ECS 171 MACHINE LEARNING

GROUP 31 PROJECT REPORT

American Sign Language Interpretation

GROUP MEMBERS:

Nicholas Mueller, Pavittar Singh, Siddarth Vinnakota, Kay Krachenfels, Alexander Hoang

GITHUB:

https://github.com/AlexHoangs/ASL-Recognition

Introduction and background

According to the National Institute of Deafness and Other Communication Disorders, American Sign Language (ASL) is a complete and natural language that is expressed by movements of the hands and face. It is the primary language of many North Americans who are deaf and hard of hearing. ASL is more than just a method of communication. It is an advanced language system with its own grammar and syntax which is distinct from spoken English. This language not only facilitates communication among the deaf community but also fosters a sense of identity and community among its users.

Originating from a combination of French Sign Language and local sign systems in the early 19th century, ASL has since evolved into a sophisticated and robust form of communication. However, ASL is not universal. It is distinct from other sign languages used around the world, highlighting the diversity in the ways deaf communities communicate.

Here, the role of machine learning becomes increasingly significant. In bridging the communicative gap, machine learning offers innovative solutions. Through the application of advanced algorithms and neural networks, machine learning models are trained to interpret the complex gestures and facial expressions of ASL. These models excel at translating ASL into text or spoken language in real time, enhancing communication between deaf and hearing individuals.

The implications of a machine learning model capable of translating ASL in real-time are vast and encompass many fields. This technology promises not only enhanced communication but also broader accessibility and inclusion.

In the educational realm, such a tool can revolutionize the learning experience for deaf and hard-of-hearing students. It would enable them to engage directly with spoken language content in real time, fostering a more inclusive and interactive educational environment. In healthcare, the benefits of real-time ASL translation are also vast. Effective communication is crucial in medical settings and a gap in understanding can have serious implications. A real-time translation tool can facilitate clear and accurate communication between healthcare providers and deaf patients, ensuring that medical information is conveyed correctly and patients' concerns are fully understood. This can lead to better patient outcomes, reduced misdiagnoses, and a higher level of care. Furthermore, in public services and emergency response, the ability to communicate effectively with the deaf community in real time can be life-saving. Law enforcement, emergency responders, and public service officials equipped with real-time ASL translation tools can respond more effectively to the needs of the deaf community, ensuring they receive timely and appropriate assistance and support.

With this in mind, the development of a machine learning model for real-time ASL translation is more than a technological advancement, it is a step towards a more accessible and inclusive world. By bridging communication gaps, it has the potential to transform the lives of deaf and hard-of-hearing individuals, allowing them to engage more fully in various aspects of society.

In this project, we aim to deliver such a model. Using various techniques learned throughout the quarter, our group has focused on developing a machine-learning model capable of accurately interpreting and translating ASL.

Literature review

Machine learning has a place in many areas of research and the field of Sign Language interpretation is no different. Countless studies have been published focusing on Machine Learning and its application to Sign Language and countless more are ongoing. Here, we will take a look at a few research efforts similar to our own.

In 2019, the International Research Journal of Engineering and Technology (IRJET) published a study that investigated the development of an application that would ”translate sign language to English in the form of text and audio, thus aiding communication with sign language”. They used a dataset with approximately 6,000 images on which they performed a 80:20 split to obtain training and testing data respectively. They would go on to implement an SVM that would perform classification on images that would be obtained from the user's webcam. They would then convert the text obtained from the SVM to audio. Their model was able to produce accurate translations 88% of the time.

Similarly, a study was published in 2012 by Carnegie Mellon University's Department of Mechanical Engineering. The team of Atwood et al. aimed to create a machine-learning model that would translate static images of ASL to English letters. They had multiple approaches one of which was a single hidden layer neural network. Every single pixel in the image served as an individual input for the neural network and k=2 fold cross-validation was used to prevent overfitting. They achieved an accuracy of 95.8%.

