Sign Language Recognition with Convolutional Neural Networks

Arnav Gangal
agangal@stanford.edu

Anusha Kuppahally
akupp@stanford.edu

Malavi Ravindran
mr328@stanford.edu

Abstract

Our paper presents a two-pronged ablation study for sign language recognition for American Sign Language (ASL) characters on two datasets. Experimentation revealed that hyperparameter tuning, data augmentation, and hand landmark detection can help improve accuracy. The final model achieved a test accuracy of 96.42%. Future work includes running the model for a greater number of epochs, tuning the minimum detection confidence parameter in hand landmark detection, further hyperparameter tuning for data augmentation, and additional hand detection bounding box or coordinate methods.

1. Introduction

Effective sign language recognition is an active area of research that intersects both computer vision and natural language processing, with a variety of methods aiming to facilitate communication among the deaf and hard-of-hearing community. This research area can help resolve a communication gap between those who use sign language, and those who do not. The existence of this gap leads to significant barriers in everyday interactions, and reducing it through effective translation models can create more inclusive and equitable spaces for deaf and hard-of-hearing people, as well as improve their quality-of-life.

To explore this application of computer vision, our project focuses on American Sign Language (ASL) character detection using CNNs. The input to our algorithm is static images of ASL character signs, with variation in the dataset coming from the image angle, the image lighting, and the specific subject performing the sign. Then, we use a dual-input CNN to identify key points on subjects' hands ('landmarks'), and output a character prediction based on a combination of these landmarks and the image itself. Our project focuses on the ASL fingerspelling alphabet, including all characters except J and Z, as signing these characters requires motion. We chose this project because it is a current area of research that has many different methods and architectures, which gave us many opportunities to test a range of different models, and compare their performance in the classification task. Additionally, this topic is an interesting example of the intersection between artificial intelligence and social good, and in particular how computer vision models can be deployed to improve the quality-of-life of often marginalized groups.

2. Related Work

2.1. Hand Detection

Within this research area, hand isolation and detection is an important component of sign language recognition. One example of this is [15], which uses Google's hand landmark model [22] to identify hand landmark coordinates, which serves as a second input channel to a CNN.

Another example of this is [19], which uses skin masking, which crops the region of interest (RoI) that only contains the hand, the Canny Edge Detection algorithm to detect the edges of the hand, and extracts features with Scale-Invariant Feature Transform (SIFT) to account for factors like rotation, scaling, etc.

One more instance of hand detection is done by [17], which uses a finetuned CNN model based on the Faster Region-based Convolutional Neural Network (RCNN), which uses a region proposal network (RPN) to predict the bounds where the hand is located, achieving 99.31% accuracy.

Additionally, [8] uses a tree-structured regional ensemble network (REN), which partitions convolution outputs into different regions, concatenates results, and regresses 3D joint coordinates in depth images with end-to-end optimization. All of these papers employ different techniques to detect hands in images, and also use data augmentation.

In particular, [17] excels at these techniques by using the detected hand and then applying various data augmentation techniques, such as 5 crops and adding noise. Adding crops in this way increased the amount of data and made the model more robust. On the other hand, one thing to note about [15] is that this paper did not sufficiently compare results between a single input channel against a multi-headed CNN after adding the hand landmark coordinates. While other papers were formatted as ablation studies, this paper lacked an explanation of how the model was built upon.

2.2. Real-time Robust ASL Recognition

While classifying and translating static images of sign language letters and words is an essential task, there is also a need for robust and scalable real-time ASL recognition models. A noted weakness of existing training datasets is that they often do not contain a variety of skin-tones, making models trained on them prone to failure at inference time when presented with a hand from an ethnicity not seen during training. One strategy to mitigate this is presented in [21], which uses a skin detection algorithm to create a mask for the input image, that works by looking at the colors of the image in terms of luminance (Y), and chrominance (Cb and Cr), a common color space in video compression. This mask was used to remove all parts of the image that were not identified as skin, and their model was able to achieve 94.7% test accuracy with a downstream classification model based on AlexNet.

Similarly, [12] proposes a multiple-stage pipeline to improve model robustness with reduced inference latency, by integrating MediaPipe landmarks into a standard CNN architecture (similar to [15]). This works validates the findings of multiple other researchers ([5], [3]), who achieved greater classification accuracy when integrating MediaPipe hand landmarks (with both static 2d images, and 3d depth images) into their data preprocessing pipeline. [12] compared their model to ones based on pre-trained Inception CNNs, and non-convolutional models such as random forests and SVMs, and found that an integrated MediaPipe landmark/raw image model was able to outperform them in terms of accuracy ($\tilde{9}0\%$ for Inception and SVM baselines, compared to over 99% accuracy for their model) and in some cases, in inference time (particularly the SVM-based image model used in [18]).

