Machine Learning for Sign Language Transcription

Dylan Leddy
Boston College
140 Commonwealth Ave,
Chestnut Hill, MA
leddyd@bc.edu

Sammy Lee
Boston College
140 Commonwealth Ave,
Chestnut Hill, MA
leedzc@bc.edu

1. Introduction

Over one million Americans experience hearing difficulties and rely on American Sign Language (ASL) to communicate with others. However, most people don't know ASL, which has led to a significant language barrier in our society. Notably, This barrier translates over to the digital space, in which deaf content creators have difficulty connecting with their viewers. To alleviate this problem, we aimed to construct an ASL-to-English subtitle transcriber using a computer vision based machine learning approach.

2. Related Work

ASL-to-English transcription is no novel endeavor. There are a handful of related works on the internet which utilize hand segmentation, gesture recognition, and even output refinement through spellchecking libraries. While many creations are capable of identifying ASL fingerspelling with adequate accuracy, it is difficult to provide a correct transcription for sign language "in the wild" (at natural speed and with imperfections). Our goal was to build upon the results of these works through various means.

3. Method

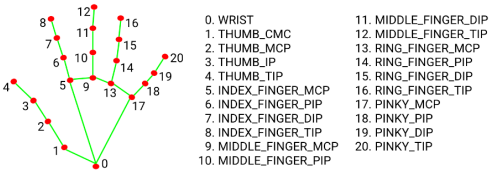
Our methodology is divided into four parts: video capturing, hand detection, classification, and output refinement. The process as a whole was focused on image classification (gesture recognition) using "hand landmarks" (key points on the hand such as joints or finger tips). Previous methods of classification involved edge detection - identifying edges and curves within the image with a two-dimensional convolutional neural network (2D CNN). However, this approach proved problematic with signs involving overlapping or hidden fingers.

3.1 Video Capturing

The program uses the Python library, Open CV, in conjunction with the laptop camera to capture a live video feed of the user. Each frame passess through the hand detection process and is eventually fed into the model for prediction.

3.2 Hand Detection

To detect the hand(s) of the user, we used the open-source framework, Google MediaPipe. While MediaPipe is capable of locating many different hands, this project is limited to ASL fingerspelling, which only requires identifying one. Once located, the image patch containing the hand is first inverted into the RGB colorspace in order to be processed by the landmark detection function in mediapipe. An array with a total of twenty keypoints is returned for classification.



3.3 Classification (Architecture)

The model building process uses Keras to construct each of the layers for our CNN. The pipeline of the network takes in the landmark

array as input and drops 20% of the units in the layer during training to prevent the model from overfitting. The resulting units are passed into a fully connected layer with a rectified linear unit (ReLU) activation function to prevent vanishing gradients. This process is repeated until the units are condensed to 28 categories and finalized with the softmax function to get a probability for each category.

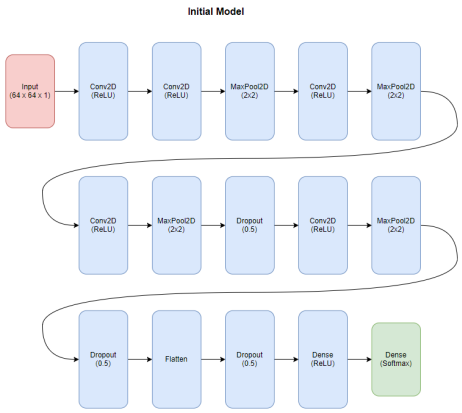


Figure 3.3.1 *A diagram of the initial CNN*

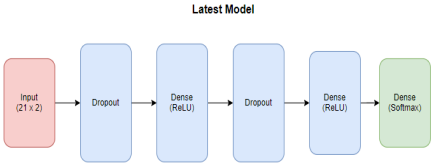


Figure 3.3.2 *A diagram of the present model*

3.4 Output Refinement

During runtime, there will inevitably be machine and human error. To dampen the impact of classification inaccuracy, the program refines the output through spell checking using Pyspellchecker, a free Python library that corrects words based on Levenshtein Distance[1]. Refinement allows for a more fluid user experience because it reduces the need for manual deletion and correction.

