SINGAPORE MANAGEMENT UNIVERSITY

ISSS610 Applied Machine Learning

American Sign Language Alphabet Classification

Group 15

Lim Pek Liang Arnol

Tan Yan Ru

Wang Zhifei Celia

Wong Chun Keet Brian

Wong Sook Xian Sophia

Contents

1 Introduction

Sign language is used by 70 million deaf people worldwide as a mean to communicate (Kozik, 2020). Singapore Association for the Deaf estimates that there are 500,000 individuals with varying degrees of hearing loss in Singapore (Chua, 2019), which accounts to about 9% of the population. Although sign language meant to bridge communication and allow the deaf community to express themselves and connect with the society, communication barrier between the deaf community and the hearing majority remains prominent as sign language is only widely understood within the deaf community. Learning and practicing sign language is not common among the hearing majority.

2 Objectives

This project aims to discover the crucial factors in developing a practical, applicable model for real-life sign language recognition – going beyond relying on just model accuracies/loss of test-train dataset splitting – and then overcoming these challenges accordingly. Various Machine Learning classification methods are employed to recognize alphabets signed in American Sign Language (ASL), the sign language branch that is widely used in Singapore and the rest of the English-speaking world.

3 Datasets

3.1 Dataset Preparation

Dataset used in this report is a Kaggle dataset (Muvezwa, 2019) that consists of over 70,000 coloured, RGB images with a resolution of 320x240 pixels. In this set of data, letters "J" and "Z" are excluded from our scope as they contain motion. With minimal pre-processing, we used this dataset to perform preliminary exploration, as illustrated in Approach 1.

Due to poor model performance and learnings from model evaluation, we extracted hand region from the images to minimise noise created by background features such as like the person's clothes, arms or even face. As the cropping process requires a lot of time and memory for 70 thousand images, we reduced our dataset to 1680 images, consisting of 10 images for each alphabet signed by 7 people. This subset of data was used was used in Approach 2 and 3.

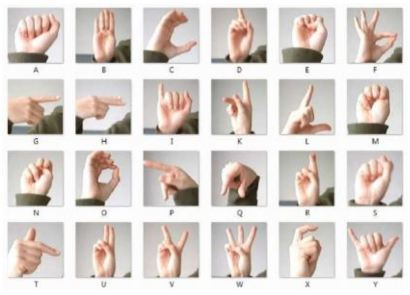


Figure 1: RGB ASL Sign Language Dataset (320x240)

3.2 Training, Validation and Test Data

The entire dataset was split into 90% training set, which was further split into 80% training and 20% validation, and 10% test set for model evaluation. Another test set obtained from a test video was used for final model evaluation.

3.3 Test Video

While the datasets provide an environment for training and testing the models, the images in these instances are mostly clean, with plain background and clear view of the gestures. While data augmentation through rotation and skewing might help provide some level of disturbance, testing these models against a real-life video clip – with different backgrounds and conditions – would determine the practicality of the trained model. If the model performed poorly at detecting gestures at first, then further feature engineering might be required in subsequent training iterations.

A YouTube video clip titled "Timothy Williams ASL Homework #1" (Williams, 2017) was used as the application video clip for model evaluation. This video consists of a total 28 seconds of content and has a frame rate of 30 frames per second. Key frames of gestures were extracted using OpenCV. To extract the right frame, we first trimmed the video to get rid of the irrelevant parts and utilised cv2.CAP_PROP_POS_MSEC to keep track of the video position in terms of timestamp and adjust the frame rate to give the best result. We extracted the trimmed video at an interval of every 0.5 second to get 27 images that were used subsequently to assess model performance.

Figure 2: YouTube video screenshot – "Timothy Williams ASL Homework 1" sample frame (Signing Y)

4 Overall Architecture

ASL images from the datasets are firstly fed as input for data preparation. The images were first scaled down to a lower resolution, split into training and testing set, before conversion into *.npy* files as for model training.

The various ML models were trained and fine-tuned against these prepared datasets. Both accuracy on testing set, and performance on the video clip frames, were used to evaluate these models. Based on the analysis on performance of these initial models, the data preparation stage was re-visited again to improve model performance by adopting different data augmentation and featuring engineering methods. Finally, a best-performing model amongst them was shortlisted for use in a live video demonstration.