One noticeable vector to reduce gesture recognition inference time is presented in [9]. This paper is distinct in that it trains on a variety of international sign languages, including Indian Sign Language (ISL) gestures. ISL is distinct from ASL in that ISL fingerspelling gestures typically use two hands instead of one, making the MediaPipe model (which is designed to be adaptable to generate landmarks for multiple hands in a single image) an appropriate modeling choice. Their model achieves low-latency by not using CNNs at all, instead using a lightweight SVM as the classifier, after reducing the size of the input search area using two-stage hand-detection/landmark extraction pipeline. [9] achieve accuracies of $> 98\%$ on Italian, Indian, and American sign language datasets, indicating that this approach is also effective at performing real-time gesture recognition.

2.3. 3D CNNs and Hand Modeling

3D CNNs are another prominent architecture in sign language recognition. One instance of this is [11], which uses Microsoft Kinect, a motion sensor that provides a color and depth stream and can track body movement, as an input device. With Kinect, the CNN has 5 inputs, including depth and a body skeleton, and achieves 94.2% accuracy, which is higher than baseline methods.

Another instance of this is [7], which uses a multi-stream architecture that comprises of CNNs and GANs to generate depth and joint information from RGB channels. Then, manual and non-manual features are processed in a 3D CNN. This multi-stream model receives the frames in RGB, segmented hands and faces, distance and speed maps, and the artificial depth maps generated by the GANs, and achieves 91% accuracy.

One more example of 3D CNNs is [1], which uses a fusion of parallel 3D CNN structures, where linear sampling is applied to select frames, and a 3D CNN learns the spatiotemporal features at certain times in the video sequence. Then, the 3D CNN extracts features from one of the clips, and then various methods for feature fusion are considered, including MLP, LSTM, and stacked autoencoders. After considering scenarios of all combinations, including signer-dependent and signer-independent (where the signers in the test data aren't included in the training data), the model that was signer dependent using MLP fusion achieved 98.12% accuracy. In particular, [7] employed state-of-the-art techniques by combining many architectures and using datasets in multiple languages of sign language. Also, [1] excels in testing many different model combinations and techniques, even applying PCA and t-SNE for data reduction. In contrast, [11] has no mention of data augmentation, which is important to consider to prevent overfitting.

3. Methods

3.1. Baselines

Our project is modeled as an ablation study—we built upon our model iteratively to test and evaluate each extension. We first started with a simple CNN, made up of 2 convolutional layers (with 32 and 64 channels respectively), each of which is followed by ReLU and 2×2 max pooling. The data is then flattened, and fed through two fully-connected layers, with 128 nodes and 24 nodes respectively, where 24 was the number of classes. Using a simple CNN as our initial model allowed us to develop a robust image preprocessing pipeline, and develop an initial understanding of how suited CNNs were as a general approach to this task. This CNN was implemented using the standard deep learning framework Pytorch [14], and image preprocessing was performed using the Python Imaging Library (Pillow) [4]

To add model complexity and additional pathways for feature extraction, we then chose to replicate the architecture in [15], using solely the input channel of grayscale images. This model will be referred to as our baseline model,

and provided us with a more comprehensive baseline for the classification task. For this CNN , we used the following architecture: 5 convolutional layers with a filter size of 3 and 32, 64, 128, and 512 filters and ReLU activation. Each is followed by a dropout layer and batch normalization, as a form of model regularization. Each convolutional block after the first is followed by a max pooling layer of size 2×2. We used a dropout probability of 0.3 and learning rate of 0.001 as initial hyperparameter values, before tuning. We modified the paper's architecture slightly to maintain output dimensions. The model ends with a fully connected-dropout-fully connected block as a classification head, to classify each image into one of the 24 classes. A complete diagram of this model's architecture can be seen in Figure 1.

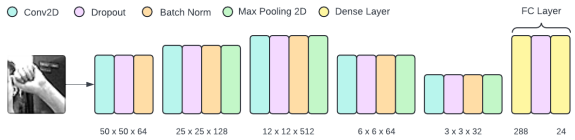

Figure 1. Baseline CNN Architecture, from [15]

A deeper CNN architecture such as this one offers the benefits of being able to extract more complex spatial features before classification. The model's loss was evaluated using cross-entropy loss, which can be seen in Equation 1.

$$\mathcal{L}_{\text{CE}}(y, \hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i), \tag{1}$$

3.2. Tuning and Augmentation

To maximize this model's ability to accurately classify letters, we conducted hyperparameter tuning on dropout probability and learning rate, with the intention of applying the most successful values to subsequent models. Since our baseline dataset was relatively small, we did not perform cross-fold validation to tune these parameters, but rather used the complete dataset. Full details of the result of this tuning can be found in Table 3. To improve this input channel's robustness to potential data sources in-the-wild, we also applied a variety of image data augmentation techniques, including salt and pepper noise, random rotation, random zoom, random shift, random horizontal flip, and random crop.

3.3. Hand Landmarks

To further compare our approach to models in the literature ([15, 5]), we integrated a second-input channel with our CNN. This channel takes in color images of signed gestures, and uses the MediaPipe hand landmark extraction model to obtain a set of 21 hand landmark coordinates [22]. This is followed by a shallow CNN consisting of 2 convolutional layers with 50 and 25 channels respectively, each with ReLU activation, batch normalization, and 2×2 max pooling. The MediaPipe landmark model itself is made up of two models - a palm detector to provide a bounding box for hands, and a hand landmark model, that provides a hand skeleton in the form of 21 hand-knuckle coordinates within the image.