Dataset Description and exploratory data analysis of the dataset

To build our ASL interpreting model, we utilized the ASL Alphabet dataset, which we found on Kaggle. The dataset consisted of over 87,000+ images which we made use of to both train and test our model. The dataset included images representing all 26 letters of the English alphabet, alongside special characters for space, delete, and 'nothing'. For each ASL character, we had 3000 samples, i.e. images, from which we could train our model to detect said character. Each image was 200 by 200 pixels and contained a hand gesture representing an ASL character. Each image was also colored. The images also varied in brightness. Figure 1 presents a series of graphs, each illustrating the distribution of brightness levels against their corresponding counts for each ASL character. The average brightness for each character was around the 140 mark however there were quite of few images that had brightnesses drastically lower and higher than this. This meant that greyscaling would be necessary in the process of normalizing our input data however we'll save this discussion for later sections.

During our initial assessment of the dataset, we noticed a wide variation in the hand gestures, both in terms of the angle and clarity. This variation is beneficial for our model, as it mimics real-world scenarios where an interpreter might encounter a broad range of gesture styles and clarity. Understanding and incorporating this variability is key to developing a model that is both versatile and accurate.

In our analysis, we also paid close attention to the diversity in angles, backgrounds, and variations in skin color and shadow in the dataset. This variance among the images added significant complexity to the gesture recognition task. The variety in hand angles and backgrounds, combined with the differences in skin tones and the presence of shadows presented a challenging environment for accurate gesture interpretation. It was imperative for our model to effectively interpret these nuances to ensure robust performance. Addressing these variations was not only crucial for the model's accuracy but also for its reliability and effectiveness in a real-world setting, where such conditions are commonplace.

Given the diversity of samples the dataset provides, you can see why our group would choose it as the dataset from which to train our model. Our goal is to create a model that can provide real-world versatility and having a well-structured and comprehensive dataset is the first and arguably the most important step. The ASL Alphabet dataset not only offers this diversity but also the depth needed for nuanced learning, setting a strong foundation for our subsequent methodology.