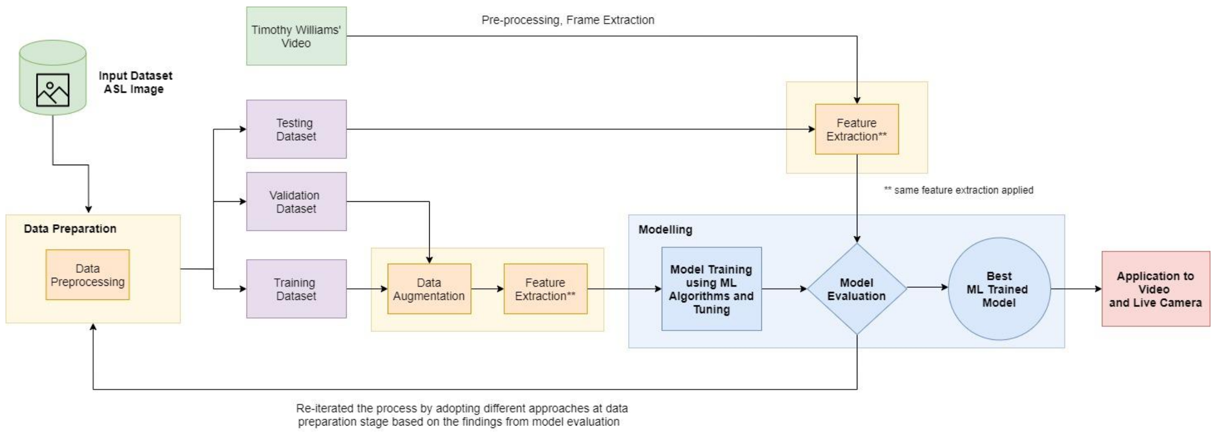

Timothy Williams' Video
Pre-processing, Frame Extraction
Input Dataset ASL Image
Testing Dataset
Feature Extraction**
Validation Dataset
** same feature extraction applied
Data Preparation
Data Preprocessing
Training Dataset
Data Augmentation
Feature Extraction**
Modelling
Model Training using ML Algorithms and Tuning
Model Evaluation
Best ML Trained Model
Application to Video and Live Camera
Re-iterated the process by adopting different approaches at data preparation stage based on the findings from model evaluation

Figure 3: Overall Architecture of this Project

5 Data Pre-processing and Approaches

The project was approached in three phases. Based on the analysis on model performance from the previous approach, data engineering and augmentation was considered subsequently to improve model performance.

5.1 Approach 1 – Full Image

Approach 1 was – in essence – a preliminary round of basic ML modelling to give a sense of potential challenges and improvement required. In this approach, both the coloured dataset and its grayscale form were directly used in model training.

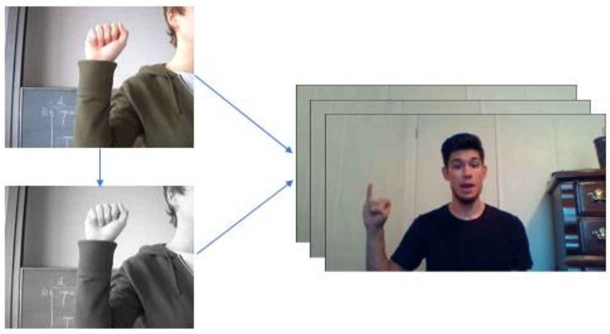


Figure 4: Approach 1 – Full Images and Conversion to Greyscale

Even though these trained models demonstrated high accuracy against test data, all of the them were unable to correctly detect letters when applied to Timothy William's video clip. The poor performance on video application led us to conduct further exploration on the possible causes.

Approach 1 Findings

Background noise: Background features such as arm and partial facial features may have been captured during training. When fitting to a data that has very different background setting, the noise could have impacted the result badly.

Greyscale causes loss of information: converting into greyscale causes some of the images to have less distinctive differentiation between background and outline of the hand as skin tone is an important element to identify hand gesture.

Scaling: we have also noticed that size of the hands in training dataset and video can be quite different. Hence, we suspect that size of hand plays a part in test result as well.