The palm detector allows the model to localize the hand to a particular area of the image. The detector uses an encoder-decoder feature extractor, built on the idea of a Feature Pyramid Network (FPN) [13]. FPNs are a type of CNN architecture that were specifically designed to enhance the ability of CNNs to detect objects at multiple scales. This helps models develop scale-invariance, which is useful for our proposed task in that it reduces the need for scale-based data augmentation. FPNs are typically computed on top of backbone CNNs such as ResNet, and work by successively extracting feature maps at different stages of the backbone network's architecture (for example, in the ResNet case, these feature maps are taken from different residual blocks). Feature maps from different stages are them combined, often using upsampling and element-wise addition, to produce a 'pyramid' of features at different scales. In the case of palm detection, standard RoI pooling methods such as Fast R-CNN [6] are used to produce RoIs from the feature maps at different levels of the pyramid. An example of this type of architecture can be seen in Figure 2.

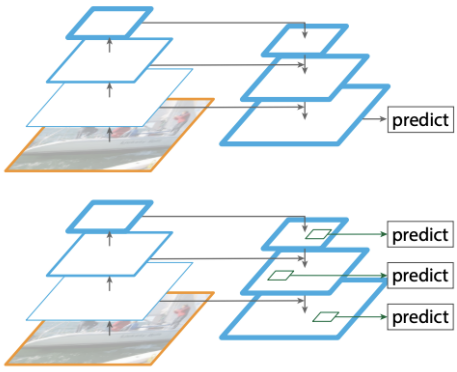

Figure 2. FPN region proposal, from [13]

Once bounding boxes around hands have been detected, the hand landmark model performs landmark localization as a regression task to find key coordinates on the hands. This model takes a proposed region of the input image from the palm detection model as input, and outputs 21 2.5D hand landmark coordinates (x, y, and z relative to wrist landmark), using some type of feature extractor (details of this extractor are not provided in [22]). The model also outputs a confidence score, indicating how likely it thinks that the region contains a hand - in the case that the confidence score

was too low, we chose to fill in the output tensor with zeros, to maintain shape consistency for the remainder of the network. The topology of the landmark coordinates can be seen in Figure 3.

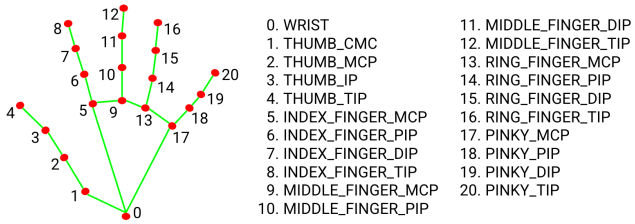

Figure 3. Hand landmark topology, from [2]

In our project, we did not implement or train this model ourselves. Rather, we used the MediaPipe Python package [22] to instantiate a pre-trained hand landmark model, and passed the outputs of that model to the shallow CNN that we did instantiate and train. The output of this shallow CNN was flattened and concatenated with the output of the grayscale channel, and the combined tensors were passed through two fully connected-dropout-fully connected classification head for final classification.

3.4. Retesting on different data

Because our initial dataset was too simple (see Section 4), we chose to replace our initial dataset with a larger ASL dataset that contained more complex images (different backgrounds, more occlusion, more potential sources of distraction like faces) and repeated the same steps mentioned above.

4. Dataset and Features

For our primary dataset, we used the same dataset used by [15], which contains 24 letters (excluding J and Z) [16]. The dataset contains images from 5 non-native signers, with over 500 images for each sign per signer. The dataset contains 65,748 images total. We split the data as following: 70% train, 15% validation, and 15% test.

For the first input mode to our model, we applied the following pre-processing steps to replicate [15]: converting the images to grayscale, sharpening using the same sharpening filter as in [15], resizing to 50×50, and normalizing the grayscale values. An example of the results of this process can be seen in Figure 4. For the second input channel, the only pre-processing step was resizing the images to 224×224 so that they could be passed to the MediaPipe detector.

However, after initial results when using the data from [16], we discovered that our images were too simple, leading to our model performing extremely well with little tuning or extensions (see section 5.1). This motivated the application of noisy augmentation methods, to make our dataset more diverse, and the classification task more difficult, as mentioned in the related work section. We applied salt and pepper noise (0.05), random rotation within 10 degrees, random zoom (10%), random shift (0.1), random horizontal flip (0.5), and random crop (50 $\times$50 pixels) (Figure 4). The salt and pepper noise was implemented ourselves, using NumPy [10], and the remaining augmentation methods were implemented using existing functions in Pillow [4]. After this augmentation, images were converted to grayscale and resized to 50×50 pixels, and normalized before being fed into the model's first input channel.