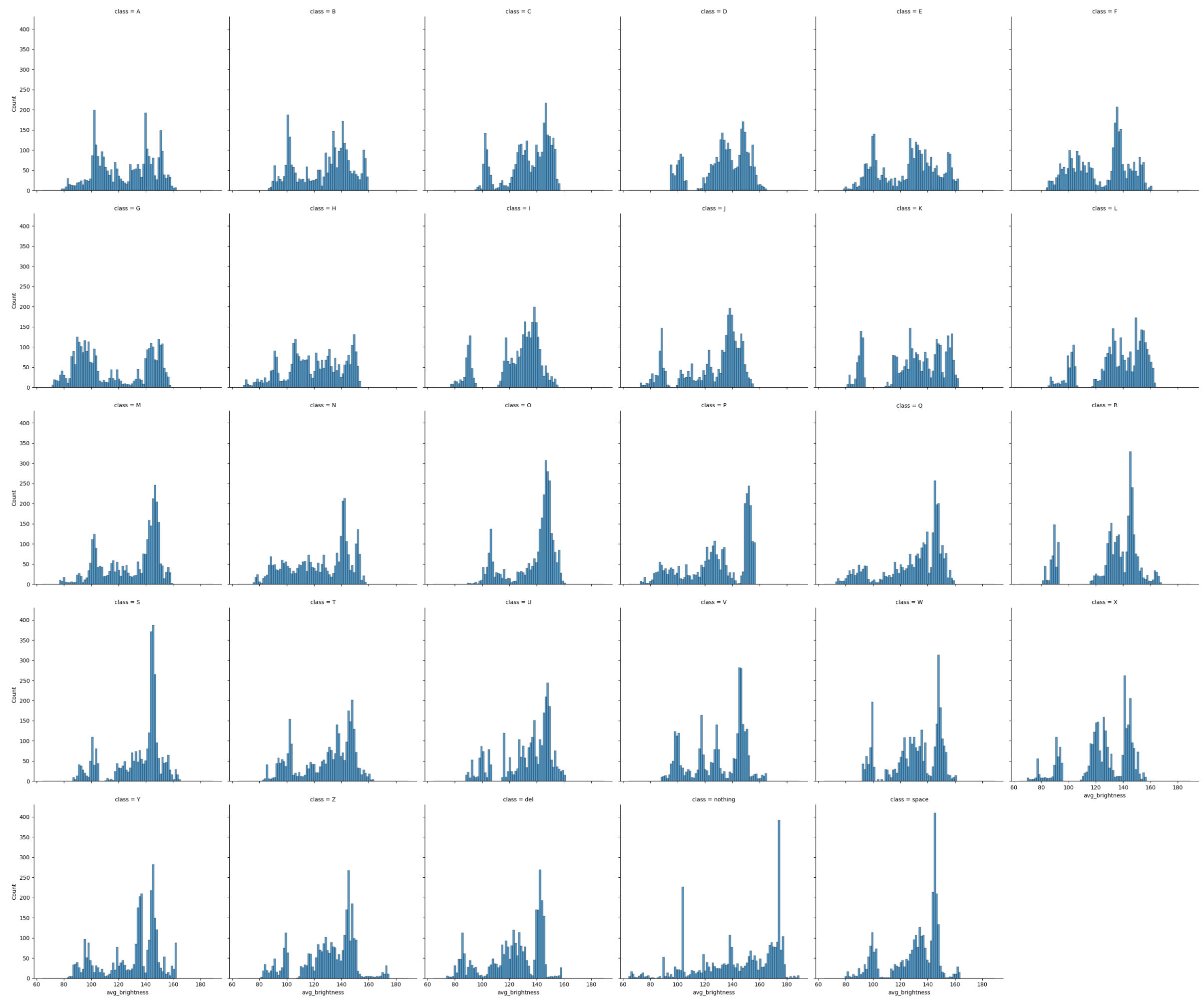


Figure 1: Distribution of brightness levels (x-axis) and their counts (y-axis) for each ASL character in the dataset.

With our in-depth exploratory analysis of the ASL Alphabet dataset complete, we now turn our attention to creating a methodology that addresses the identified challenges. As we proceed to the next section, we will delve into how we have capitalized on these dataset features to tailor a robust and accurate ASL interpreting model that is adaptable to the dynamic conditions of real-world use.

Proposed methodology

Before we begin to construct a model, we must first determine the characteristics of the problem we're trying to solve and choose an algorithm best suited for delivering on those needs. Based on what we learned from the exploratory data analysis, we can conclude that determining what character a hand is gesturing is a classification problem and more specifically, a multi-classification problem because there are multiple distinct classes into which the inputs can be categorized. Based on both our own research and knowledge we've acquired in class, we knew that Neural Networks were the algorithm of choice for building our model. However, because our input to the model would be images, we learned that a Convolutional Neural Network (CNN) would be our best bet. Therefore, we decided to build a CNN as it fit our needs best.

Once we know what we are going to build, we need to prepare and sanitize our data for the training process. Recalling our previous discussion regarding the brightness of our input images, we know that we have to greyscale them because the brightness in the images varies. This approach would address not only the brightness variations in the images but also the challenges posed by different shadows and angles. All three of these characteristics are common in real-world scenarios therefore greyscaling our images is a crucial step we must take while developing our model. By implementing greyscaling, we ensure that each image processed by our model is standardized and mirrors the conditions of the training data. This consistency is key to achieving reliable and accurate model performance.

To effectively greyscale our images, we could use the OpenCV Python library, which offers comprehensive tools for image processing. OpenCV's functionality for converting color images to greyscale will allow us to normalize the brightness and contrast variations across our dataset. This normalization is crucial for reducing the complexity of the image data and focusing the model's learning on the shapes and patterns of hand gestures, which are the key elements in ASL character recognition.

