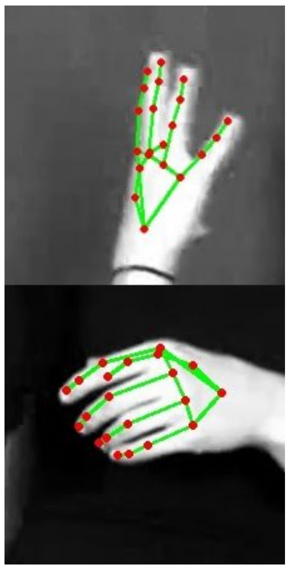
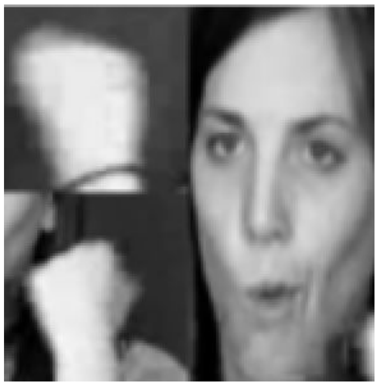
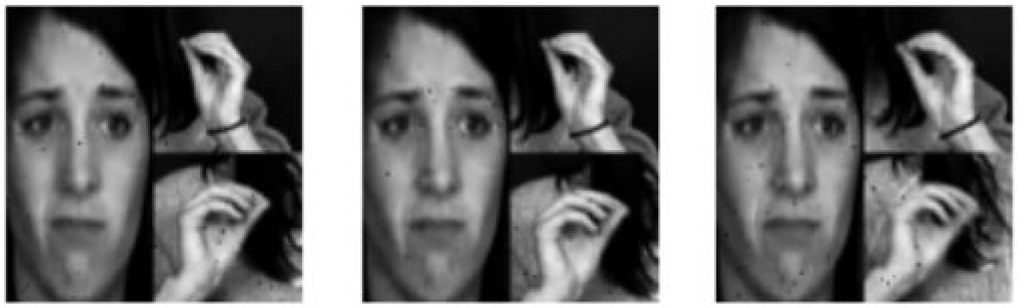
b)

c)

d)


Figure 9. (a) Example of unannotated extracted L&R hand pose for sign “BEAUTIFUL” (b) Example of annotated extracted L&R hand pose for sign “BEAUTIFUL” c) Concatenated Hand and Face Images for sign “FIGHT” (unannotated) d) Sample sequence of 3 frames of hand and face images fed into the model

5. DESIGN AND IMPLEMENTATION

STATIC SIGN CNN

The static sign CNN was trained for 50 epochs with a batch size of 8 and learning rate of 0.00001 and weight decay of 0.0001 using the Adams optimizer. The last fully connected layer has a dropout of 0.5 to prevent overfitting. The static sign CNN was constructed in order to improve the Resnet18 model’s weights for the purpose of gesture and facial feature detection. One flaw with the static sign CNN was the lack of data. Unlike the CNN + LSTM training that was done on sequences of frames, this

CNN was trained on single images, with one video in our dataset mapping to exactly one image. Thus, there was definitely a lack of data when training this initial model. However, it was simply seeking to finetune the weights of Resnet18 that was pre-trained on ImageNet and was not our ultimate classifier.

STATIC SIGN CNN (PRETRAINED) + LSTM (25 Classes)

This is our baseline model trained on all the 25 sign classes. It was composed of the Resnet18 CNN architecture pre-trained on ImageNet then fine-tuned on the static signs combined with an LSTM architecture for the classification of the input sequences. **The input sequences were three consecutive image frames, marked with the same label, that were passed through the CNN component of the model first to extract features from the images and then the accumulated features of the sequence were passed through the LSTM for a sequential classification of the input.**

To extract the CNN's features, we fed the individual images through all the layers of the CNN except for the final fully connected layer that typically makes a prediction from the features. Instead of feeding the CNN's features into a fully connected layer, we concatenated the features from the 3 images frames together into a torch stack (a method to concatenate a sequence of tensors) and fed the features into the LSTM.

By incorporating the LSTM architecture, we allowed our model to have the capability of using information from previous frames in order to help it more accurately predict the class of the entire sequence. Only the classification from the last input image of the sequence was used as this final classification would likely be more accurate since it would be making this prediction via accumulated knowledge from earlier frames.

We initially encountered overfitting issues, resulting in the loss plateauing relatively quickly. Through this step in our implementation, we employed various ways to resolve the problem of overfitting specifically hyperparameter tuning (regularization), different optimizations methods (adams, SGD), data augmentation, and other CNN models (i.e. Resnet50, Alexnet). Data augmentation turned out to be one of the more useful solutions to solving our problem of overfitting.

CNN (FINETUNED RESNET18 PRETRAINED) + LSTM (25 Classes)

We experimented with several different methods of CNN feature extraction to see which method gave us better results when combined with our LSTM. **In addition to using the weights from a Resnet18 CNN pre-trained on ImageNet and then fine-tuned on static signs,** we also experimented with a Resnet18 CNN architecture pre-trained on ImageNet where all layers were trainable. This allowed us to finetune the weights to be applicable to our sign language recognition task.