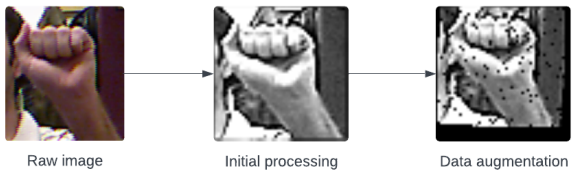

Figure 4. Image Processing and Data Augmentation Example

To further evaluate how well our model architecture could perform on more difficult data, we replaced our existing dataset with more complex data, from [20]. This new dataset consists of 233,104 images for 29 ASL classes, including all 26 characters of the alphabet as well as signs for "delete", "nothing", and "space". For consistency and comparability against results using our first dataset, we excluded the signs for J and Z, and also excluded the signs for delete, nothing, and space. We applied the same pre-processing and data augmentation to these images, again resulting in images of 50×50 pixels for the first input channel. We split the data as before, into 70% train, 15% validation, and 15% test. This dataset is significantly more complex than the first one, as it contains more diverse and larger backgrounds, along with the faces of signers. For example, some images are taken in rooms where there are multiple different objects in the background. We hoped that by using this data, that data augmentation and hand detection would result in improved performance.

5. Experiments, Results, and Discussion

For quantitative evaluation, our primary metric was overall accuracy on the validation set, i.e. the percentage of validation examples which were correctly classified. On particular models, we also calculated the precision, recall, and F1 score for each individual class for a more detailed breakdown. These metrics are calculated as follows:

$$\text{Precision}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i}$$

$$\text{Recall}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i}$$

$$\text{F1 Score}_i = 2 \times \frac{\text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

5.1. Smaller Dataset

For our baseline models, the model's overall accuracy on a randomly selected validation set made up of 15% of the data found in [16] ("smaller dataset") is found in Table 1. This subsection contains an explanation of each of these models, and our decision-making process when considering how to extend the baselines.

Model	Validation Accuracy
Results from [15], single input channel, with data augmentation	96.29%
Results from [15], two input channels, with data augmentation	98.42%
Simple CNN	98.14%
Baseline Model	99.34%
Baseline Model with Data Augmentation	96.40%
Complete Model	96.5%

Table 1. All Model Results, Smaller Dataset

Our initial point of comparison, the top row of 1, is the validation accuracy reported in [15] on a model with only the 50×50 grayscale input channel, and data augmentation. Our second point of comparison is the validation accuracy reported in [15] on a dual-input model (grayscale images and landmarks), with data augmentation.

The first model we tested, as a proof-of-concept, was a relatively simple CNN with two convolutional layers and a single grayscale image input head. We were surprised to find that this model was able to achieve extremely high accuracy on the smaller dataset (98.14%), when the data had not been augmented. Our initial thoughts were that we were overfitting to the data. However, plotting our training and validation accuracies and losses as a function of epoch (Appendix 12) indicated that we were not overfitting to the training data, but that the model was actually extremely effective at classifying this dataset. One important thing to note about these plots is that the validation loss and accuracy outperforms training loss and accuracy due to the fact that the model uses dropout, and so the model's full classification capability is only seen at test time.

We then replicated the deeper CNN architecture in [15] ("baseline model"), but again only initially worked with a single input data source. As expected, this model outperformed the Simple CNN, as the increased depth likely allowed for it to extract richer semantic features from the input images. A detailed breakdown of our precision, recall, and F1 scores for this model is provided in Table 2, and the accuracies and losses are plotted in Appendix 13.

Class	Precision	Recall	F1 Score	Support
A	1.00	0.99	0.99	412
B	0.99	1.00	0.99	430
C	1.00	0.99	0.99	431
D	0.99	0.99	0.99	403
E	1.00	1.00	1.00	400
F	1.00	0.99	0.99	388
G	1.00	0.99	0.99	376
H	0.99	1.00	0.99	409
I	0.99	0.99	0.99	395
K	1.00	0.99	0.99	435
L	0.99	0.99	0.99	396
M	0.98	0.99	0.99	410
N	0.98	0.98	0.98	418
O	0.99	0.99	0.99	387
P	0.99	0.98	0.98	434
Q	0.98	0.98	0.98	378
R	0.97	1.00	0.98	442
S	0.99	0.99	0.99	415
T	0.99	0.99	0.99	430
U	0.99	0.99	0.99	407
V	0.97	0.99	0.98	418
W	0.99	0.98	0.99	457
X	0.99	0.97	0.98	389
Y	0.99	0.99	0.99	406
Total	0.99	0.99	0.99	9866

Table 2. Baseline Model, Smaller Dataset Classification Report

At this point in our experimentation process, we chose to conduct hyperparameter tuning on the Adam learning rate and dropout probability of each layer. Our results indicated that the best combination of parameters were a dropout probability of 0.2 and a learning rate of 0.0005, and these parameters were used for all subsequent models. The classification accuracies for the various combinations of parameters can be found in Table 3.