Given that our training images are already 200 by 200 pixels, our preprocessing routine for them primarily involves greyscaling. However, for any new unseen images that the model will encounter in real-world scenarios, it is essential to implement an additional preprocessing step which would be to resize these images to the same 200 by 200 pixel resolution such that it ensures consistency with the training data. We will develop a preprocessing function that automatically rescales any input image to these dimensions, maintaining the aspect ratio and using padding if necessary to avoid distortion.

To further enhance our model's ability to generalize, we will also apply image augmentation techniques during the training process. These techniques, such as rotation, zooming, and horizontal flipping, will artificially expand our dataset and simulate a wider range of hand positions and orientations. TensorFlow's Keras preprocessing layers offer a convenient and efficient way to implement this augmentation via the **keras.Sequential** model

After augmenting our dataset, we can pass it to our CNN. Here, we perform the bulk of our training. We will rescale the pixel values of the images to a range of 0 to 1. This will normalize our input. Then we will pass the input to a convolutional layer which extracts features from the images and then to a max pooling layer which will reduce computational complexity and prevent overfitting. We will repeat this two-step process, alternating between a convolutional layer and a max pooling layer, multiple times. This will provide maximum feature extraction while avoiding overfitting which is perfect for our model. The convolutional layers will also use the ReLu activation function because it introduces non-linearity into the model.

Once feature extraction is satisfactory, we will introduce a dropout layer with a probability P, again to prevent overfitting.

To actually get output from our model, we must flatten the output of the convolutional layers via a flattening layer. This will convert their multi-dimensional output into a 1 output which we can then pass to dense layers. The dense layers will actually perform the classification and give us an output.

This entire methodology will be implemented using the TensorFlow library. It has all of the useful functions we need to implement this CNN to the fullest.

To actually test our model, we will utilize the test images provided by our dataset. They will introduce unseen inputs to our model and will allow us to see just how accurate and performant our model is. This testing process is crucial in determining how well our model generalizes to new data. This will provide a measure of its effectiveness in interpreting ASL.

Experimental results

With the full 3000 images, we ran our model twice. The first one was the initial run of the Sequential CNN, without any major changes to the images. Our results illustrated in Figure 2. While the training accuracy, i.e. the blue line, yields an extremely high training accuracy of 99.47 percent after 10 epochs, it is larger than the validation accuracy, i.e. the orange line, which after 10 epochs is 98.4 percent. While the validation accuracy is still high, its comparison to the general accuracy in terms of being lower could indicate that the model is overfitting. Overfitting means that the model is trained too well on the training data. While this may seem to be beneficial, this means that if the training data has some internal variances or noise and if the model takes those into account, it may fail to accurately translate to a validation set. There are many methods to remedy this, such as employing cross-validation techniques as well as increasing the sample size.

In our case, we decided to augment the images. This means that for each of the images, we randomly transform the data into three categories: a random horizontal flip, a random rotation, and a random zoom. It is important to know that these changes are very minimal. By doing these random augmentations, we could yield more believable-looking data i.e. data that is likely to occur in real-world scenarios. These images could expose new features for the neural network to take into account, and by doing so, we could reduce if not outright prevent overfitting. After augmenting the images, we ran the same model with the same number of images, and the results are displayed in Figure 3. As we can see, there is a slight decrease in the training accuracy. In the original run, we tallied a 99.47 percent training accuracy, while this variation indicated a training accuracy of 95.19, so there is a small decrease of 4.28 percent. However, the biggest difference lies in the validation accuracy, which in this run yields a 97.43 percent accuracy, which is larger than the training accuracy in the same run. In Figure 3, we see the orange line which represents the validation accuracy is consistently higher than the blue line, which represents the training accuracy. We consider this model an improvement because it does reduce the overfitting problem we faced in the original run. Also, even if the accuracies are less than the first run, it is not a significant loss. However, both variations could be used for American Sign Language identification.