STATIC SIGN CNN (PRETRAINED) + LSTM WITH SEGMENTED FACE + IMAGE

This model followed a similar architecture and training procedure to the baseline model with the key difference being **this model was fed image data that concatenated the segmented face and the image frame.** We froze the CNN layers so that they were used solely for feature extraction from the images and then fed the features into the LSTM. With its main difference being the input, we wanted to see how well our model might turn out when we placed an emphasis on the face within the features.

STATIC SIGN CNN (PRETRAINED) + LSTM WITH SEGMENTED FACE + HAND

This model followed a similar architecture and training procedure to the baseline model with the key difference being **this model was fed image data that concatenated the segmented face and segmented hand**. We froze the CNN layers so that they were used solely for feature extraction from the images and then fed the features into the LSTM. With its main difference being the input, we wanted to see how well our model might turn out when we placed an emphasis on the face and the hand within the features.

6. RESULTS

QUANTITATIVE ANALYSIS

OVERVIEW OF RESULTS

Architecture	Data	Test Accuracy (%)
ResNet18 CNN (Static Sign CNN)	Static Signs (1 Frame)	63.24
Static Sign CNN + LSTM (Base Model)	**3 Frames / Sign (Entire Body)**	**94.79**
Pretrained and Finetuned ResNet18 CNN + LSTM	3 Frames Per Sign (Entire Body)	86.53
Static Sign CNN + LSTM	3 Frames / Sign (Entire Body + Segmented Face)	83.55
Static Sign CNN + LSTM	3 Frames / Sign (Segmented Hand + Segmented Face)	61.49

Static Sign CNN + LSTM	3 Frames / Sign (Segmented Face)	25.73

Figure 10. Overview Table of Test Accuracy across variations of our architecture's implementation. (Note: Highlighted result was our baseline model which also had the highest test accuracy.)

STATIC SIGN CNN

For the static sign CNN, the final train accuracy and loss was 95.28% and 0.2571 respectively. The test accuracy and loss was **63.24%** and 1.8103 respectively. Again because of the lack of data, this model showed evidence of overfitting at its final epochs.

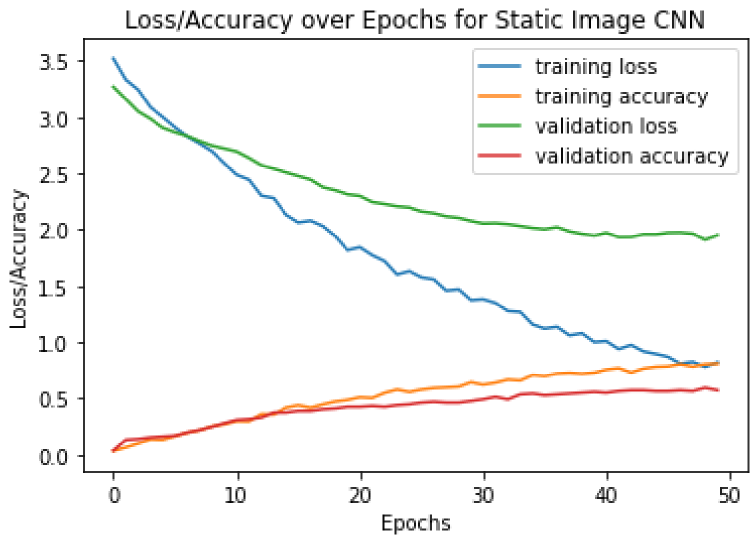
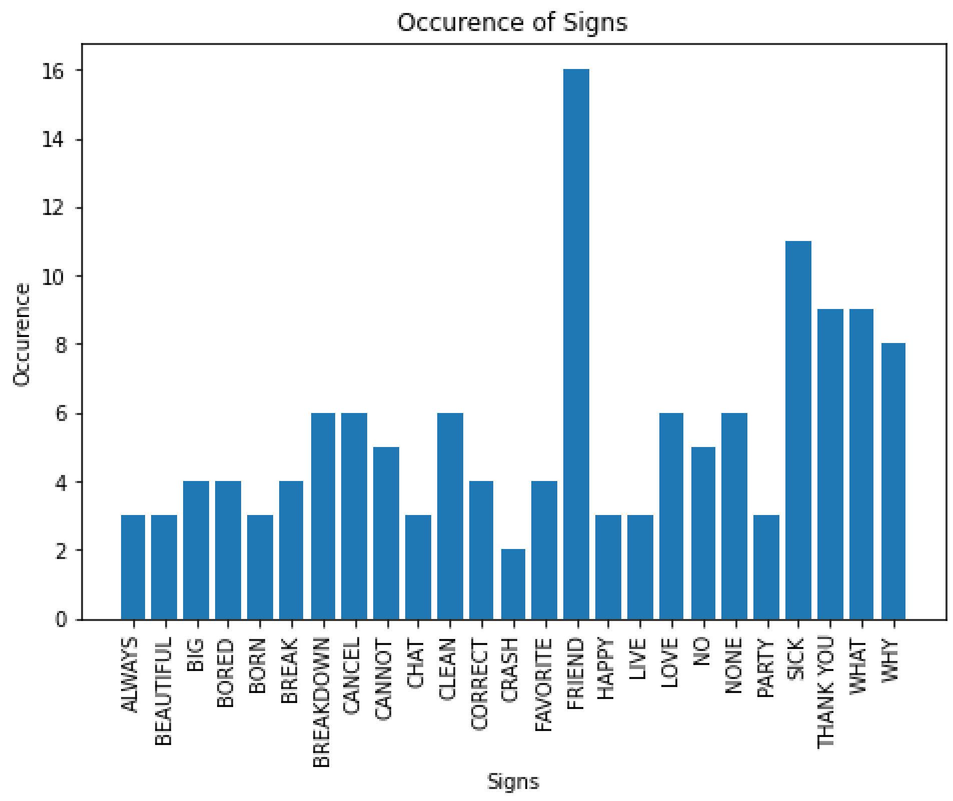