		Dropout Rate			
		0.1	0.2	0.3	0.4
Learning Rate	1e-4	99.44%	99.40%	98.84%	97.40%
	5e-4	99.49%	99.62%	99.46%	99.14%
	1e-3	99.41%	99.61%	99.49%	98.81%
	5e-3	99.25%	99.17%	99.02%	98.23%
	0.01	98.82%	98.55%	97.92%	95.84%

Table 3. Hyperparameter Tuning Classification Accuracy

To further bring our experiments more closely in line with those in the literature, we then applied data augmentation to the grayscale images (as detailed in Section 4). A baseline model with only the grayscale input head trained on these augmented images was able to achieve 96.4% val-

idation accuracy. As expected, due to the simplicity of the initial data, our final performance was worse than the baseline model. Similarly to the baseline model, validation accuracy was higher than training accuracy, likely due to the fact that our model uses dropout layers (Appendix 14). As seen in the confusion matrix, the characters most confused were classes 14 and 15 (P and Q), 19 and 20 (U and V), 20 and 21 (V and W), and 0 and 18 (A and T) (Figure 5). Given the signs for these characters, this misclassification is likely, as the signs for P and Q look very similar, and the signs for U, V, and W, and A and T, all have a similar hand position (Figure 6)[16]. Also, since this dataset consists of signs from non-native signers, slight errors and variations in signs may contribute to this misclassification.

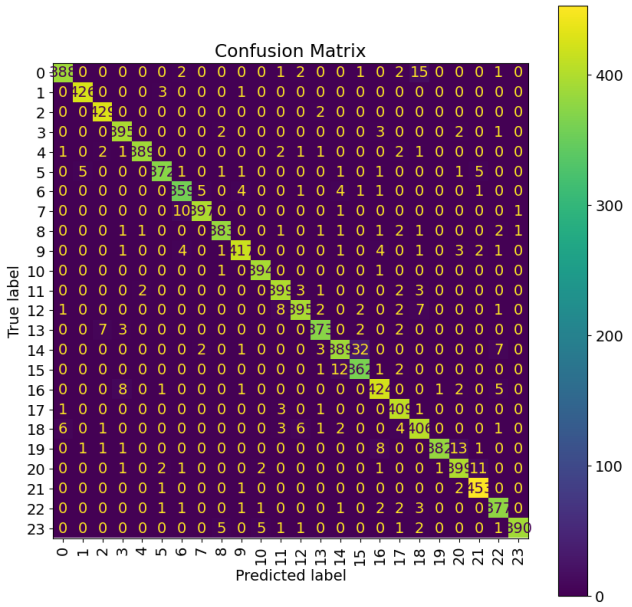

Figure 5. Confusion Matrix for Baseline Model, Smaller Dataset, Data Augmentation

After adding hand landmark detection using a minimum detection confidence of 0.5, we achieved a slightly higher validation accuracy of 96.5% on our complete model (Figure 7). and reduced the misclassification among classes 19, 20, and 21 (U, V, and W), and classes 0 and 18 (A and T) (Figure 8). For test accuracy on our complete model, we achieved 96.8% accuracy. Our test accuracy is lower than the results achieved by [15], but there are a few differences to note—our model ran on 10 epochs and used a fixed learning rate of 0.0005 due to hardware restrictions, while [15] ran their model on 50 epochs with a dynamic learning rate.

Despite improved accuracy overall when using hand landmark detection, we noticed that the number of misclassified images among classes 14 and 15 (P and Q) remains the same, and surprisingly, the misclassification between classes 6 and 7 (G and H) and 12 and 18 (N and T) increased. As seen in Figure 6, the signs for these characters are quite similar. Even with hand landmark detection, it is possible that the model has confused these two signs due to their similarity.

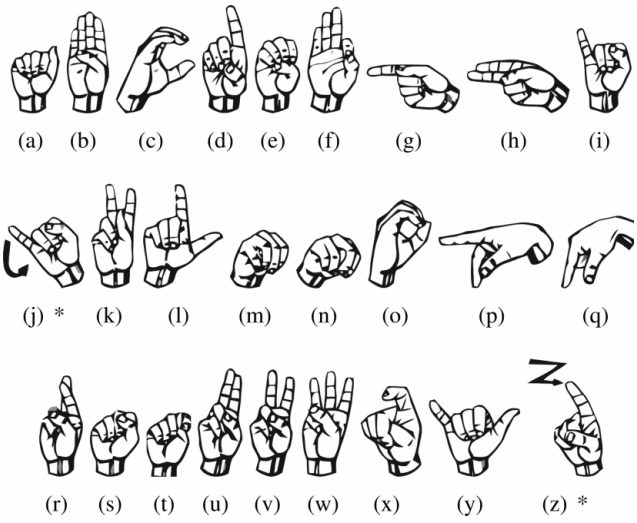

Figure 6. ASL alphabet

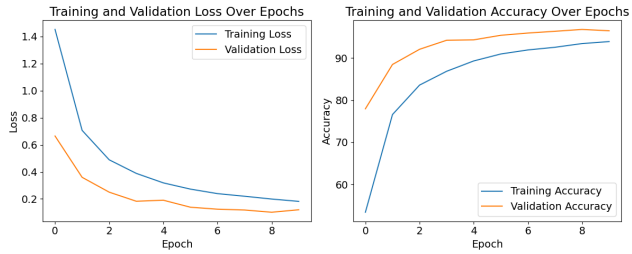

Figure 7. Losses and Accuracies for Complete Model, Smaller Dataset

There are a few reasons why this could have happened. First, there are some images where the hand landmark model is not able to detect a hand in the image. In this case, the prediction is solely based on the image, so for commonly confused signs, the coordinates are not able to assist in preventing misclassification. It is also possible that the coordinates are incorrect based on the image, which may worsen the prediction. Examples of both the missing landmarks case and the incorrect landmarks case can be seen in Figure 9. With a minimum detection confidence of 0.5, 23.27% of the images had landmark coordinates. We tried using a lower minimum detection confidence, but noticed that images tended to have incorrect coordinates—we decided to err towards no coordinates rather than incorrect coordinates as we believed this was more likely to produce a correct prediction.