Below, there is a table depicted in Figure 4 that illustrates the lists of other variations to the model, including the prior 2 runs. We ran this Sequential model with the other possible variations of our images. For one, we altered the sample size to see if similar results could be achieved with a smaller computational time. We also have a set where we did more of the image augmentation. We also used images that are grayscaled, or converted to black and white, to verify if the images' color is not required for classification. Additionally, we incorporated an image set with binary thresholding, which converts the colors of the image into one of 2 colors. This table records not only the training and validation accuracy but also their respective losses. It is also important to know that the validation accuracy sorts this table, so the better-performing variant is on top, while the weaker variants are on the bottom.

In the table, we see a change in accuracy depending on the size of the sample. With the full data set of 3000 images, we achieved the highest accuracy of around 99 percent, which means that the model was able to identify the correct ASL sign 99 out of 100 times. However, the model training process takes hours, due to the sheer volume of images. Consequently, we looked at subsets of the size, both 300 images (which is one-tenth of the original number) and 100 images, to see if we could achieve similar results with a training input of a smaller number. Looking at the 300 image data, for example, the overall accuracy doesn't seem to change much. In fact, the 300 images with just the class input yield a 99 percent training accuracy. However, one important thing to note is the validation accuracy, which is not extremely low, but it does indicate a notion of the model overfitting,

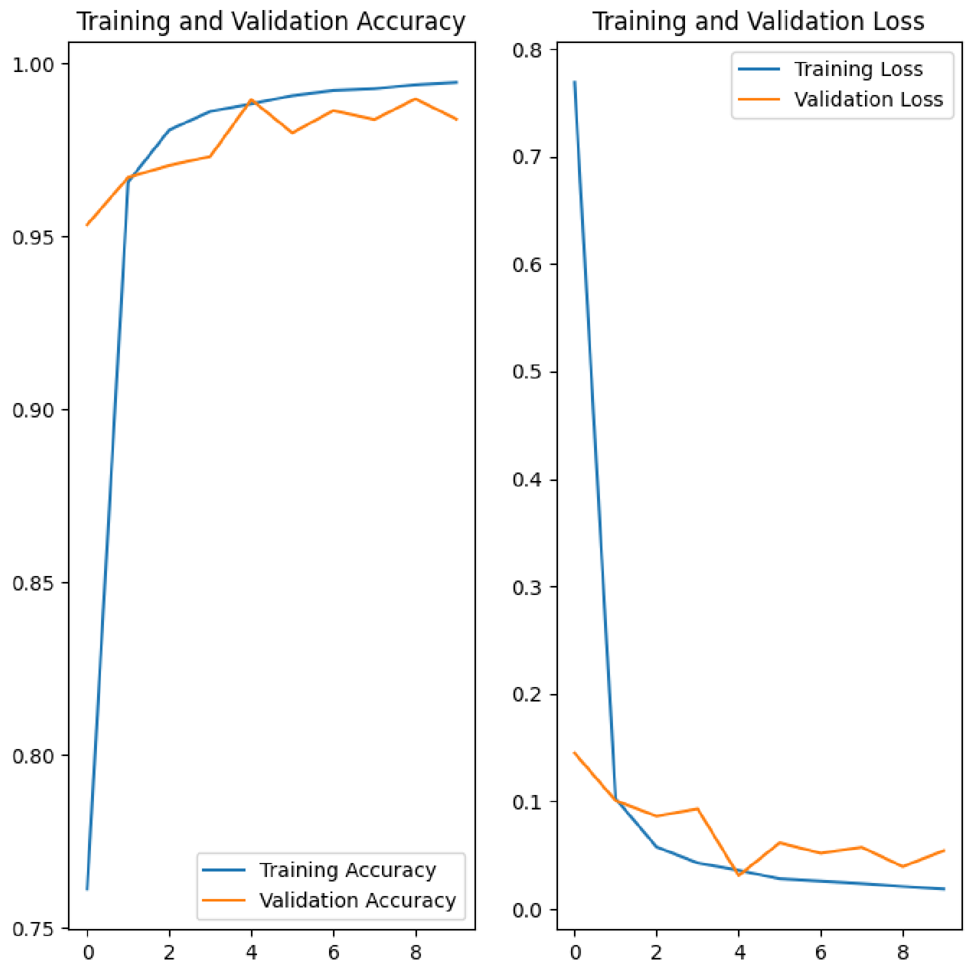


Figure 2: This graph shows the training and validation accuracies and losses for the initial run of our model.

