Sign Sight: A Real Time ASL Recognition System

Abbey Liu, Dalvir Singh

May 3, 2022

1 Abstract

Sign language is a complete, natural system which uses movements of the hand and face to communicate, rather than with spoken words. Those who are mute, deaf, or hard of hearing make up the largest group of sign language speakers. There is a large communication barrier between those who use spoken language and those who use sign language, in which most daily activities and tasks assume that a person is able to communicate with spoken language. In addition, there is a general lack of awareness and shortage of available interpreters, which further emphasizes this barrier. In this project, we aim to use artificial intelligence and machine learning methods to break the communication barrier by creating an image classifier than can predict the meaning of various hand gestures. We focus on American Sign, a variety of sign language used in in the United States and in English speaking areas of Canada. After training and testing various model architectures, we implement one of the models into a web application so that it is easy to use and access for the general public.

2 Introduction

Human language is a powerful tool that allows us to communicate our ideas, share our thoughts and feelings, and express our viewpoints. However, most of our daily activities assume that a person is able to use spoken language, which plays a critical role in almost all human interactions. Therefore, it is easy to overlook the struggles of people with hearing disabilities such as the mute, deaf, or hard of hearing, who cannot use spoken language. Instead, these individuals use sign language, which utilizes movements of the hands and face to communicate. However, this causes them to face a large communication barrier between signed and spoken language. This results in sign language speakers becoming a segregated minority in society. Moreover, there is an overwhelming lack of general awareness for sign language, which further increases the language barrier dividing spoken and signed language.

For this project, we focused on American Sign Language (ASL), which is the variety of sign language commonly used in the United States and English speaking areas of Canada. There is a limited amount of laypeople who are proficient in ASL, so typically those with hearing disabilities may utilize an interpreter. However, there is a large gap between the number of interpreters and the population with a hearing impairment. Although the use of interpreters can help facilitate communication, it often makes them overly reliant or dependent on the interpreter. In this project, we aimed to create a ASL Recognition System that can be used as a tool to bridge the gap between hearing-impaired community and those untrained in ASL. Ultimately, we hope our project can empower hearing impaired population to be more independent and improve the comfort and ease in everyday human interactions.

3 Related Works

We are not the first to recognize the potential benefits of a sign language recognition system to people with hearing disabilities. Previous literature has tackled the problem of sign recognition in dynamic videos and static images using a variety of machine learning methods and algorithms.

A popular approach to solve this problem is to use a Hidden Markov Model (HMM), a probabilistic model that is based on the statistical Markov model to recognize the relationship between observable events that depend on internal factors. Previous literature has shown success with using an HMM model to recognize sign languages. Thad Starner and Alex Pentland [1] created a real time system that interpreted ASL using Hidden Markov Models. The system tracked a person's hand by their shape, orientation, and gesture which served as the input to the HMM model to recognize the signed gesture. The research group employed two experiments. The first experiment tracked hands wearing colored gloves while the second experiment used hands without gloves. Ultimately, they found their system achieved a higher accuracy when tracking hands with gloves.

With the rising popularity of neural networks and deep learning, more recent works explore how deep learning approaches compare to those of the past. The most traditional approach to solve computer vision related problems are convolutional neural networks (CNN) which utilize the features of an image to aid in classification. Machine learning models with CNN-like architectures have shown much success in this topic.

Simming He [2] proposed a deep learning approach which consisted of using a combination of methods. He used Faster R-CNN, which has an embedded region proposal network (RPN) module to locate the hands in the images. These location results are fed into a 3D CNN feature extraction network and sign language recognition framework based on long- and short- term memory (LSTM) coding and decoding network.

Rabeet Fatmi et al. [3] created a system that utilized wearable motion sensors and Artificial Neural Networks (ANN) and Support Vector Machines (SVM) models to recognize words in ASL. The research showed high accuracy

in ANN compared other machine learning approaches.

Lionel Pigou et al. [4] built a real time system to recognize 20 Italian gestures using Microsoft Kinect, CNN, and GPU acceleration. The system utilized a CNN model to extract features from the frames and an Artificial Neural Network for classification.