Then, we examined the precision, recall, and F1 score. The letters G and T had the lowest precision, while the letters N and Q had the lowest recall. Overall, classes N, Q,

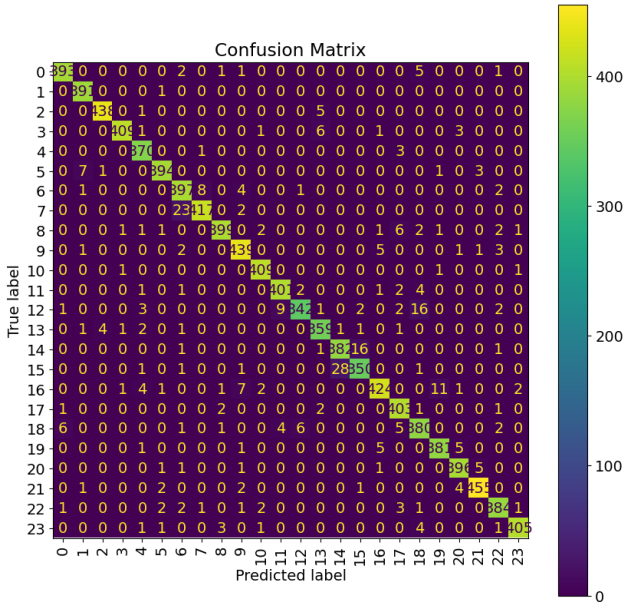

Figure 8. Confusion Matrix for Complete Model, Smaller Dataset

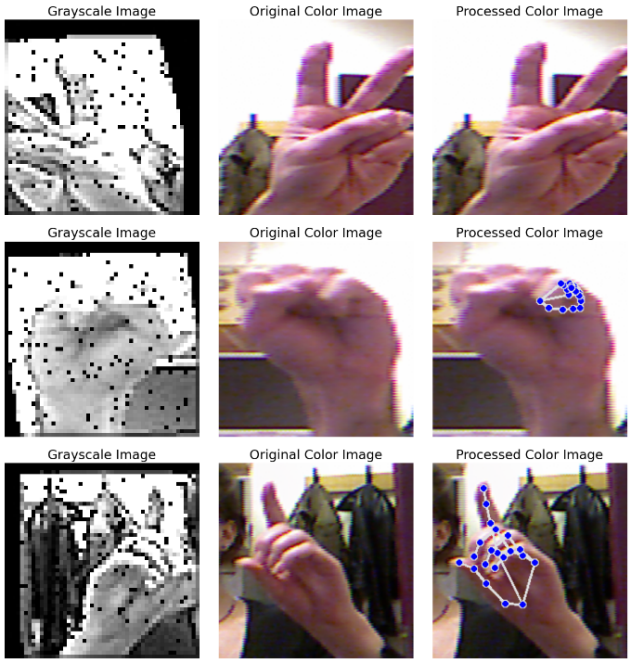

Figure 9. Data Augmentation and Hand Landmark Detection Examples

and T had the lowest F1 scores (Table **??**).

5.2. Larger Dataset

We then repeated the following steps after replacing our data with a larger and more complex dataset. A summary of our results is found in Table 5.

For the simple CNN, the model did not perform as well as the same model did with the simpler data, which is expected (Appendix 15).

Next, after running the baseline model on the new data, our validation accuracy significantly improved to 98.25% (Appendix 16).

With data augmentation, we achieved a higher validation accuracy of 98.93%–this indicates that data augmentation helped make our model more robust to the complex data (Appendix 17).

Then adding hand landmark detection, we had a validation accuracy of 96.6% (Figure 10). We also had a test accuracy of 96.42%. Our test accuracy is again lower than the results obtained by [15], but this is likely due to slight changes in architecture as mentioned above.