5.2 Approach 2 – Cropped Images and Hand Classifier

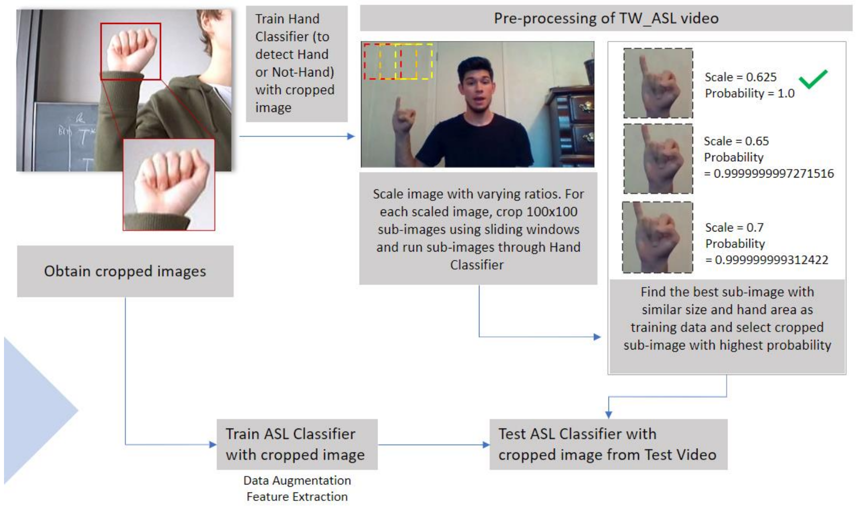

Train Hand Classifier (to detect Hand or Not-Hand) with cropped image
Pre-processing of TW_ASL video
Scale = 0.625
Probability = 1.0
Scale = 0.65
Probability
= 0.9999999997271516
Scale = 0.7
Probability
= 0.9999999999312422
Scale image with varying ratios. For each scaled image, crop 100x100 sub-images using sliding windows and run sub-images through Hand Classifier
Obtain cropped images
Find the best sub-image with similar size and hand area as training data and select cropped sub-image with highest probability
Train ASL Classifier with cropped image
Data Augmentation
Feature Extraction
Test ASL Classifier with cropped image from Test Video

Figure 5: Approach 2 – Crop Images using Sliding Window, and Hand Classifier

With the findings from Approach 1, for Approach 2 we trained models with cropped images of just the hand area to help the model focus on the details of the hand without background distraction. To be able to apply the models on real-life frames from the video, we need to pre-process these images to achieve a cropped frame of the hand.

To do so, we used the cropped training images to train 2 models, first a simple binary Hand Classifier to predict whether an image contains a hand or not a hand. For each video frame, we scaled the image to different sizes and for each scaled image, we used the Sliding Windows technique to get 100x100 pixel sub-images. We fed all these into the Hand Classifier and collected those sub-images with the highest probability of a hand.

Separately, we trained variations of ASL Alphabets Classifier with the same cropped training images, some with Augmentation and Feature Extraction. Only simple Augmentation was applied (e.g. rotation_range=10, brightness_range=[0.8, 1.2], zoom_range=0.2) so as not to distort the cropped training images, and also ASL Alphabet hand gestures are normally signed with an upright wrist with limited range of rotation.

Approach 2 Findings

With this approach, we see a great improvement in both validation and test accuracy, and the best models managed to predict 10 out of 13 alphabets from our test video. However, the models were unable to differentiate very similar alphabets like 'A', 'M' and 'S', these gestures have a common clenched fist and just slight differences in the position of the thumb.

Also, using the Sliding Windows cropping method is very computationally inefficient, for a typical 1280x720 pixel image, thousands of 100x100 pixel sub-images would be derived per scaled image, resulting in a heavy load of calculations for the Hand Classifier to predict the probability for each of these sub-images. Hence, we have concluded that a hand detection model will be needed to identify the position of the hand in the image and crop it out for model training. A hand detection model will perform much faster than the hand classifier method that was used in this approach.

5.3 Approach 3 – Hand Annotation with MediaPipe

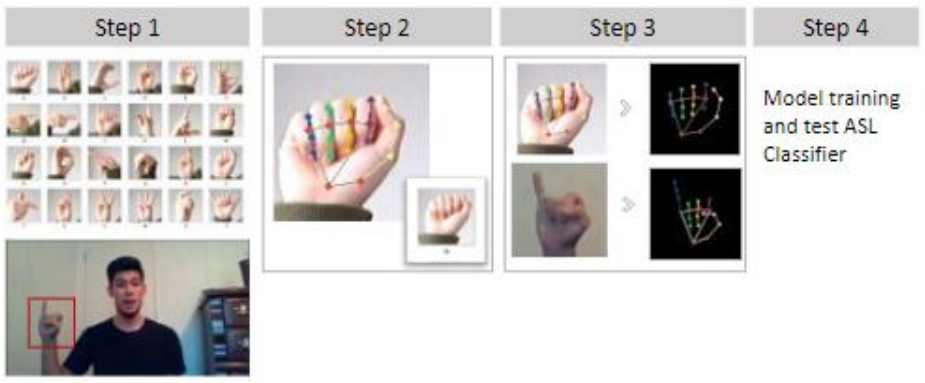


Figure 6: Approach 3 – Hand Annotation with MediaPipe as Feature Engineering

For Approach 3, we used MediaPipe Hands library (mediapipe, n.d.), a hand detection model, to detect the hand in the images. The library adds annotations for different parts of the hand. The different fingers and the palm are annotated with different colours and the dot (landmark) sizes have been adjusted with respect to the size of the hand. The annotations will help our training models to identify the positions of each finger, especially for a clenched fist.