4 Methodology

Solving machine learning problems can be categorized into the following steps: task definition, data collection, exploratory data analysis (EDA), machine learning modeling, model testing, and deployment.

Task Definition: We wanted to create a ASL Recognition System that can accurately classify American Sign Language gestures in real time and on static images. To achieve this objective, we decided to train multiple machine learning models and pick the best model as the engine for our recognition system. Given an image of American Sign Language gesture, our system should classify the gesture into one of 24 alphabetical letters. Thus, we can define our task as a mutually exclusive, multiclass classification problem, where each image can only be classified as one letter. Note that in ASL, the gestures for J and Z contain movement. So, we did not include these in our classification task definition.

Data Collection: To train a robust machine learning model that works well in practical applications, it is important to collect a large dataset that is representative of the general problem you are trying to solve. Due to the limited time of the assignment and the lack of expertise in ASL, we could not create our own comprehensive dataset. Instead, we had to make the best out of the publicly available datasets for this problem. We used the popular Sign Language MNIST dataset [5] that consisted of 27,455 training and 7,172 testing gray-scale images of size 28 x 28. Each labeled image represents an American alphabet letters except for J and Z, which require moving gestures that cannot be captured in static images.

Exploratory Data Analysis: EDA is an important step when creating machine learning models because it provides insight into the raw data and allows you recognize the relevant and irrelevant features. The Sign Language MNIST dataset comes from originally colored images of the alphabetical character gestures in ASL, made by different people against various backgrounds. This resulted in 1,704 original images. Synthetic data augmentation was applied to each original image with ImageMagick, creating a larger quantity of image data. Some examples of augmentation used are various filters and changes in brightness, pixelation, and rotation. The images were then converted to gray-scale, cropped to only contain the hand gesture, and then resized to 28 x 28 pixels in size. Lastly, the full dataset was split into 27,455 images for training and 7,172 images for testing. We accessed the dataset through Kaggle, where the images are saved as pixel values in CSV files. Figure 1 provides an image set of the 24 alphabetical characters with the original color. Figure 2 shows the

same image set in Figure 1, but after data augmentation and preprocessing. So, Figure 2 is a visual sample of what is actually in our dataset.

Figure 1: Sample of ASL image set, with original colors

Figure 2: Sample of ASL image set, after gray-scaling and resizing

When we further analyzed our dataset, we found that each alphabetical character had roughly 1,000 samples in the training split. So, our training data is roughly balanced since each class has about the same amount of presence and representation. We found the distribution of the classes to be similar in the testing split. Figure 3 shows a bar chart of the alphabetical character frequencies. Note that the index numbers 0-25 correspond to the letters A-Z, where indexes 9 and 25 have no images because they are the letters J and Z.

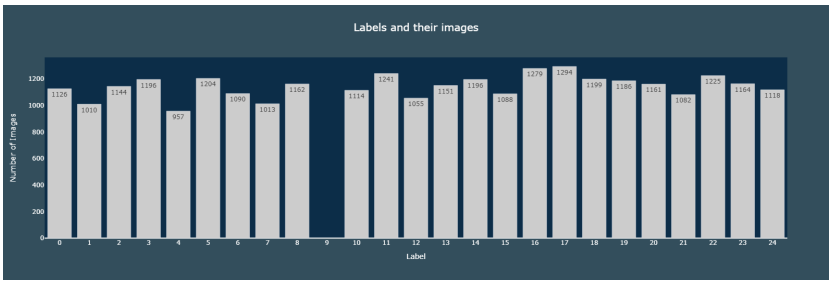

Figure 3: Bar chart of alphabetical character frequencies in training split

Upon further visual analysis, we found that our dataset is extremely clean. Although the creators of the dataset had different people make ASL gestures and used varying backgrounds, the variety between the images was minimal. All the pictures feature the hand gestures at the same viewpoint and orientation, and there were no objects obstructing the view of the hand. Moreover, the images were cropped to only feature the hand, so there was no body or face present in the images. When two people are communicating through sign language, often times they can see at least each other's upper body. This is because some gestures in ASL include movement or facial expressions. Since our dataset does not contain these elements, the ASL hand gestures in this dataset have been taken out of their real world context. In addition, gestures for words or phrases are more commonly used in American Sign Language where alphabetical characters are only used when a gesture for a particular word doesn't exist, such as spelling out someone's name. We tried to find ASL image datasets that had more real world context, but were unable to find any.