Figure 11. Loss and Accuracy over 50 epochs for the training and validation datasets

It is also important to note that there was a severe class imbalance in our dataset, as shown in Figure 4. With Figure 12 & 13 below, we can observe a trend between higher validation accuracies per sign and the number of images that were used in training for that sign. This is a flaw in our model due to the lack of publicly available annotated ASL videos.


Occurence of Signs
Occurence
Signs
0
2
4
6
8
10
12
14
16
ALWAYS
BEAUTIFUL
BIG
BORED
BORN
BREAK
BREAKDOWN
CANCEL
CANNOT
CHAT
CLEAN
CORRECT
CRASH
FAVORITE
FRIEND
HAPPY
LIVE
LOVE
NO
NONE
PARTY
SICK
THANK YOU
WHAT
WHY

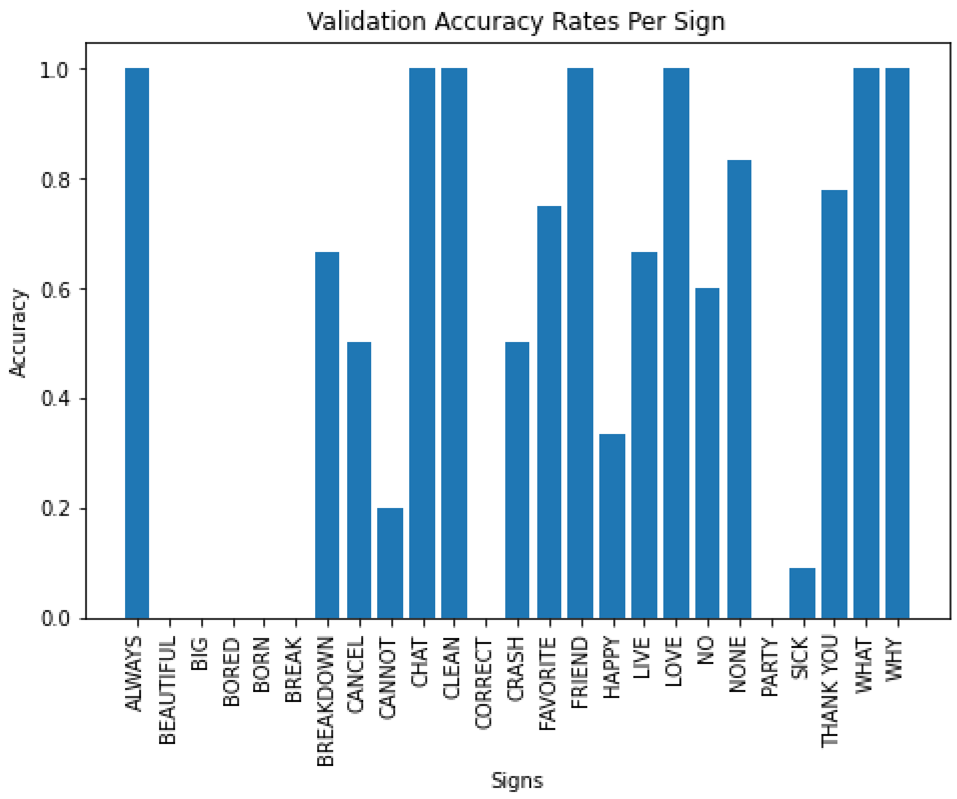

Validation Accuracy Rates Per Sign
Accuracy
Signs
0.0
0.2
0.4
0.6
0.8
1.0
ALWAYS
BEAUTIFUL
BIG
BORED
BORN
BREAK
BREAKDOWN
CANCEL
CANNOT
CHAT
CLEAN
CORRECT
CRASH
FAVORITE
FRIEND
HAPPY
LIVE
LOVE
NO
NONE
PARTY
SICK
THANK YOU
WHAT
WHY

Figure 12 & 13. Occurrence of signs in Validation Set and Validation Accuracy rate per sign in the final epoch of training

CNN (FINETUNED RESNET18 PRETRAINED) + LSTM (25 Classes)

For the CNN that was finetuned with Resnet18 weights pre-trained on ImageNet combined with the LSTM (fully trained), the final train accuracy and loss was 97.51% and 0.085637 respectively. The test accuracy and loss was **70.45%** and 1.174886 respectively. This model showed significant evidence of overfitting at its final epochs and even with data augmentation and hyperparameter tuning, we were unable to achieve better performance on the test set. Thus, instead of fine-tuning the Resnet18 CNN on our inputs that were image sequences, we decided to experiment on a pre-trained static sign CNN as the base CNN for our CNN + LSTM model and freezed these initial feature extracting layers.

