ASL Fingerspelling Prediction

Anant Vatta
ASU ID:1219323846
Arizona State University
avatta@asu.edu

Austin Gilmore
ASU ID: 1216396051
Arizona State University
acgilmor@asu.edu

Manoj Mysore Srinath
ASU ID: 1222220922
Arizona State University
mmysores@asu.edu

Neeraj Sreedhar
ASU ID: 1222289653
Arizona State University
nsreedh3@asu.edu

***Abstract*—A python/android application that predicts the ASL signed word from an input video. This application first performs a segmentation algorithm to divide the input video into frames containing individual letters. These frames are then cropped using the MediaPipe python library to focus only on the hand. These cropped images are finally passed into a trained CNN model to predict the signed letter. The concatenation of these letters form the predicted word from the given input video.**

I. Keywords

Sign Language, Segmentation, Convolutional Neural Network

II. Introduction

According to a survey created in 2019, there are around 430 million people suffering from hearing disorders. And this number is projected to increase to around 711 million by 2050. Hearing loss is a far more common problem than people think. Conductional hearing loss, sensorineural hearing loss, and mixed (conductive and sensorineural) hearing loss are the three main types of hearing loss. Among all types of hearing loss, sensorineural loss is the most common. Some common causes of hearing loss in adults are exposure to loud noise, Acoustic neuroma, Physical head injury, Presbycusis.
American Sign Language (ASL) is one of the primary sign languages that are used by people with hearing disabilities. One can use hands, face expressions, or body movements to effectively communicate by means of this visual and gestural language.
A Convolutional Neural Network is the most popular type of Deep Learning algorithm for visual images. The CNN uses a multitude of weighted and biased factors that can be learned, uses them to classify images and then uses these to differentiate them. CNNs are inspired by Neurons in the brain and are actually nothing more than a fully connected set of neurons in one layer to another. We are using a CNN model that is trained on American Sign Language images.
In this project we are building an application that allows the user to record their gesture video on their mobile device and upload it to the server where it is processed and interpreted for the sign enacted in the video. The input word video is pre-processed and then given to a trained Convolutional Neural Network for prediction. The individual alphabets predicted by the CNN are then combined to form a word.

III. Collecting Data

In order to validate our finger-spelled word model, we first generated alphabet data, and then we generated our testing data for the entire finger-spelled word. For generating the alphabet and word videos, we developed an Android mobile application that captures user gesture video and upload it to a server. To begin, we captured 5 second alphabet videos for each letter using the mobile application we developed. Then we randomly selected 40 words with 3 to 5 letters in them and captured the video of us finger-spelling the words with 3 seconds per alphabet. This 3 second length is essential for our segmentation algorithm that will be explained in the following section.

IV. Implementation

To detect the palm in the image, the video, which contains individual alphabets, must be divided into frames. We then take the middle frame from the extracted frames and this is used as a representative of the ASL alphabet in the video (Figure 2). The OpenCV library was imported into the python script and then selected the middle frame by dividing the video length by 2.

Once the middle frame is extracted, it is passed into the mediapipe library function to derive the annotated image (Figure 3). The mediapipe library provides cross-platform machine learning solutions for live and streaming media. Some of the solutions it provides are: Face detection, iris detection, hands recognition, pose detection, face mesh and many more. For our purposes, we used the hand detection algorithm solution in the mediapipe library to extract the hands from our derived middle frame.

Once the annotated frame is obtained, the image is then cropped in this fashion. An imaginary rectangle box is drawn around the annotated image which completely covers the whole hands area. All of these points are obtained from the mediapipe hands function and using the obtained co-ordinates and the OpenCV cropping, we get the cropped image. The cropped image is then written (saved) into a .png file. The mediapipe library is also capable of detecting multiple hands as well. Mediapipe can also detect the orientation of the hand (left or right). Since the CNN model will be trained on images with right hands, we can flip the cropped image if mediapipe detects the orientation of hand in the given image as left.