Thus, our dataset is very clean, to the point that it can be considered a bit of a limitation. The provided images only show the gestures out of context, and have very minimal backgrounds. Although the dataset creators altered the original images, this synthetic data augmentation will not create the same level of diversity as a dataset made of completely different original images. Despite the usual case of messy data, our dataset is so clean that our machine learning models may suffer when they try to predict on images outside of the dataset. However, we believed that there is still some value in this dataset, and decided to continue using this dataset for the project. In future work, we hope to search for or curate an ASL dataset that contains images of the upper body of a ASL user and features gestures of words and phrases rather than only alphabetical characters. This dataset would fit our task better, since it would show ASL gestures in a real world context.

Machine Learning Modeling: To find the best model, we employed various supervised machine learning techniques such as CNN, ResNet50, InceptionV3, and YoloV3 to find the one that works best for this problem. All models for this project were implemented using Keras and Tensorflow Python libraries, and

were trained using Google Colab. Note that for all models, we use the training and testing splits provided in the raw dataset.

CNN: Convolutional neural networks are a classical approach used to solve computer vision problems. CNN is a neural network model that consists of multiple convolutional layers, flatten layers, dense layers, and finally a softmax activation function that provides the classification. CNN is a powerful because its convolutional layers perform dimensionality reduction, which can automatically detect and learn the important features to classify an image without any human supervision. Additionally, they can detect features anywhere in the image, whereas its predecessors had difficulties learning features that were in spatially different locations [6]. For this project, we built a small CNN from scratch. It includes three convolutional layers, followed by one layer for flattening, two dense layers, and finally a softmax activation function. Compared to the other models we trained and tested, our CNN is the simplest and smallest model architecture.

ResNet50: Deep learning uses large machine learning models to solve more complex problems than with simpler or more traditional models. However, a large problem in deep learning architectures is the vanishing gradient problem. In traditional or vanilla models, the output of each model layer is fed directly into the next consecutive layer. By passing through so many layers during backpropagation, the gradient may become too small for effective model training. Residual Networks, also known as ResNets, are a particular type of CNN that use residual network mapping blocks in their architectures to avoid the vanishing gradient problem. These mappings contain skip connections, which allow the model to skip layers when needed. This protects deep learning models from encountering the vanishing gradient problem. ResNets were first proposed by Microsoft researchers in 2015 [7]. There are several different deep architectures that utilize the general idea of ResNets, but vary in the amount of residual blocks used. We implemented ResNet50, which features 50 residual blocks in its architecture. Due to its sheer size, the model is much larger and more complex than our initial CNN model. In addition to this, another difference is in the initial weights used for the ResNet50 model. The Keras implementation of ResNet50 has pretrained weights, which were originally trained on the ImageNet, a well-known computer vision dataset. In 2015, it contained about 1 million training images to classify into 1,000 classes (Now, the dataset has expanded around 14 million images). We used these pretrained weights to fine-tune our model for our ASL classification task. The use of pretrained weights is known as Transfer Learning, which provides many benefits. Transfer Learning lets us take weights from one problem and reuse them for a similar problem. It can also help the model converse faster than if it had started training on initially randomized weights.

InceptionV3: Similar to ResNet50, Inception is another type of deep learning CNN architecture. However, instead of focusing on avoiding the vanishing gradient problem. Inception's main goal is to be more computationally efficient

by allowing for deeper networks without increasing the number of parameters. InceptionV3 [8] was introduced in 2015 as the third documented version of the Inception architecture. It boasts that it uses less than 25 million parameters. While this may seem like an enormous number, other models have well over 60 million parameters, so compared to other models InceptionV3 has much less parameters. Some details about InceptionV3's architecture is its use of Label Smoothing, which is a regularization technique that puts some noise into the labels to account for possibly mistakes in the dataset. The architecture also features an auxiliary classifier and batch normalization to improve convergence speed. Like ResNet50, the Keras implementation of InceptionV3 also has pre-trained weights. It was pretrained on the same version of ImageNet as ResNet50, with 1 million training images to classify into 1,000 classes.