4. Experiments

We employed a custom dataset, "SAMMYInTheDorm," to train our model, comprising 28 distinct folders. These folders encompass hand signs representing the alphabet, along with symbols for delete and space. Each folder is meticulously labeled with the corresponding letter and contains 600 photographs. These images exhibit variations in lighting and background movement, ensuring the dataset's diversity and enhancing our model's training process.

Our pipeline processes each image through the Mediapipe API to detect and extract the landmarks for each hand. In the training phase, our model utilizes a learning rate of 0.001 and a batch size of 128. Leveraging the Adam optimizer eliminates the need to specify a momentum parameter. The training process typically converges after approximately 100 epochs, with a callback function in place to halt training early if loss values cease to decrease after 15 epochs.

We employ sparse categorical cross-entropy as our loss function, given that each class is represented by integer labels (ranging from 0 to 28) for the alphabet letters, delete, and space, instead of one-hot encoding. Notably, our model does not employ 2D Convolutional Neural Networks (CNNs) on images. As a result, training the model using an integer array of landmarks takes a mere 3 minutes, a stark contrast to the initial base model, which required a daunting 7 hours to complete all 5 epochs.

In a head-to-head comparison with the traditional 2D CNN classification approach, our current model exhibits superior accuracy, particularly in classifying challenging letters like M, N, and T, which involve intricate finger poses that the baseline method often struggles to accurately identify.

Model 1 (Baseline)

Input	M	N	T	E	V
Output probability	T 0.82	F 0.57	O 0.86	E 0.54	D 0.94

Model 2 (Current Model)

Input	M	N	T	E	V

[1] A similarity measurement based on the number of insertions and deletions needed to make two words equal.

Output probability	M 0.75	N 0.65	T 0.56	E 0.72	V 0.67

5. Conclusion

In our project, we created an innovative ASL-to-English subtitle transcriber to tackle the language barrier experienced by over one million Americans with hearing impairments. Employing a computer vision-based machine learning framework, our solution seamlessly integrated video capturing, hand detection, classification, and output refinement processes. A distinctive feature of our approach was its emphasis on image classification using "hand landmarks," a departure from conventional 2D CNN methods. This strategic shift yielded a product that exhibited substantial enhancements over our initial approach, notably excelling in the accurate classification of complex ASL signs. As a result, our project has played a valuable role in exploring ways of advancing digital communication for the deaf and hard of hearing community.

5.1 Future Work

- Expanding capabilities to whole words rather than just letters.
- Introducing optical flow to handle letters/words dependent on movement
- Use of a contextual spell checking tool, such as Bidirectional Encoder Representations from Transformers (BERT) for better output refinement.
- Superior dataset (larger, more diverse signers/lighting, etc) and more powerful resources to train with it.

6. References

[1] Ilango, G. (n.d.). Recognizing "hand gestures" using opencv and python. GitHub. Retrieved October 18, 2022, from https://github.com/Gogul09/gesture-recognition

[2] Fowley, F., & Ventresque, A. (2021). Sign Language Fingerspelling Recognition using Synthetic Data. Retrieved October 18, 2022, from http://ceur-ws.org/Vol3105/paper23.pdf. [3] Lee, D. (n.d.). Interface with Roboflow in python. GitHub. Retrieved October 18, 2022, from https://github.com/roboflow-ai/roboflow-python

[4] Tay, Y. H. (n.d.). Computer Vision based hand gesture recognition using artificial neural ... Retrieved October 19, 2022, from https://www.researchgate.net/profile/YongHaurTay/publication/240825724_Computer_Vision_Based_Hand_Gesture_Recognition_Using_Artificial_Neural_Network/links/0deec532525995be39000000/Computer-VisionBased-Hand-Gesture-Recognition-Using-Artificial-NeuralNetwork.pdf