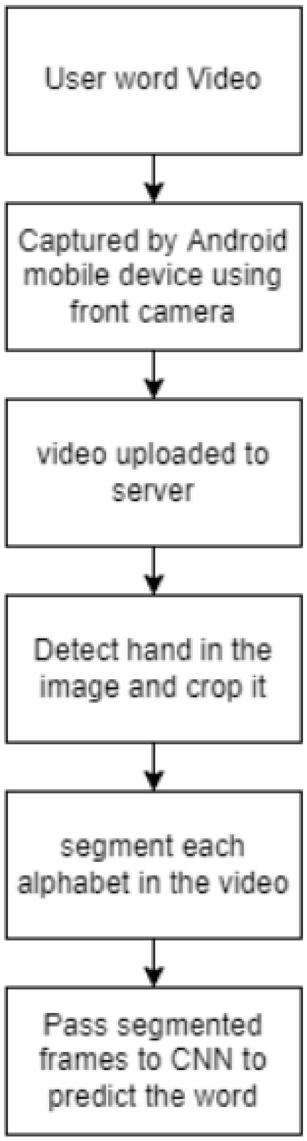
User word Video
Captured by Android mobile device using front camera
video uploaded to server
Detect hand in the image and crop it
segment each alphabet in the video
Pass segmented frames to CNN to predict the word

Fig. 1. ASL Prediction workflow.

Fig. 2. Middle Frame from Alphabet Video.

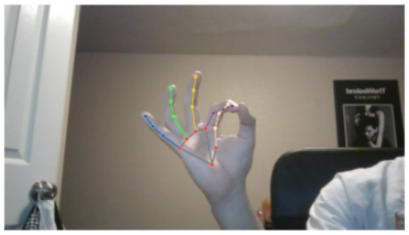

Fig. 3. Annotated Image from Alphabet Video.

Fig. 4. Cropped Image from Alphabet Video.

V. Alphabet Detection Using CNN

The CNN used for our purpose of detecting the ASL alphabet was a ResNet CNN which was pre-trained. Since the CNN was trained to recognize the wide variety of objects, we need to only train the last layer of the CNN using the dataset obtained from kaggle here. The dataset only consists of right hand images, therefore the model is only trained to recognize the right handed ASL alphabets.. This fashion of training a CNN is called transfer learning. The training was done using Adam optimizer and categorical cross entropy loss, and the model converged within 15 epochs. The Adam optimizer is highly compatible for image recognition problems and since we need to extract the prediction over multiple classes, we need to use the cross entropy loss function.

The trained model was then loaded into the python script and the cropped image was passed into the CNN model. Before passing the image in the CNN model, the cropped image needs to be pre-processed, by resizing it to the size of the input layer of the CNN and also converting into grayscale image. After preprocessing the image, it is passed into the CNN model and the feature vector is extracted from it.

VI. CNN Accuracy Analysis

Once we received a prediction for each of our alphabet videos using the trained CNN model, we then developed a python script to calculate the F1 score metrics for each classification[4]. To determine each F1 score, the precision of each classification was first calculated using the following formula (using class ‘a’ as an example):

$Precision(class =' a' = \frac{TP(class='a)}{TP(class='a')+FP(class='a')})$

In this formula, TP represents True Positives, or the number of test cases (alphabet videos) in which the predicted classification (signed letter) matches the actual classification (class ‘a’), and FP represents False Positives, or the number of test cases in which the predicted classification was ‘a’ but the actual classification was any other class (‘b’-’space’). The recall of each classification was then calculated using the following formula (using class ‘a’ as an example):

$Recall(class =' a') = \frac{TP(class='a')}{TP(class='a')+FN(class='a')}$

In this formula, FN represents False Negatives, or the number of test cases (alphabet videos) in which the actual classification was ‘a’, but the predicted classification was any other class (‘b’-’space’). Using these values, the F-1 Score for class ‘a’ can be calculated by the following formula:

$F1 - Score(class =' a') = 2 \times \frac{Precision(class='a')}{TP(class='a')+FN(class='a')}$