Using the coordinates of the annotations of the hand, we can crop out the hand from the image by taking the minimum and maximum value of the height and width of the hand and then added some padding around the image of the hand. We have also created an image dataset where only the hand annotations were extracted on a black background, thus removing all background noises from the image.

We found that models trained with both versions of annotated images achieved similar results. Hence, we proceeded with just annotations with black background, since black pixels are represented with zeros which we believe would be more computationally efficient for the models and helps to remove any remaining background noises from the original images.

Approach 3 Findings

The models trained with this approach managed to differentiate very similar alphabets like ‘A’, ‘M’ and ‘S’ very well, which previously could not be done by models in Approach 2.

6 Model Development

6.1 Model and Machine Learning Algorithms

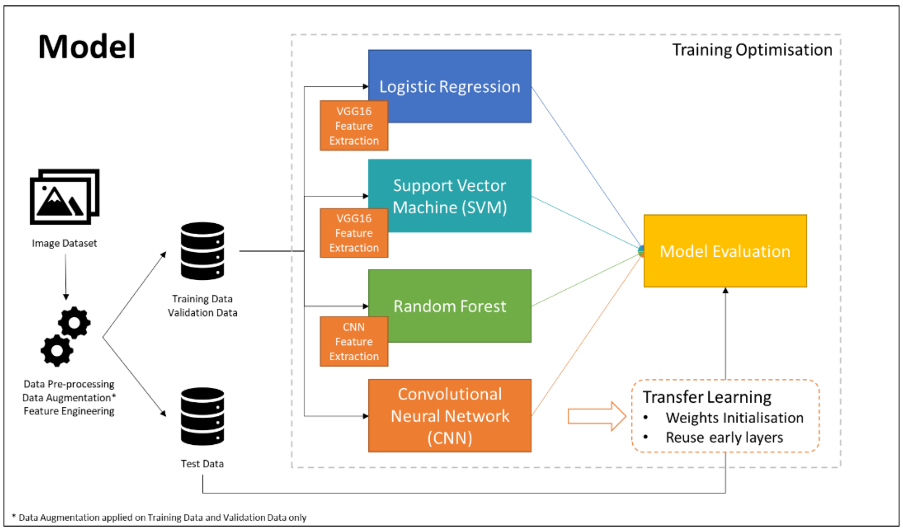


Figure 7: Model Overview

6.1.1 Random Forest

Random Forest Classifier is an ensemble model which uses the Bagging method to train Decision Trees randomly and independently in parallel, then make the final prediction on majority vote. RandomizedSearchCV varying the number of Decision Trees from 50 to 200, with and without Bootstrap was used to build a Random Forest with cross-validation.

Pure Random Forest trained on image arrays (Approach 1 and Approach 2) achieved above 80% accuracy but did not perform well on test video, most probably unable to generalise on very different conditions. Hence, we used the earlier layers of CNN models (convolution, pooling and flatten layers) which is trained using our training dataset as a Feature Extractor as an input to train another variation of Random Forest. The CNN Feature Extractor helped the Random Forest model to increase Accuracy and F1-Score by more than 10%, slightly higher than pure CNN models, but is unable to outperform the pure CNN models for prediction of test video images.

Pure Random Forest trained on just annotations with black background (Approach 3) could achieve results on test video similar to CNN models, because annotations act as a form of Feature Extraction and black background removes noise.

6.1.2 Convolutional Neural Networks (CNN)

The convolutional neural network (CNN) is a specialised form of neural network designed for working with two-dimensional image data. The key aspect of the CNN is the use of convolution layers to detect a specific type of feature in an input.

Pooling layers are used to perform non-linear down-sampling on its inputs, to make the representation approximately invariant to small translations of the input. This is useful when we are concerned about the presence of a feature rather than its position.