STATIC SIGN CNN (PRETRAINED) + LSTM (25 Classes)

For our static sign CNN + LSTM model, the final training accuracy and loss was 94.60% and 0.2686 respectively. The test accuracy and loss was **94.79%** and 0.240 respectively. As shown in Figure 14 below, both our training loss and test loss goes down to 0 throughout the epochs and while the two accuracy curves are relatively close, this simply meant that our model probably had more room for improvement and we could've potentially increased the complexity of our model. However, when we tried more complex models such as Resnet50 on our CNN + LSTM (6 classes) previously, we saw that our model easily overfit on our small dataset and thus, we decided to stick with Resnet18 for simplicity of extracting image features.

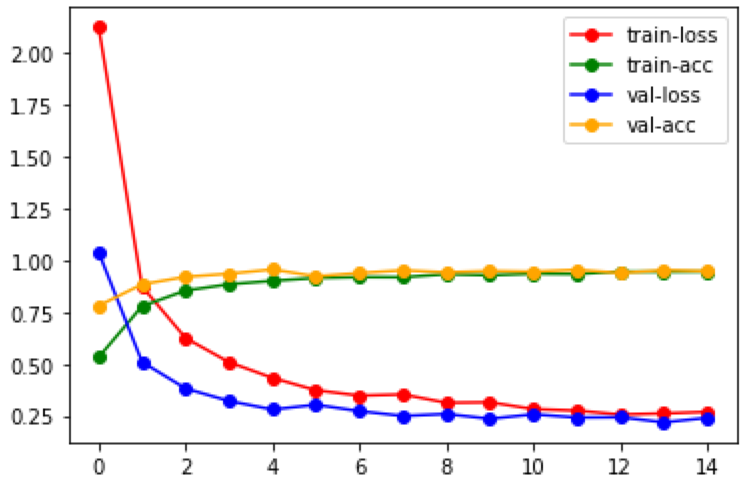


Figure 14. Training/Validation Loss and Accuracy across 15 epochs for CNN + LSTM on Image only

STATIC SIGN CNN (PRETRAINED) + LSTM WITH SEGMENTED FACE + IMAGE

For our static sign CNN + LSTM model using the segmented face plus image dataset, the final training accuracy and loss was 84.16% and 0.583 respectively. The test accuracy and loss was **83.55%** and 0.623 respectively. As shown in Figure 15 below, while the loss did decrease, it eventually plateaued and the same for the accuracies that were increasing consistently. This model performed worse than when we had simply passed in the original image (entire body). However, it can be seen in both that the model has some potential to learn more and be more complex in how the training accuracy and validation accuracy curves are fairly close.

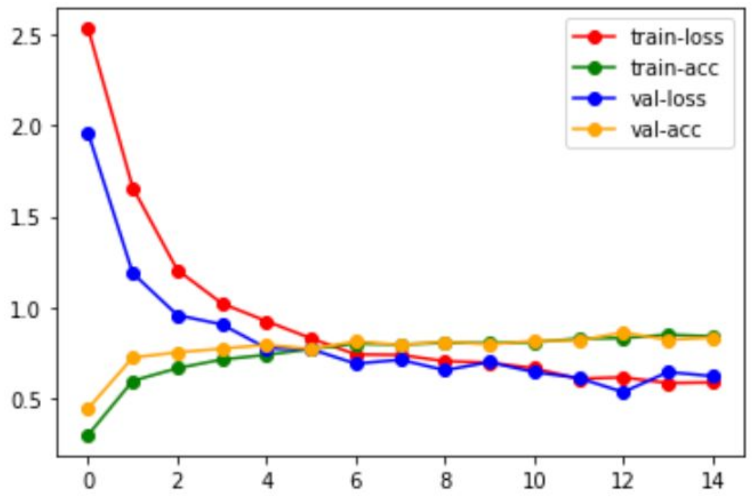


Figure 15. Training/Validation Loss and Accuracy across 15 epochs for CNN + LSTM on Segmented Face + Image Dataset

STATIC SIGN CNN (PRETRAINED) + LSTM WITH SEGMENTED FACE + HAND

When we altered the dataset by segmenting out face and hand and simply feeding these two components into our model, we saw a decrease in accuracy. The final training accuracy was 70.51% and the training loss was 1.0327. The test accuracy was **61.49%** and the test loss was 1.22455. Clearly, our model did not perform better than our baseline model and did not support our expectations that segmenting face and hand might improve classification accuracy. We also experimented with feeding in images that contained the hand pose annotations but this did not improve the model accuracy.