YoloV3: Yolo is one of the most popular real-time object detection algorithms that uses the concept of you only look once (yolo) to perform object detection [9]. The named was coined for its ability to detect multiple objects in only one pass. Traditional object detection systems pipeline multiple neural networks to classify images that contains a specific object, and then use another network to create the bounding boxes. Unlike its predecessors, Yolo uses a single neural network to classify and predict the bounded boxes for the detected object. This makes it much faster than other traditional approaches, and suitable for real-time detection. Yolo splits the images into a square grid, each cell is responsible for detecting whether the cell belongs to one of the objects so each cell needs to predict B bounding boxes and confidence score for each box. The classification score ranges from 0.0 to 1.0 with 0 being the lowest confidence level and 1 being the highest. The bounding boxes are made up of 5 components, x and y coordinates (location of the center of the predicted box), confidence level, width, and height. In this project, we attempted to train a YoloV3 [10] model from 2018, since we found a Keras implementation and dataset of ASL alphabetical characters that were more in-context (images that include the arm and messier backgrounds) than our current dataset. In addition, the YoloV3 comes with pretrained weights, which were originally trained on Darknet. Darknet is an open-source neural network for object detection. However, we ultimately faced several issues with when implementing YoloV3 and were not able to successfully train and test the model. Specifically, we ran into dependency issues between Keras and Tensorflow. YoloV3 required downgraded versions of Keras to run, but then this required a downgrade of Tensorflow as well. It was difficult to achieve a combination of the correct versions to get the model training. Even when we were able to train the model, the resulting metrics and loss values were much worse compared to the other three models we had already implemented. Eventually, we decided to discontinue our progress on YoloV3 so we could focus on analyzing the other three models and deploying the web application and real-time detection software.

Model Evaluation: In this phase of the development, we trained and tested the different model to find the model that works the best for this task. We

used four metrics to evaluate a model: accuracy, recall, precision, and F1 score. In most tasks in supervised learning, accuracy alone does not provide enough information about a model's performance. Precision is a proportion of how many samples for a class are accurately predicted versus the total number of samples the model has predicted as that class. Recall is a bit different, and is a proportion of how many samples for a class are accurately predicted versus the total number of samples in that class. F1 score is the harmonic mean between precision and recall, and represents the two metrics as one value. Thus, to get a more informative results, in the Results section we report the accuracy, recall, precision, and F1 score of the CNN, ResNet50, and InceptionV3 models. For each metric's equation, consider a confusion matrix M, where M_{ij} is the number of play sequences with the ground truth label i that is predicted as class j.

$$precision_i = \frac{M_{ii}}{\sum_j M_{ji}} \tag{1}$$

$$recall_i = \frac{M_{ii}}{\sum_j M_{ij}} \tag{2}$$

$$F1score_i = \frac{precision_i * recall_i}{precision_i + recall_i} \tag{3}$$

Model Deployment In practical applications, it is best to deploy trained machine learning models into user facing interfaces. We created two user interfaces for the user to leverage our AI engine to make predictions. One is predicts based on static images, while the other predicts with real-time input.
We first developed a simple web application that can use our AI engine to classify ASL hand signs from static images. We designed a simple web application using Django, Bootstrap, CSS, and HTML. Through the web application, the user can upload a static image of a ASL hand sign. The application will then preprocess the image and forward it to the AI engine to make the classification. The results are returned to the user in the form of a predicted label and probability/confidence of the predicted sign.

Additionally, we utilized OpenCV to create a real-time system. OpenCV accesses a laptop camera and processes each frame one at a time. For each frame, we resized the image into a 28 x 28 image, converted into a gray-scale, and then transform it into an array that can be processed by our AI engine. Initially, our model underachieved when evaluating the ASL signs in real time.