After examining the confusion matrix, it is clear that the

Class	Precision	Recall	F1 Score	Support
A	0.98	0.98	0.98	403
B	0.97	1.00	0.98	392
C	0.99	0.99	0.99	444
D	0.99	0.97	0.98	421
E	0.96	0.99	0.97	374
F	0.98	0.97	0.97	406
G	0.92	0.96	0.94	413
H	0.98	0.94	0.96	442
I	0.98	0.96	0.97	417
K	0.96	0.97	0.96	452
L	0.98	0.99	0.99	412
M	0.97	0.97	0.97	412
N	0.97	0.90	0.94	378
O	0.96	0.97	0.96	371
P	0.93	0.95	0.94	401
Q	0.95	0.92	0.93	382
R	0.97	0.93	0.95	454
S	0.95	0.98	0.97	410
T	0.92	0.94	0.93	405
U	0.96	0.97	0.97	393
V	0.97	0.98	0.97	405
W	0.98	0.98	0.98	465
X	0.96	0.96	0.96	398
Y	0.99	0.97	0.98	416
Total	0.96	0.96	0.96	9866

Table 4. Complete Model, Smaller Dataset Classification Report

Model	Validation Accuracy
Simple CNN	57.45%
Baseline Model	98.25%
Baseline Model with Data Augmentation	98.93%
Complete Model	96.6%

Table 5. All Model Results, Larger Dataset

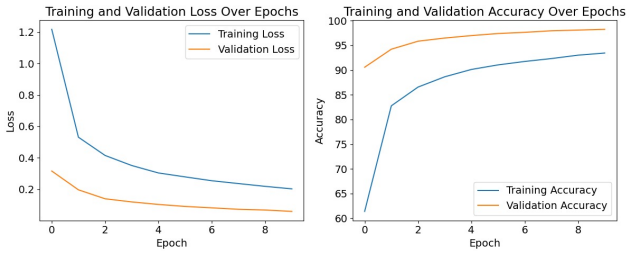

Figure 10. Losses and Accuracies for Complete Model, Larger Dataset

model confuses classes 19, 20, and 21 (U, V, and W), 0 and 18 (A and T), 16 and 19 (R and U), and classes 11 and 12 (M and N) (Figure 11). Similarly to the full model on the smaller data, the model still confuses signs that have similar hand positions even with hand landmark detection. However, this model slightly out performed the full model on the smaller data, which is important to note considering this data is more complex.

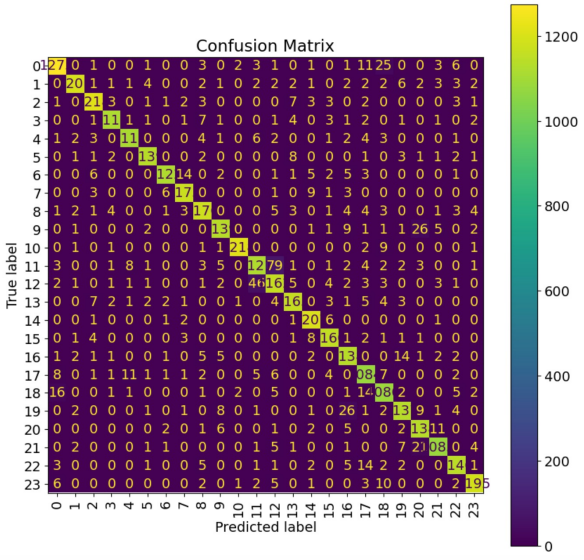

Figure 11. Confusion Matrix for Complete Model, Larger Dataset

Using the new data, we had 66.52% of images have coordinates with the same minimum detection confidence of 0.5, which is a significant improvement. We were surprised to see that adding hand landmark detection slightly worsened results. This may be because of the added complexity of the data—it may be that additional hand detection methods may be needed to make the landmark detection more effective, like RPN or cropping the RoI.

6. Conclusion and Future Work

Overall, our project was aimed at ASL character recognition with CNNs, and hoped to improve performance by implementing hyperparameter tuning, data augmentation, and hand landmark detection on two different datasets, one more complex than the other. After running various models, we found that our model using the more complex data with data augmentation had a 98.93% validation accuracy. However, our full model, which includes hand landmark detection, achieved a test accuracy of 96.42%. Based on our error analysis, it makes sense that adding data augmentation on the more complex data improved accuracy, while doing so on the simpler data worsened accuracy. While we expected hand landmark detection to improve accuracy on the complex data, it is likely that further work in hand detection and position prediction is needed to improve results. Some future work may include running the model on a greater number of epochs, further tuning of the minimum detection confidence parameter for hand landmark detection, tuning the parameters for data augmentation, and implementing additional hand detection or coordinate techniques.

7. Contributions & Acknowledgements

All coding and report writing was split equally across the milestone and the final report. Specifically, Arnav implemented the baseline models and second input channel, Anusha did data pre-processing, error analysis, and augmentation, and Malavi adapted the models to the new dataset, ran the full models, and collected error analysis plots.

8. Appendices

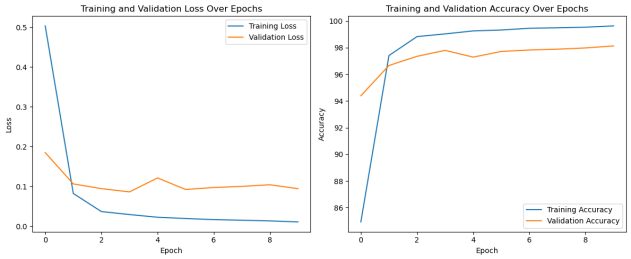

Figure 12. Losses and Accuracies for Simple CNN, Smaller Dataset

Figure 13. Losses and Accuracies for Baseline Model, Smaller Dataset

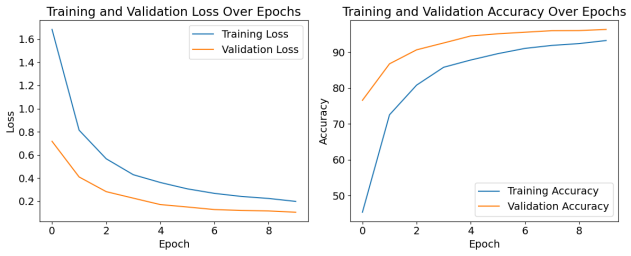

Figure 14. Losses and Accuracies for Baseline Model, Data Augmentation, Smaller Dataset