After using the CNN predictions for all 108 of our test cases, the following F1 scores were calculated for each class:

Class	F1-Score
a	0.4
b	0
c	0
d	0
e	0
f	0.087
g	0
h	0
i	0
j	0
k	0
l	0.142
m	0
n	0
o	0
p	0
q	0
r	0
s	0
t	0
u	0
v	0
w	0.2
x	0
y	0
z	0

These F1-scores provide insight in determining the accuracy of our model. The low F1-Scores for many of our classifications can be attributed to a variety of factors. For one, each of us conducted our alphabet video in different environments. This introduced noise in the background of our videos that may have led to inaccurate predictions. Each of our group members also used different hands (right vs. left) when signing the letters in the videos. Since the data set used to train the CNN model only included right hands, we should have restricted our test videos to only right hands or flipped all test videos conducted with left hands. This would decrease the variability in hand orientation and, ultimately, increase the F1 scores for each class.

VII. Word Segmentation

In order to use our CNN model to predict a signed ASL word from a given video, we first had to develop a segmentation algorithm to divide the video into individual letters. Our algorithm assumed that input videos are recorded at 30 frames per second and that each signed letter lasts for 3 seconds. To develop it, we first used the OpenCV library in Python to determine the total number of frames in each word video. Then, using our previous assumptions, we segmented the video into 90 frame sections (3 seconds x 30 frames per seconds), each section representing an individual letter. We then extracted the middle frame of this section to minimize any unwanted noise resulting from transitioning between letters. Finally, we passed each extracted frame into our hand extraction algorithm to produce a cropped hand image for each letter in the inputted word video.

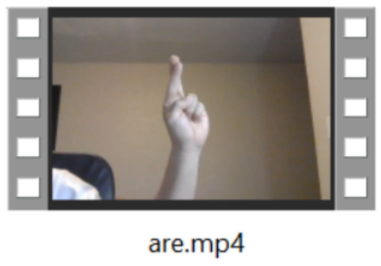

Fig. 5. Word Video File.

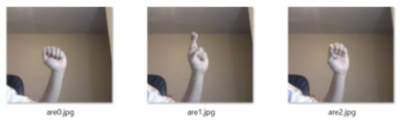

Fig. 6. Cropped Hand Images.

VIII. Word Prediction Using CNN

After getting the middle frames for every letter from the video we preprocess those images to reshape them to a size of 200×200. We also turn these images into grayscale because of the input layer of the CNN requires the image to be of the size $200 \times 200 \times 1$ and flip them for a better prediction. We then use these images with our CNN model for predicting the words being demonstrated in the video. This model returns an alphabet that is like the one in image.

These resultant alphabets are joined to get a word which the model predicts in the video. The actual words which we extract from the video filename are used for comparison with the predicted word to check if the model has made the correct prediction. The accuracy is calculated based on alphabets and not for the complete words.

IX. Conclusion

In this project we can segment the video and extract exact frames depicting an alphabet from the American Sign Language. These extracted frames were used for word prediction. Using the CNN Model, we were able to use these extracted frames to correctly predict 10 out of 129 total letters in our word videos (7.8%).

This model can be very useful for better communication between people using the American Sign Language. Even people who don't understand sign language will be able to better understand what someone is trying to tell them.

References

[1] Akash. 2018. Image data set for alphabets in the American Sign Language. Kaggle. (2018). https://www.kaggle.com/grassknoted/asl-alphabet.

[2] Valentin Bazarevsky and Fan Zhang. 2019. On-Device, Real-Time Hand Tracking with MediaPipe. Blog. (19 July 2019). https://ai.googleblog.com/2019/08/ on-device-real-time-hand-tracking-with.html.

[3] Prasanth Sukhapalli. 2019. For Generating Key Points using Tensorflow's PoseNet. Github. (15 July 2019). https: //github.com/prashanthnetizen/posenet$_nodejs_setup$.

[4] Baeldung. 2020. F-1 Score for Multi-Class Classification. (19 October 2020). https://www.baeldung.com/cs/multi-class-f1-score