The output of consecutive convolution and pooling layers are then passed into a fully connected neural network to learn features from all combinations of the previous layer. Being a machine learning model specifically designed for the purpose of working with image data, we expect the CNN models to perform well across all approaches. The final chosen model (CNN #2) differed from our initial CNN models in that a dropout layer was included to reduce overfitting. This addition likely contributed to the model's ability to generalise and perform better on data that are more different from the training data, such as the test video images.

6.1.3 Logistics Regression and Support Vector Machine

Logistics Regression (LR) is a classification model based on the sigmoid function to predict the most likely class, while Support Vector Machine (SVM) is a supervised learning model that tunes a decision boundary (vector) that best separates the classes. Both models were accompanied by VGG16 Feature Extraction – a visual CNN architecture with 16 layers of convolution, max-pooling and softmax activation functions, developed by Simonyan & Zisserman (2015). LR with VGG16 was used in both Approach 2 and 3, and SVM with VGG16 was considered in Approach 3.

In Approach 2, while LR with VGG16 produced a good accuracy on test data (99%), the model performed poorly on Williams' application video clip, only correctly identifying one gesture. Like other models, this was likely because LR is unable to generalize on different conditions. In Approach 3, a SVM model was also trained to compare with LR, to determine if other base models with VGG16 feature extraction would perform better. Both models gave high accuracy on the split test set, but they were only able to pick up 8 out of 13 gestures from the Timothy test set, which further validates that annotations proved very helpful.

6.1.4 Transfer Learning Models (InceptionV3 and VGG16)

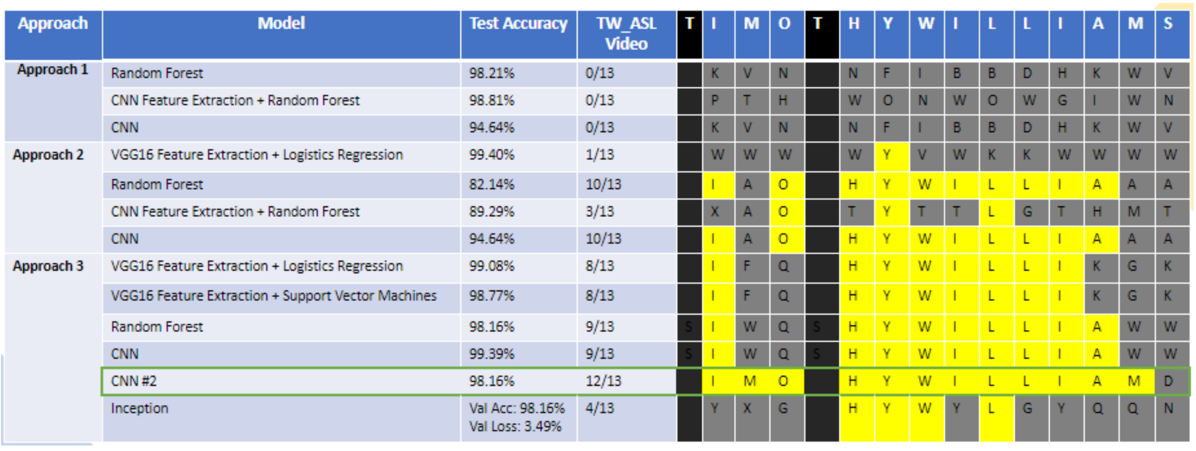
We have used two pre-trained models, InceptionV3 and VGG16. VGG16 was dropped halfway as the training time required was too long for approach 2 and 3. As the dataset from Approach 3 is smaller, the Inception model was only trained using annotated images. We have set the include top parameter to False, thereby excluding the classifier at the top, added a flatten, and two dense layers with the last one using a softmax activation function. Early stopping was also used to train the model. The validation accuracy was high at about 98% but it did not perform well with the test images from the test video. As our CNN #2 model was able to give a high test accuracy on the images from the test video, we did not try to further improve the model by considering freezing more layers in the model.

6.2 Model Performance and Analysis

The Test Accuracy and alphabet pick-up rate of the abovementioned models from Williams' video are summarized in the table below. It was observed that Williams' had signed the letter "T" incorrectly, hence they were removed from consideration and blanked out.

From Approach 1 to 3, the models slowly performed better at detecting letters from the video clip, thus validating the effectiveness of the various approaches used to enhance feature detection.

CNN #2 model performed the best across the models in Approach 3 in terms of its ability to predict ASL Alphabets on real-life test video and its model simplicity, hence was used in our demo application to show the prediction of ASL Alphabets through a webcam.