To achieve better performance, we used OpenCV to first crop the inputted image, so only the user's hand would be processed for the prediction. We created a fixed green box in the frame to mark the region in that will be used by the model to make the prediction. By limiting the ASL hand gestures to this box, we can make the real-time input more like the training and testing data from our dataset. The image in the cropped box is then preprocessed as previously stated, with resizing, converting to gray-scale, and then transforming into an array. Finally, the preprocessed image is fed into the model for the classification.

5 Results

We evaluated the models on the accuracy, recall, precision, and F1 score on the test set. Figure 4 shows the testing results for each model. All the models performed well on the test set achieving almost perfect values. We acknowledge that this is probably due to our training data, which is very clean (See Exploratory Data Analysis and Discussion sections for further details). From these metrics, it appeared that all three models were good candidates to serve as the AI Engine. We ultimately chose CNN as the model that would be deployed and integrated into the OpenCV real-time system and web application. We chose CNN because it is the simplest model. Based on our results, it achieves similar metrics to the deep learning models so there is no need to use a larger, more computationally expensive model.

Model	Accuracy	Recall (Weighted)	Precision(weighted)	F1 score(weighted)
CNN	0.9911	0.9889	0.9918	0.9903
InceptionV3	0.9838	0.9917	0.9736	0.9826
ResNet50	0.9976	0.9976	0.9976	0.9976

Figure 4: Resulting metrics for each model on testing split data

After looking at the resulting metrics, we then observed some concrete examples from the testing split. Since our dataset is so clean and uniform, we found that all models do very well on ASL gestures with shapes that distinctly distinguish it from all other gestures. Figure 5 shows two visualized examples from the testing split, with predicted and ground truth labels. Predictions for both images were done by our ResNet50 model. On the left, we can see that our model correctly predicted the label B. The ASL gesture for B is very distinct from other gestures because it is the only hand sign that shows the palm with four straightened fingers, which makes it easier for the model to identify (Refer to Figure 1 for a clearer image of the gesture). Meanwhile, all other gestures are in the general shape of a fist or place the hand in a different shape. Either way, all other gestures are distinctly different in shape compared to B, so our model is able to accurately identify it.

Despite our high metrics, our model can still make mistakes. From analyzing the visual testing results, we can see that most of the mistakes are on gestures whose general shape is a fist. While there are details between the gestures that would help a human viewer distinguish them apart from one another. The images in this dataset contains are very small, with only 28 x 28 pixels. From Figure 5, we can see that the pictures are rather pixelated due to the resizing. This blurs the details that would help distinguish one sign from another. While some gestures have a distinct hand shape, like B, W, V, and F, others like A, E, M, N, and S all have a general fist shape. With less distinct details, the

mistakes our models make tend to involve these four letters. Figure 5's right image gives a concrete example of this, where the model predicted the letter to be S, but the ground truth label is actually N. Humans can have trouble distinguishing between these hand signs in the dataset as well, which makes it reasonable to assume that a machine learning model would face a similar challenge.

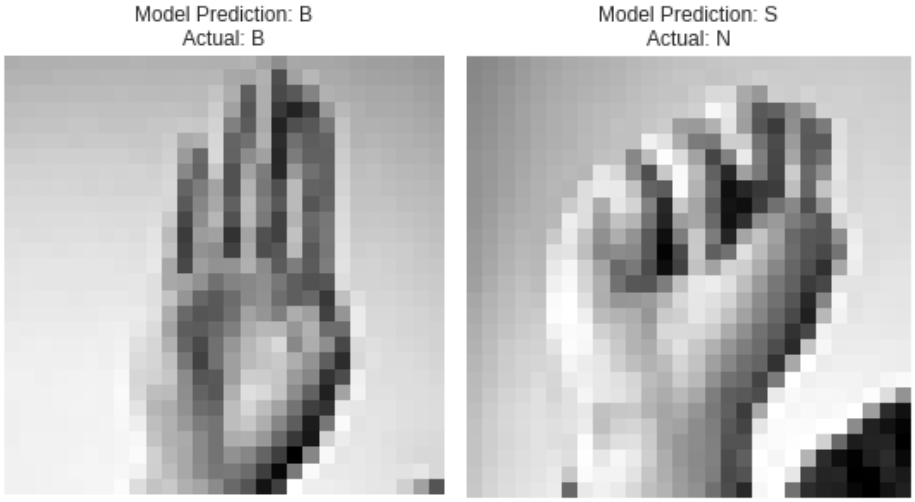

Figure 5: Sample images from testing split, with predictions made by ResNet50. Left: correct prediction of B. Right: Incorrect prediction of S on ASL gestures for N

