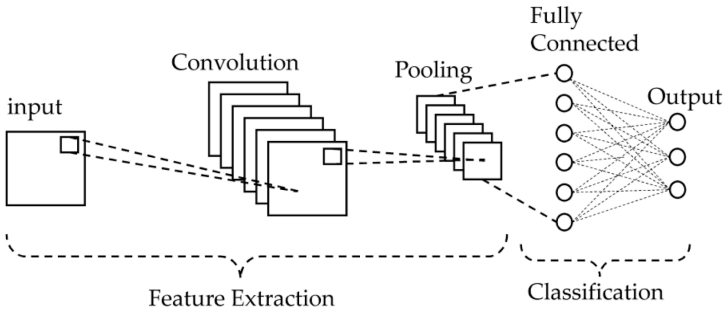
# A Voice For The Voiceless

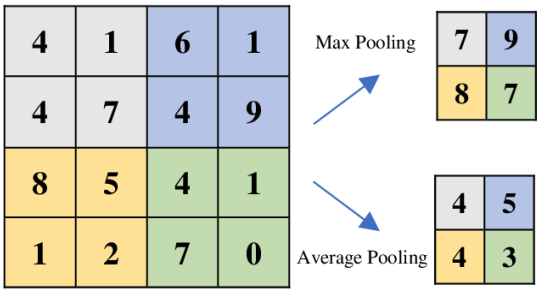
By Vedant Sheel

## Introduction

American Sign Language (ASL) is a natural and ancient form of human communication. Computer vision research has taken an interest in recognizing human gestures from camera images. A real-time translating Python program that can translate ASL characters into computer-generated speech in real-time was created, using Convolutional Neural Networks (CNN).

A neural network is a computer program that learns from data, similar to how the human brain learns. It has layers of processing nodes that work together to predict/recognize patterns in new data. It is useful for image recognition. A CNN is a machine learning algorithm that is great for image classification, with multiple layers that learn to detect patterns/features in the input data. Starting with raw images, the CNN detects low-level features like edges, and gradually learns to recognize more complex shapes and textures. The output layer then produces a prediction based on patterns and features identified in the input data.




A pooling layer in a CNN down samples the output of the previous layer, using either max or average pooling. Max pooling selects the maximum value from each region. Average pooling takes the average from each region. This reduces the spatial dimensions of the feature maps, improving efficiency and performance by reducing the parameters. In this project, three pooling layers are tested for which the CNN is trained on an image dataset of hand positions and orientations, enabling real-time recognition of finger spelling in ASL. Data image-inputs are then filtered and classified based on predicted hand gestures. The system further exports the recognized alphabet as a string with dictionary suggestions with a "speak" button for voice output.

## Purpose

Effective communication is essential in our daily lives. However, for over 5% of the world's population, or the estimated 450 million human population who have hearing difficulties according to the World Health Organization (WHO), sign language is the primary means of communication. While ASL is widely used by the Deaf community worldwide, not everyone is proficient in it, resulting in a language barrier that can lead to exclusion, misunderstanding, and isolation. This project aims to address this challenge by providing a tool that can translate ASL into text and subsequently to voice in

real-time. This has the potential to transform the way people communicate with Deaf individuals, making communication easier, efficient, and more effective.

**Materials**

Python 3.10, Pycharm 22.1, OpenCV 4.2, Mediapipe 3.1, Numpy 2.3, Pyttsx3 2.6, Math 1.3, Keras 2.2.5, Tensorflow 2.12, Pyenchant 3.2.2, Tkinter 8.6, Pillow 9.4, Google's Teachable Machine, computer, keyboard, mouse, webcam, monitor, power supply, HDMI cord.

**Methodology**

Two approaches to predict a character with high accuracy were suggested:

A. To create a model using a dataset of each character with a very large repository. For instance creating a model using 2000-3000 images per character.The issue with this method is that the overall program will occupy large memory space which does not suit the future goal which is to inherent this in a mobile application.