which means that the model is heavily relying on the training data, including some specific training details that might not be present in the validation data. Some notable ways to minimize this would be to add more data, which is what the full model would entail.

Grey scaling the data, as mentioned earlier, means that the images in the model are converted to a black-and-white variant. By doing so, we are removing all the significant colors in the images. The point of this model is to determine if the color quality of the images is a strong factor in the accuracy of the model. Comparing the gray-scaled images of the 300 images to the regular images, we see a small drop in the accuracy. More specifically, there is about a 3 percent decrease in the training accuracy and about a 4 percent decrease in the validation accuracy. This indicates that the color of the images is a factor that affects the classification of the American Sign Language images, but due to these small decreases, it is not too important of a factor. At the very least, it means other factors such as shape, position, and angle are more important than color. These features are still maintained within the grayscaled images.

One change to the input variable that did not yield very accurate results was when we applied binary thresholding to the data. Despite the sample size of 300 being greater than some of the variations that used a sample size of 100, it is the lowest-performing model in both accuracies, with a 70 and 40 percent

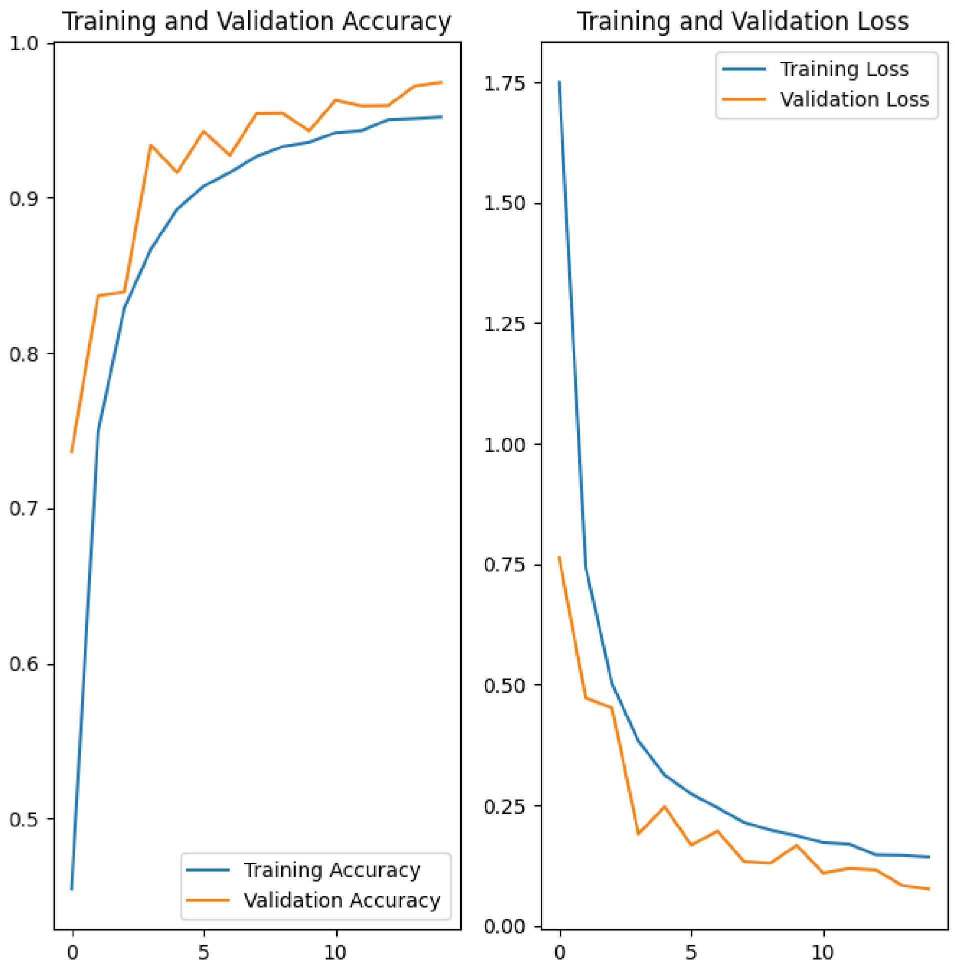
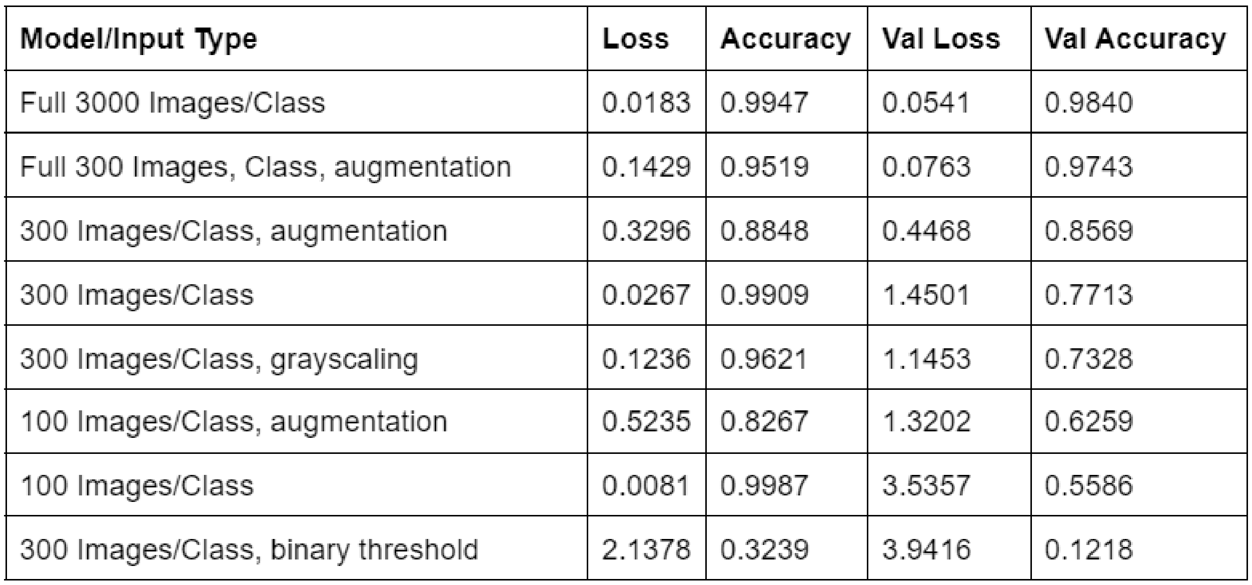


Figure 3: This graph shows the training and validation accuracies and losses for the augmented run of our model.

drop from the second-worst-performing variation of 100 images. A possible explanation is that the transformation made the sample size much poorer. Binary thresholding, as a reminder, takes the colors of an image and classifies it into one of two colors depending on the brightness and value. This gives an overall clear outline of the hand sign, but by doing so, the depth and shape of the images are lost in the process. This leads the classification to struggle to recognize them. In this case, we made the images much simpler, but by doing so, we lost some of the features obtained in each image that the model needs to be more efficient. Like with the grayscaling, color was a slight factor in the accuracy of the mode. With binary thresholding, a major feature in the model is the depth of the image and we cannot have an image of just the silhouette as shown in this variant.