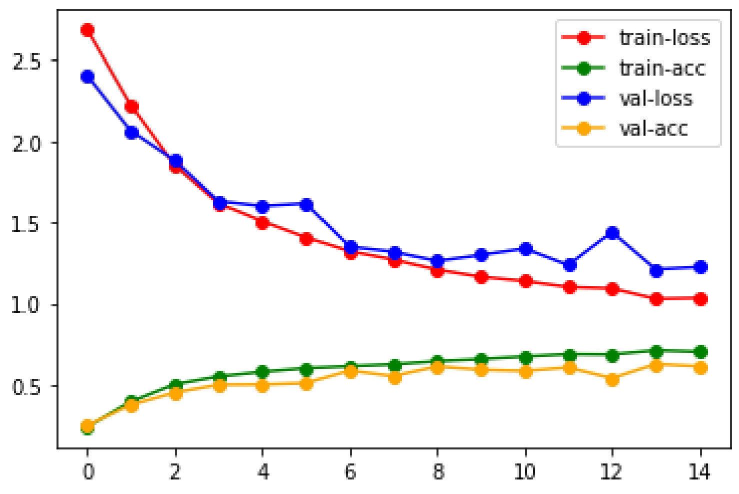


Figure 16. Training/Validation Loss and Accuracy across 15 epochs for CNN + LSTM on Segmented Face + Hand Dataset

STATIC SIGN CNN (PRETRAINED) + LSTM WITH SEGMENTED FACE ONLY

For our static sign CNN + LSTM model using the segmented face plus image dataset, the final training accuracy and loss was 78.28% and 0.6571 respectively. The test accuracy and loss was **25.73%** and 4.6853 respectively. As shown in Figure 17 below, our validation loss during the training process only increased as our model continued to overfit on the input images of faces. It is possible that because so many of the faces likely showed similar facial expressions, our model simply memorized these facial expressions for particular classes and thus performed extremely poorly when only given the faces. However, since this accuracy is not completely 0% and our model did seem to have some small amount of success with using only faces to classify classes. Perhaps there was useful information in the faces for the model but there may have been insufficient distinctions between faces of different classes.

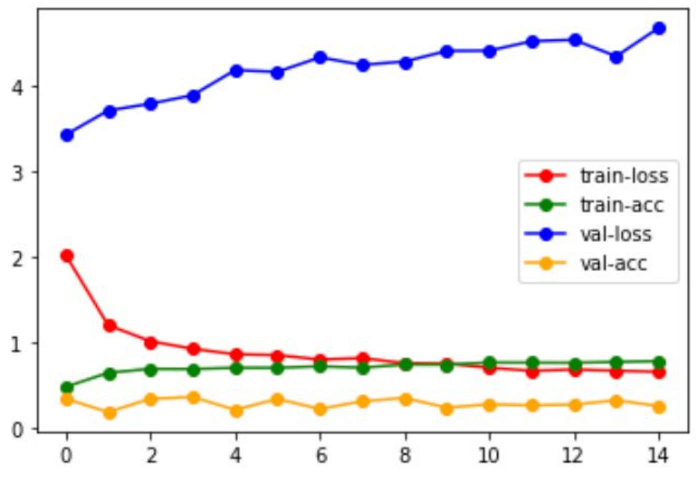


Figure 17. Training/Validation Loss and Accuracy across 15 epochs for CNN + LSTM on Segmented Face ONLY

Our main goal for this project was to improve the accuracy upon adding in faces for sign language gesture recognition so prior to our exploration, we initially hypothesized that the explicit inclusion of facial features would improve the classification accuracy of existing models. As mentioned previously, other works have included faces within the image, but ended up removing them due to a decrease in accuracy, leaving only the lower half of the body. Hence, we initially speculated that if we could segment out the hands and the faces and remove all other noise (the body, background, etc) and place emphasis on these two components, then we might be able to see an improvement in accuracy instead.

However, as shown by our quantitative results, **the segmentation of hands and faces from the original images did not improve our model performance**. The first step in our approach was segmenting out faces and feeding it in with the original image. Our resulting model had worse performance than when we only fed in the entire image. This did not align with our initial hypothesis that putting an emphasis on faces might give our model more clues to narrow down the right class. The reason that this happened was that putting emphasis on the face probably made our model more confused since some facial expressions were similar across different signs. **From this, we learned that obtaining data that is more face-conscientious would be necessary to see potential improvement in the model. Not only that, obtaining data that contain minimal pair signs would also be crucial.** Furthermore, for this project we used quite a number of signs that were different, but it may have been better to potentially use sign gestures that looked similar. Perhaps in doing so, our model may realize that some parts of the sign that are similar is less significant for classification and thus may try to extract features and put emphasis on other parts of the body (potentially the face) to aid in classification.