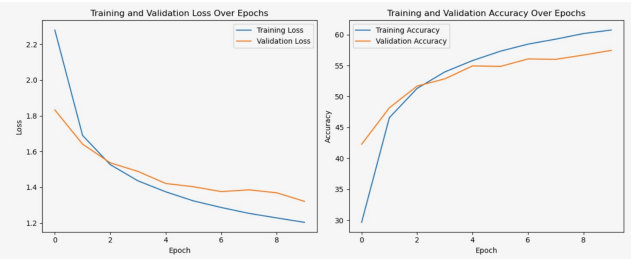

Figure 15. Losses and Accuracies for Simple CNN, Larger Dataset

References

[1] M. Al-Hammadi, G. Muhammad, W. Abdul, M. Alsulaiman, M. A. Bencherif, and M. A. Mekhtiche. Hand gesture recognition for sign language using 3dcnn. *IEEE Access*, 8:79491–79509, 2020.

Figure 16. Losses and Accuracies for Baseline Model, Larger Dataset

Figure 17. Losses and Accuracies for Complete Model, Larger Dataset

[2] V. Bazarevsky, F. Zhang, A. Vakunov, C.-L. Chang, and M. Grundmann. Mediapipe hands: On-device real-time hand tracking. `https://github.com/google-ai-edge/mediapipe/blob/master/docs/solutions/hands.md`, 2019. Accessed: 2024-06-05.

[3] J. Bora, S. Dehingia, A. Boruah, A. A. Chetia, and D. Gogoi. Real-time assamese sign language recognition using mediapipe and deep learning. *Procedia Computer Science*, 218:1384–1393, 2023.

[4] A. Clark and Contributors. Pillow - the friendly pil fork, 2024. Version 10.3.0.

[5] A. Deep, A. Litoriya, A. Ingole, V. Asare, S. M. Bhole, and S. Pathak. Realtime sign language detection and recognition. In *2022 2nd Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–4. IEEE, 2022.

[6] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[7] R. R. G. Giulia Zanon de Castro and F. G. Guimarães. Automatic translation of sign language with multi-stream 3d cnn and generation of artificial depth maps. *Expert Systems with Applications*, 215(119394), 2023.

[8] H. Guo, G. Wang, X. Chen, C. Zhang, F. Qiao, and H. Yang. Region ensemble network: Improving convolutional network for hand pose estimation. *2017 IEEE International Conference on Image Processing (ICIP)*, Sept. 2017.

[9] A. Halder and A. Tayade. Real-time vernacular sign language recognition using mediapipe and machine learning. *Journal homepage: www. ijrpr. com ISSN*, 2582:7421, 2021.

[10] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe,

P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy, 2020. Version 1.26.4.

[11] J. Huang, W. Zhou, H. Li, and W. Li. Sign language recognition using 3d convolutional neural networks. *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2015.

[12] R. Kumar, A. Bajpai, and A. Sinha. Mediapipe and cnns for real-time asl gesture recognition. *arXiv preprint arXiv:2305.05296*, 2023.

[13] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[15] R. Pathan, M. Biswas, S. Yasmin, M. Khandaker, M. Salman, and A. Youssef. Sign language recognition using the fusion of image and hand landmarks through multi-headed convolutional neural network. *Nature*, 13(16975), 2023.

[16] N. Pugeault and R. Bowden. Spelling it out: Real-time asl fingerspelling recognition. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1114–1119, 2011.

[17] K. K. R. Rastgoo and S. Escalera. Multi-modal deep hand sign language recognition in still images using restricted boltzmann machine. *Entropy*, 20(11), 2011.

[18] J. Rekha, J. Bhattacharya, and S. Majumder. Shape, texture and local movement hand gesture features for indian sign language recognition. In *3rd international conference on trendz in information sciences & computing (TISC2011)*, pages 30–35. IEEE, 2011.

[19] S. T. A. S. S. Shanta and M. R. Kabir. Bangla sign language detection using sift and cnn. *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6, 2018.

[20] D. Sau. Asl(american sign language) alphabet dataset, 2022.

[21] S. Shahriar, A. Siddiquee, T. Islam, A. Ghosh, R. Chakraborty, A. I. Khan, C. Shahnaz, and S. A. Fattah. Real-time american sign language recognition using skin segmentation and image category classification with convolutional neural network and deep learning. In *TENCON 2018-2018 IEEE Region 10 Conference*, pages 1168–1171. IEEE, 2018.

[22] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.