Figure 6 shows the home page (left) and classification result page (right) of the developed web application that can be used by users to predict ASL hand signs on static images. A user can take a picture of their hand making a ASL gesture, and upload it to the web application to see if the AI engine correctly classifies their sign. The classification result page shows the predicted label and the model's probability/confidence of the classification for the uploaded image. The example shown in Figure 6 uses an image from our testing split. We can see that the model is very confident about its prediction, giving a confidence score of nearly 100%. In addition, this is a correct prediction, where the ground truth label from the data is the letter B.

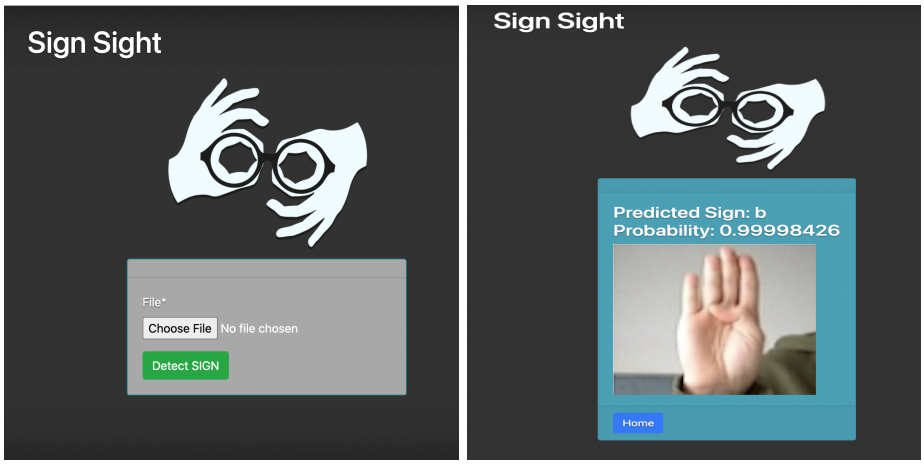

Figure 6: Static Images Web Page, home (left) and classification result (right)

To evaluate the web application on real use cases, we compared the performance between images from the test set with images of our own hand signs. Observe in Figure 7 how the model's confidence score for our hand sign (left) is significantly less than image from the test set (right). Both images in Figure 7 give a correct prediction, however the model is much less confident when predicting our image versus predicting images from the test set. Moreover, to get the results in the figure, we had to make many modifications to how we took the picture to achieve a correct prediction. Some modifications include placing a white piece of paper to prevent any noise in the background from appearing in the image, zooming in on our hand, and making sure we took the picture in a similar orientation to the dataset's hand orientation. This low confidence score on our inputted static image is likely due to our dataset being too clean and out of real world context for our model to make a confident prediction on slightly different data.