QUALITATIVE ANALYSIS

The accuracy of the model decreased further when it was fed only face and hand data. To investigate why this was the case, we found that since hand identification is a challenging computer vision task, hands were not detected in the images in some cases. We found that hands in our dataset can often occlude each other (Figure 18c), not be present in the photo (Figure 18b), or occlude the face making it difficult for our hand detection model to locate the hand. We experimented with different methods of dealing with no detection of hands including replacing the hands with the original image (as displayed in Figure 18b and 18c) or replacing the hand image with a blank, black image. We found that replacing the hands with the original image gave us a test accuracy of 61.40% and replacing the hands with a black original image gave us a test accuracy of 26.23%.

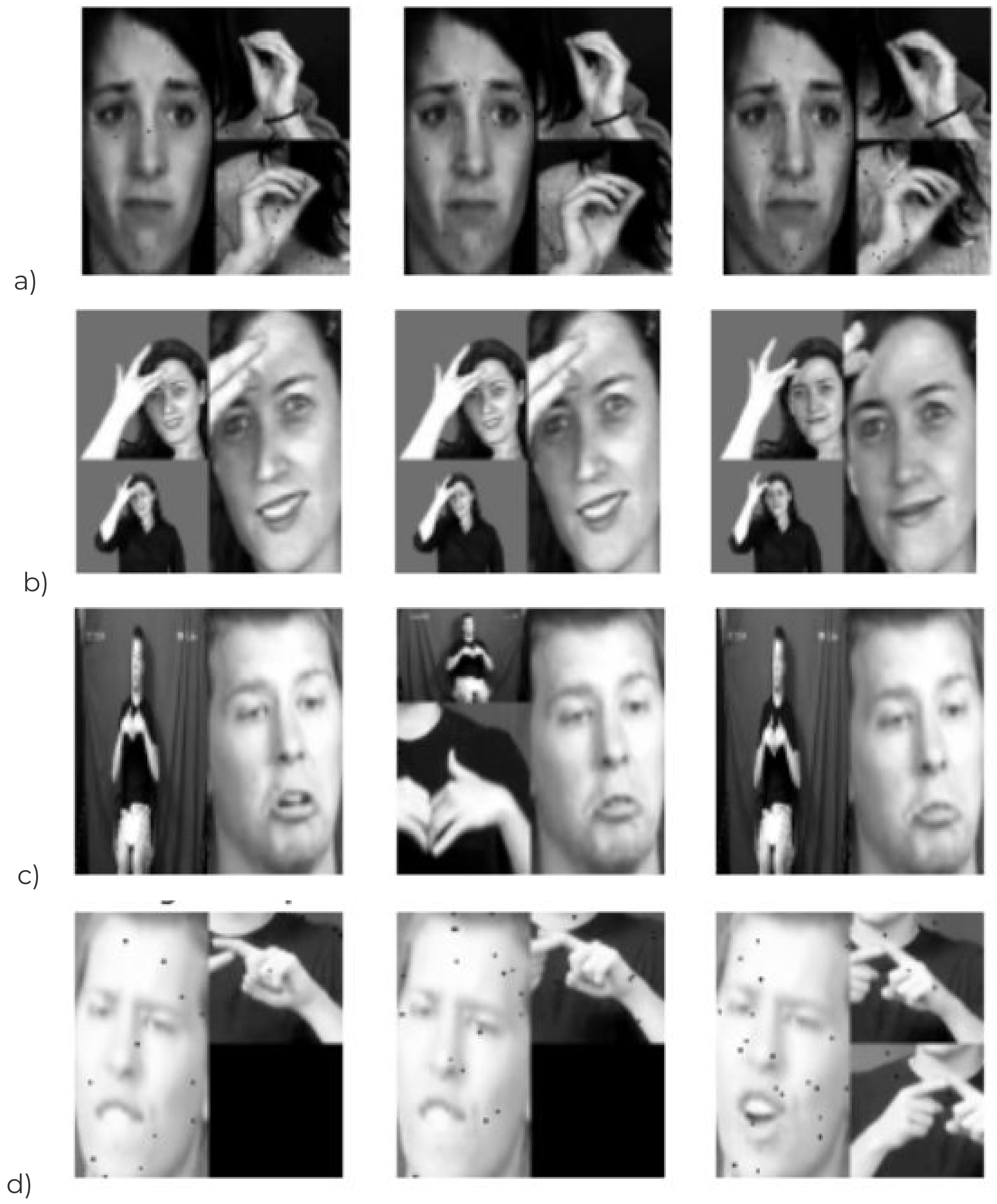


Figure 18. Examples of face and hand images fed into the CNN + LSTM Network. a) An example image where hand identification was successful. b) An example image where hand detection for one of the hands was unsuccessful because it was out of the frame. c) An example image where hand detection was unsuccessful because the hands occlude each other. In the first frame extracted, neither hand was detected. In the second frame extraction, only one hand was detected. In the third frame, neither hands were detected. d) An example image (with data augmentation) fed into the CNN to test out replacing the misclassified hand image with a black

image. However, this CNN did not perform well because rendering black images instead of the image frame results in extreme loss of information.