Approach	Model	Test Accuracy	TW_ASL Video	T	I	M	O	T	H	Y	W	I	L	L	I	A	M	S
Approach 1	Random Forest	98.21%	0/13		K	V	N		N	F	I	B	B	D	H	K	W	V
	CNN Feature Extraction + Random Forest	98.81%	0/13		P	T	H		W	O	N	W	O	W	G	I	W	N
	CNN	94.64%	0/13		K	V	N		N	F	I	B	B	D	H	K	W	V
Approach 2	VGG16 Feature Extraction + Logistics Regression	99.40%	1/13		W	W	W		W	Y	V	W	K	K	W	W	W	W
	Random Forest	82.14%	10/13		I	A	O		H	Y	W	I	L	L	I	A	A	A
	CNN Feature Extraction + Random Forest	89.29%	3/13		X	A	O		T	Y	T	T	L	G	T	H	M	T
	CNN	94.64%	10/13		I	A	O		H	Y	W	I	L	L	I	A	A	A
Approach 3	VGG16 Feature Extraction + Logistics Regression	99.08%	8/13		I	F	Q		H	Y	W	I	L	L	I	K	G	K
	VGG16 Feature Extraction + Support Vector Machines	98.77%	8/13		I	F	Q		H	Y	W	I	L	L	I	K	G	K
	Random Forest	98.16%	9/13	S	I	W	Q	S	H	Y	W	I	L	L	I	A	W	W
	CNN	99.39%	9/13	S	I	W	Q	S	H	Y	W	I	L	L	I	A	W	W
	CNN #2	98.16%	12/13		I	M	O		H	Y	W	I	L	L	I	A	M	D
	Inception	Val Acc: 98.16% Val Loss: 3.49%	4/13		Y	X	G		H	Y	W	Y	L	G	Y	Q	Q	N

Figure 8: Summary of model performance: CNN #2 accurately picks up 12 of 13 letters in Timothy William's video clip.

7 Application using Webcam

We have done a simple application using the webcam to predict the ASL alphabet that is being signed. The images from the webcam were constantly processed using MediaPipe Palm Annotation library to annotate the hand and transform the images required to fit into our model for prediction. To highlight the hand and show the predicted alphabet, a bounding box was drawn around the hand in the video and the result of the prediction is shown on the left top corner of the bounding box. The webcam video application is just an example of how our model could be used for people to understand sign language as a person is signing in front of a camera. Appendix A showed the prediction result of a live video we have done.

8 Conclusion and Future Works

This project outlines the learning path – with three iterative approaches – in developing an applicable model to accurately detect ASL gestures in a webcam. While earlier trained models could already achieve high levels of accuracy in train-test splitting, these models were not practical in accurately detecting gestures in a video. With data augmentation, feature engineering and hand classifiers across the approaches, model accuracies gradually improved, and the best performing model detected most of the letters in the Test Video clip. This model was successfully demonstrated using the webcam to sign the full ASL alphabet of 24 letters.

To further improve robustness, these models could be further trained against left-handed gestures (the dataset in this scope only considered right-handed gestures). Furthermore, to reduce overfitting, additional images signed by more people should be added to supplement the dataset.

Lastly, the future works for our project could include alphabets and digits from other sign languages like Brazilian or Japanese to make sign language recognition more robust. Furthermore, moving on to processing videos for prediction of words will help greatly in predicting sign language words as words require motion of the hands.

9 References

Chua, H. (2019). Healthcare access for the deaf in singapore: Overcoming communication barriers. *Asian Bioethics Review*, *11*(4), 377–390. https://doi.org/10.1007/s41649-019-00104-3

Kozik, K. (2020, October 28). *Without sign language, deaf people are not equal*. Human Rights Watch. https://www.hrw.org/news/2019/09/23/without-sign-language-deaf-people-are-not-equal.

MediaPipe Hands. mediapipe. (n.d.). https://google.github.io/mediapipe/solutions/hands.html.

Muvezwa, K. (2019, March 2). *Significant (ASL) sign language alphabet dataset*. Kaggle. https://www.kaggle.com/kuzivakwashe/significant-asl-sign-language-alphabet-dataset/version/1.

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR, abs/1409.1556*.

Williams, T. (2017). *Timothy Williams Asl Homework #1*. *YouTube*. https://www.youtube.com/watch?v=gdvxcYZi0KA.

10 Appendices

Appendix A – Demonstration of CNN#2 Model recognising different ASL Alphabets with OpenCV