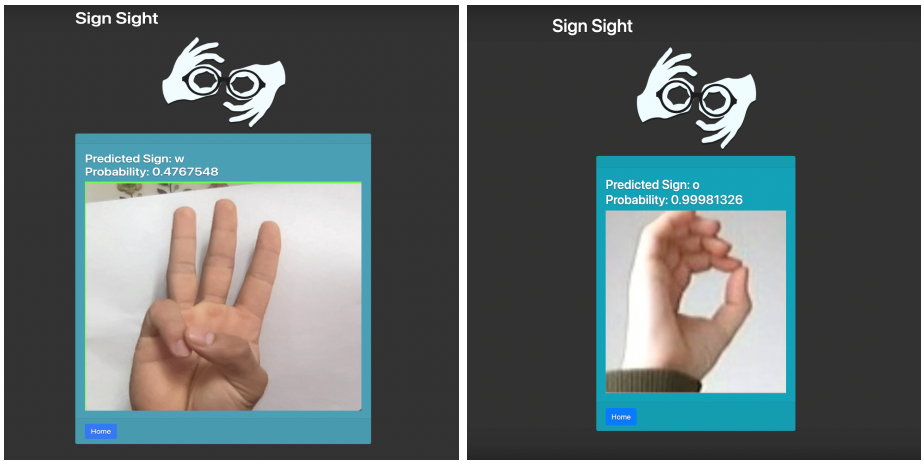

Figure 7: Static Images Web Page, testing on real cases

Figure 8 demonstrates the OpenCV Real-Time System that utilizes the user's laptop camera and creates a video panel that so the user can make real-time ASL hand signs and the system will predict the signs one at a time. We observed the real time system worked well for the following signs: W, V, T, L, and F but did not work as well for other signs. One possible reason for this is that these particular hand gestures are very distinct, and there are no other gestures in our dataset that closely resemble them. On the other hand, other gestures such as A and S look very similar to each other. Thus, the model is often unsure of which letter to predict. Additionally, we noticed that we had to make similar modifications to our real-time input. The model would pick up on subtle changes in hand orientation, shape, and movements impacted the predicted sign from the model. Thus, to achieve the correct prediction, we had to fine-tune how we put our hands in front of the camera for results. This is not ideal for this application, which should be user friendly and easy to use. We hope to improve our models and web application in future work.

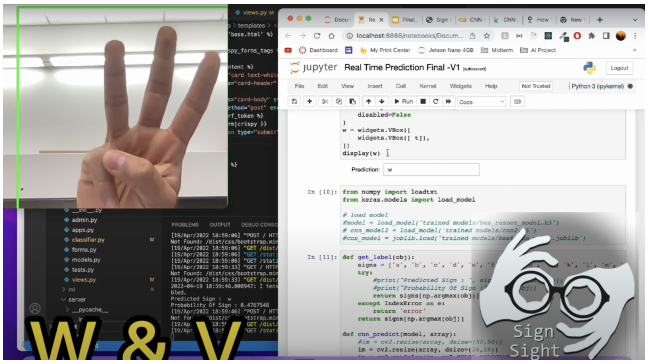

Figure 8: OpenCV Real Time System

6 Discussion

From our results, we believe that our dataset is too clean and out of context to produce reliable results for our application. Our data features small gray-scale images that only show the ASL hand gesture. Unlike in the real world, the rest of the person's body is not shown and there are no background objects that are noisy or obstructing the hand. This puts the hand gestures out of context, and causes our models to be very sensitive to noise. Our models only give confident predictions on images within the dataset, while suffering when given other images to predict on. This is seen in the web application for static images, where our CNN model performs well unseen images from the testing split. However, its performance drops greatly when predicting on unseen images outside of the dataset. In addition to this, the web application for real-time input is also very sensitive. The model would change its prediction with subtle changes in hand orientation, shape, and slight movements. These results show that there is a large gap between the dataset used for training and real world input. Despite the dataset's creators augmenting their original images, synthetic data augmentation is not a replacement for gathering more original images. Although the creators boast over 30,000 images in total, these all still come from 1,704 original images. Changing them synthetically does not give the same diversity of 30,000 distinctly original images, especially since they mainly focused on filters and changing the brightness or contrast of the images. This means that other aspects like the camera's viewpoint of each hand gesture and the size of the hand in the images is same as the original images. These aspects are greatly variable when we try to provide our own images, which caused the model to become less confident in its predictions. Furthermore, our images from the dataset are so small that they can appear pixelated and less detailed. This can hide the distinguishing features of some gestures and make it hard for the model to accurately predict on some dataset images. However, this problem is further amplified when using our images. This pixelation, combined with the noise from our own pictures causes the model to become even less confident when predicting on our own images.