We can extend our qualitative analysis by analyzing the exact facial expressions made by the signers in our dataset as well. Initially, we wanted to choose our dataset such that it explicitly included minimal pairs (such as those presented in Figure 1 earlier) but were limited by the amount of publicly available data and unfortunately could not be picky with our training and testing set. Thus, we opted to choose signs from classes that had a lot of signs and signs that were more “emotive” such as LOVE and BIG. **However, there was an immense amount of facial expression variation across signers for the same sign.** There was even variation across the same signer. This is shown in Figure 19 below.

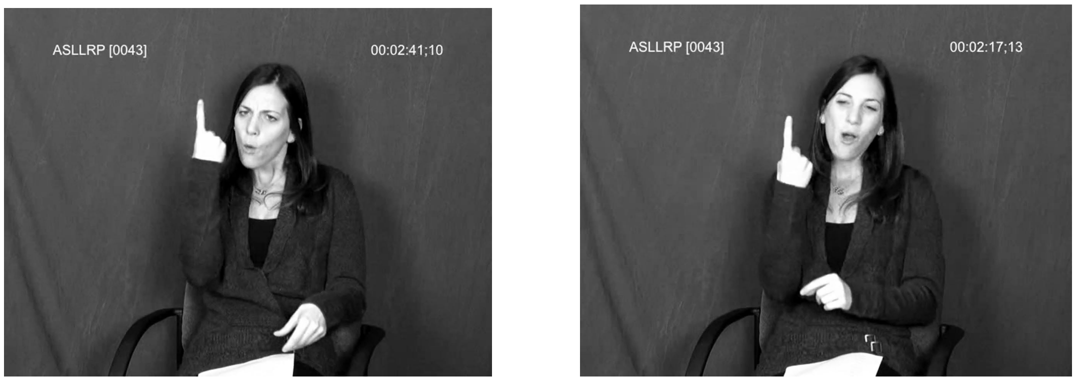


Figure 19. The same signer signing ALWAYS in the context of two different sentences

This difference in facial expression of the sign was most likely due to the context in which the sign was being produced, since these signs were clips of a longer video in which the signer signed a sentence. We took our data in this context because this is the natural way in which signers sign, but perhaps for neutralization it would be worthwhile to assess how our model performs on signs that are signed in isolation, rather than taken from a full sentence. If we wanted our model to function in the way we envisioned and treat the two signs shown previously in Figure 1 as distinct, then our model would necessarily need to classify the two images in Figure 19 as distinct due to the difference in facial expression. This could be the reason why segmenting out the face caused the model to perform worse.

7. DISCUSSION

One notable strength of our system was its **ability to perform well on the grayscale static image frames as input for the static sign CNN + LSTM architecture**. In Bantupalli et al’s paper, they created their own data by using the ASLLVD as reference, but noted that including faces degraded their model significantly. Thus, they trained and evaluated their model on images that depicted the signer from the chin down.

However, **we did notice our model's accuracy degrade when we concatenated the extracted face onto the original static sign image.** This may be due to the variation of facial expressions across signers signing the same sign. This **inserts noise into our model during the training process**. Since our dataset was small, the model may have also learned the subtle facial expression differences that occur across signers signing the same sign and used that as a metric to further distinguish the signs. However, we did not anticipate the difference to be so great that the model would begin to classify signers signing the same sign as different due to differing facial expressions. Furthermore, across different sign classes, **some signers showed the same facial expressions even for different signs**, which is another contributing factor to the loss in accuracy when adding emphasis on the face. When we trained our baseline model on only faces, we saw an immense decrease in test accuracy, suggesting that while facial features did present some information (since test accuracy wasn't completely 0), the faces were more likely to be noise and unnecessary information for the model that actually hindered classification. In other words, when we added emphasis on the faces, this did not help our model narrow down potential correct classes for a particular input and instead may have made the model even more confused.

Perhaps our model had the capability of learning more and we could've made it more complex, but for the sake of comparison with our baseline model, we kept it the same. By changing the dataset from just image frames to input sequences that consisted of images frames plus segmented faces, we saw a decrease in accuracy of sign classification. Although we had hoped to see an improvement in accuracy since facial expressions are likely to aid in sign gesture performance, there are a few reasons why our model did not perform as well with the addition of the segmented faces. Firstly, we saw that **while there was some consistency in facial expressions amongst the signers, this was not always the case for all of the signs**. In addition, some of the facial expressions between different signs were also relatively similar. Hence, with the inclusion of segmented faces, our model would've seen features that were actually more similar across different sign classes, thereby decreasing the accuracy. As noted previously, **there are minimal pair signs that differ by a single parameter, however, our dataset did not contain many of these minimal pair signs as it was difficult to find**. Rather, many of the signs that we saw in our dataset actually had similar facial expressions across different signs and different facial expressions within the same sign, accounting for the decrease in accuracy.

When we fed our model a segmented face and hands, we saw the model performance decrease. This is due to the difficulty of identifying hands as discussed in the *Qualitative Analysis* section and the fact that the position of the hand is crucial to understanding a sign. The location of the hand sign in relationship to the body is crucial to understand the meaning of the sign and removing that information from the input resulted in information loss and thus decreased the accuracy of our model.