Ultimately, our results indicate a high sample size is better for the model. Still, the accuracy does not drop significantly, so if we're required to speed up training, dropping samples would be okay if one would like to sacrifice a little accuracy. Grayscaling the data indicates that the color has a small effect on the accuracy of the model, but it is not an overall significant one. Conversely, the depth and detail of the images are very important in classification, and attempting to simply classify the images by transforming them to consist of only two colors only makes this attempt more inaccurate and difficult to classify. These augments are important to consider, as any future im-



Model/Input Type	Loss	Accuracy	Val Loss	Val Accuracy
Full 3000 Images/Class	0.0183	0.9947	0.0541	0.9840
Full 300 Images, Class, augmentation	0.1429	0.9519	0.0763	0.9743
300 Images/Class, augmentation	0.3296	0.8848	0.4468	0.8569
300 Images/Class	0.0267	0.9909	1.4501	0.7713
300 Images/Class, grayscaling	0.1236	0.9621	1.1453	0.7328
100 Images/Class, augmentation	0.5235	0.8267	1.3202	0.6259
100 Images/Class	0.0081	0.9987	3.5357	0.5586
300 Images/Class, binary threshold	2.1378	0.3239	3.9416	0.1218

Figure 4: This table shows the accuracies and losses of all the variants of the model.

ages we would like to use as data might have to take into account some of these features to yield a better classification result.

One other factor to mention is the losses of the variations. We see this in the two figures with the Loss Graph that measures and tracks the training and validation loss, and also the table that records the losses. The Training loss is a measure of how well the model is performing on the training data by measuring the error between the predictions and the actual training data. The validation loss is a similar metric, but it is on the validation data instead. Ideally, we would like to minimize the overall loss while maximizing the overall accuracy. In our data, the variation with the lowest training loss is interestingly enough the 100 images with no alterations. The lowest validation loss is the original 3000 images, followed closely by the augmented 3000 images. With this knowledge, it seems that the original 3000 images is the best variant of the model, but as mentioned before, the augmented images reduce the model overfitting while suffering minimal decreases in accuracy and increases in losses, so this model is also effective in order to generalize our model.

Conclusion and discussion

Throughout our experimentation on creating an image classification model , we are able to create an accurate model that has a high accuracy in identifying the letter associated with an ASL symbol. The entire process started with the extraction of data, processing the data by analyzing the attributes of an image. We then split the images into a training and testing set. Using that training set, we establish a convolutional neural network that we use for our testing data. And with the given results, we evaluate our model. We also alter the images by various methods, such as reducing the sample size or changing the images directly, to help provide a visual approach of what features best affect the accuracy of the model. We have 2 possible variations of models: one with the lowest loss and the highest accuracy with regular images, but also the augmented images which reduces the overfitting of the model, making it more generalizations.

It is important to recognize that despite our high accuracy, it is not 100 percent. It is important to recognize this distinction because when it comes to something akin to a health-based classification model that identifies the presence of a disease in a patient, false positives and false negatives would be detrimental to a patient. Though false positives and false negatives don't have that severe of an effect in our ASL recognition, it is important to consistently improve our model to ensure a 100 percent accuracy.

For future work, we would like to apply our neural network to more complex examples. For example, our model is trained and tested on static images, so it would be interesting to identify the effectiveness of the model on a more dynamic environment, such as a video. Additionally, our model identifies individual symbols, but it would be interesting to see how it would fare with complete words, as when it comes to general sign language, practitioners have certain movements dedicated to words.

References

Dataset:

https://www.kaggle.com/datasets/grassknoted/asl-alphabet/

Literature Review:

Vedak, O., Zavre, P., Todkar, A., & Patil, M. (2019). Sign language interpreter using image processing and machine learning. International Research Journal of Engineering and Technology (IRJET).

Atwood, J., Eicholtz, M., & Farrell, J. (2012). American sign language recognition system. Artificial Intelligence and Machine Learning for Engineering Design. Dept. of Mechanical Engineering, Carnegie Mellon University.