One possible solution to this would be find a dataset that is more robust and in-context to the real world application of American Sign Language. However, we were unable to find an available dataset of this kind during the duration of this project. Since neither of us are fluent in ASL, it would great to gather expert domain knowledge from ASL interpreters. By taking pictures of multiple ASL interpreters signing, we could potentially create the dataset needed for this project. The pictures can feature the upper body so that the sign is in-context with the real world. Moreover, pictures of the same signs could be taken at different angles and with varying backgrounds to further increase the diversity of the dataset. Another detail is that the dataset should contain gestures for common words and phrases, so that the labels used in the dataset more appropriately match everyday ASL conversations rather than only alphabetical characters. The dataset should also contain as many original images as we can

obtain, so that the dataset is truly diverse.

To take the dataset discussion even further, perhaps image is not the most ideal representation of American Sign Language since there are multiple gestures that require movement. Hence, maybe a video dataset would be even more suited for this problem in the long-term. However, this would probably be more costly (both in data collection and computational data processing) than an image set. We would also need to find and train models for object detection and tracking. So, the short-term improvement on data would be to find a dataset that is more robust and features gestures in context. The long-term improvement would be to find a dataset that can also capture movement.

Despite the many limitations of the dataset used, we believe that our results shows that there is potential in further pursuing this project. While the web applications and their backend models have much room for improvement, the basic idea of creating an easily accessible application to bridge the gap between signed and spoken language with machine learning is within reach with today's technology. However, for this to be possible, there is a need for a ASL dataset that shows gestures in context and is more robust than our current dataset.

7 Future Works

In the future, we hope to work finding or creating a dataset to better fit our classification task. Specifically, a dataset that shows the gestures in-context of the real world and shows the gestures at different angles or camera viewpoints. If possible, a dataset that also contains words or phrases is preferable, since those gestures are used more often in common ASL conversations. While searching for more applicable data, we can also try other models besides the neural networks trained and tested in this project. For example, we can try traditional machine learning models like K Nearest Neighbors, Logistic Regression, Decision Trees (and Random Forest), and Support Vector Machines. Furthermore, we can also return to our Yolo model, and try to find a different version and implementation. The literature makes this model seem very suitable for our classification task, so it would be great to implement it and compare the results to the other models we have used so far.

References

[1] Starner, Thad and Alex Pentland. "Real-time American Sign Language recognition from video using hidden Markov models." (1995).

[2] He, Siming. "Research of a Sign Language Translation System Based on Deep Learning." 2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM) (2019): 392-396.

[3] Fatmi, Rabeet et al. "American Sign Language Recognition using Hidden Markov Models and Wearable Motion Sensors." Trans. Mach. Learn. Data Min. 10 (2017): 41-55.

[4] Pigou, Lionel et al. "Sign Language Recognition Using Convolutional Neural Networks." ECCV Workshops (2014).

[5] Tecperson. "Sign Language MNIST." Kaggle, 20 Oct. 2017, https://www.kaggle.com/datasets/datamunge/sign-language-mnist.

[6] Tatan, Vincent. "Understanding CNN (Convolutional Neural Network)." Medium, Towards Data Science, 23 Dec. 2019, https://towardsdatascience.com/understanding-cnn-convolutional-neural-network-69fd626ee7d4.

[7] He, Kaiming et al. "Deep Residual Learning for Image Recognition." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 770-778.

[8] Szegedy, Christian et al. "Rethinking the Inception Architecture for Computer Vision." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 2818-2826.

[9] Aggarwal, Ani. "Yolo Explained." Medium, Analytics Vidhya, 7 Jan. 2021, https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31.

[10] Redmon, Joseph and Ali Farhadi. "YOLOv3: An Incremental Improvement." ArXiv abs/1804.02767 (2018): n. pag.

Links to Code References/Materials

We referenced and modified the code from the following links.

1. CNN
    - https://www.kaggle.com/code/madz2000/cnn-using-keras-100-accuracy
2. ResNet50
    - https://www.kaggle.com/code/lordkun/go-deeper-with-resnet-sign-language-99-acc
3. InceptionV3
    - https://www.kaggle.com/code/mfaaris/transfer-learning-for-sign-language-mnist
4. YoloV3
    - https://blog.roboflow.com/training-a-yolov3-object-detection-model-with-a-custom-dataset/