8. CONCLUSION AND FUTURE WORK

One of the main challenges in this data was **finding appropriate data**. To improve upon our current implementation and to truly identify if facial expressions can improve model performance, we would need to find data that consists solely of minimal pairs (signs that differ by a single parameter which would be face in this application). This was challenging to find because there is often not enough widely available data. In addition, another major challenge was the inconsistency in facial expressions amongst even the same sign. However, this also ties back to finding sign gestures that rely more heavily and significantly on facial expressions in interpreting its meaning.

In conclusion, we were able to build a model consisting of a CNN that extracts features that are fed into an LSTM that identifies signs from a corpus of 25 signs with an accuracy of 94.79%. We experimented with placing an emphasis on facial features or hand positions by segmenting out the face and hand; however we did not observe an improvement in the model's performance.

For future improvements on this work, we would want to consider **trying out more CNN architectures** and **experimenting a bit more with the LSTM layers as well**. In addition, instead of a CNN architecture which we thought would give good model performance, perhaps we can **try out more classical and primitive methods of feature extraction**, potentially to focus on extracting features more closely related to hands and faces only. In addition, we would also want to try **balancing out our dataset a bit more amongst the different classes or simply gathering a lot more data from other sources**. It would also be crucial to perhaps find or create datasets that contain largely minimal pair signs. Overall, finding or even creating annotated data appears to be one of the more critical steps and challenges for this work.

9. REFERENCES

PAPERS AND WEBSITES

[1] "Deafness and hearing loss"
https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

[2] "American Sign Language"
https://www.nidcd.nih.gov/health/american-sign-language

[3] “Can you Read a Language You Can’t Hear?”
https://www.psychologytoday.com/us/blog/talking-apes/201507/can-you-read-language-you-can-t-hear

[4] Hrastinski I, Wilbur RB. Academic Achievement of Deaf and Hard-of-Hearing Students in an ASL/English Bilingual Program. J Deaf Stud Deaf Educ. 2016 Apr;21(2):156-70. doi: 10.1093/deafed/env072. Epub 2016 Feb 10. PMID: 26864688; PMCID: PMC4886322.

[5] “Non-manual signals used in sign language”
https://www.handspeak.com/learn/index.php?id=158

[6] “American Sign Language parameters”
https://www.lifeprint.com/asl101/pages-layout/parameters.htm

[7] “American Sign Language: LATE”
https://www.lifeprint.com/asl101/pages-signs/l/late.htm

[8] “American Sign Language: NOT-YET”
https://www.lifeprint.com/asl101/pages-signs/n/not-yet.htm

[9] C. N. Nyaga and R. D. Wario, "Sign Language Gesture Recognition through Computer Vision," 2018 IST-Africa Week Conference (IST-Africa), Gaborone, 2018, pp. Page 1 of 8-Page 8 of 8.

[10] Y. Ye, Y. Tian, M. Huenerfauth and J. Liu, "Recognizing American Sign Language Gestures from Within Continuous Videos," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Salt Lake City, UT, 2018, pp. 2145-214509, doi: 10.1109/CVPRW.2018.00280.

[11] K. Bantupalli and Y. Xie, "American Sign Language Recognition using Deep Learning and Computer Vision," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 4896-4899, doi: 10.1109/BigData.2018.8622141.

[12] “Understanding LSTM Networks”
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[13] “isolated & continuous sign language recognition using CNN+LSTM/3D CNN/GCN/Encoder-Decoder” https://github.com/0aqz0/SLR

[14] “ResNet-18 Architecture”
https://www.researchgate.net/figure/ResNet-18-Architecture_tbl1_322476121

[15] “Data Augmentation for Deep Learning”
https://towardsdatascience.com/data-augmentation-for-deep-learning-4fe21d1a4eb9

[16] “Extracting faces using OpenCV Face Detection Neural Network”
https://towardsdatascience.com/extracting-faces-using-opencv-face-detection-neural-network-475c5cd0c260

[17] “MediaPipe Hands: On-device Real-time Hand Tracking”
https://arxiv.org/pdf/2006.10214.pdf

DATASETS

[1] http://www.bu.edu/asllrp/av/dai-asllvd.html

[2] “Dataset proposed in WACV 2020 "Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison"
https://dxli94.github.io/WLASL/

OPEN SOURCE CODE

[1]
https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html - We used this to help construct the initial static sign ResNet architecture

[2] https://github.com/0aqz0/SLR - We used this code to help construct our CNN LSTM and the data loader for our input of image sequences.

[3] https://google.github.io/mediapipe/solutions/hands.html - We used this code to obtain keypoints for hand pose. We modified this code to use the hand key points to crop out only the hand from the dataset and apply 50px padding.

[4] https://github.com/kb22/Create-Face-Data-from-Images - We used this code to detect faces from our image frames. We modified this code to parse through different sign classes.

[5] https://www.geeksforgeeks.org/python-program-extract-frames-using-opencv/ - We looked at this code to understand how we can write code to extract frames from the downloaded videos.

OUR CODE

Link to our code: https://github.com/gracecuenca/COS429_ASL_Recognition