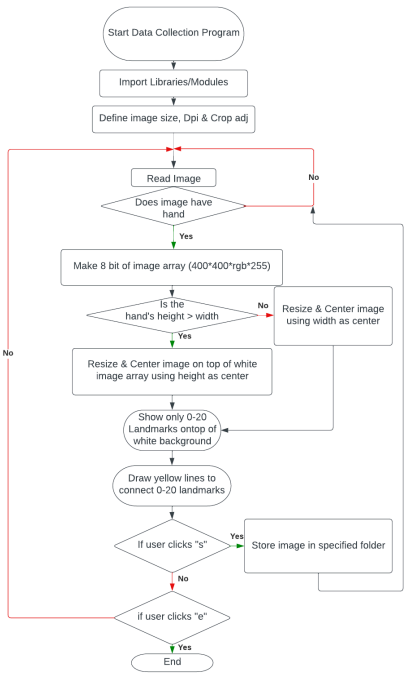
B. To use a small sample size of dataset and classify each similar looking gesture into their own classes. Then to further extract the character within that class using a novel approach wherein a distance between various landmarks will be calculated and utilized to determine the character.

Since option B uses very small memory space in contrast to option A, this was the chosen method. Two big challenges:

I. How to recognize a hand in the eyes of a computer? Humans can easily spot a hand, but how can a computer without any self-intelligence?

II. How to make computer classify what gesture the user is performing?

To address this, a five-stage methodology approach was created for an ASL character to text real-time translator using OPTION B.



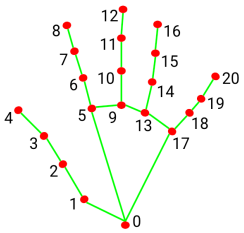

**1. Data pre-processing & image segmentation**

In this Python language project, a frame-by-frame image is captured from the webcam using the OpenCV open-source library. The hand in the image is then identified and tracked with 21 landmarks with the help of cvzone and mediapipe libraries. This addressed the first problem of detecting the hand. Once a hand is detected, the hand image is cropped and the size is calibrated so that each hand image is 400x400 pixels. To minimize the memory size and to keep only useful information needed for prediction, three separate pooling layers are created.

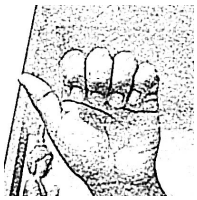
**1. The Grayscale layer:** This pooling layer converts the image to black and white, removing saturation.

**2. Gaussian blur over grayscale image.** This filter will blur all the background from the image leaving only the hand's important outlines in black and white.

**3. The landmark segmentation.** In this filter the camera input is not shown, only displaying the 21 landmarks on a white 400x400 image-array. A code is further written to connect each landmark, outlining the hand shape using a yellow coloured line, solving lighting and background confusion for the computer.



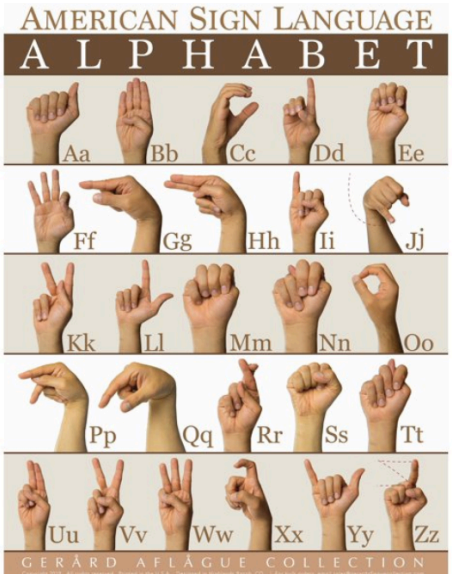

0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP



2. **Data collection:**
   In this step a dataset of images of each ASL letter was created. One hundred and eighty sample images of each letter were collected and stored in a directory for each letter. This dataset was used for the CNN training model, providing various samples and possibilities of each ASL letter gesture.

3. **Data training / modeling**
   Once the dataset is collected, the next step is to train the model. In this project, Google's Teachable Machine was used to create a keras ".h5" model. Teachable Machine is a web-based tool that allows users to train machine learning models. The preprocessed images were used as input to train and classify into different alphabet classes. In the first few trials for this project, there was one huge model holding twenty-six different characters, which was overloaded to accurately predict the image sample size (180 classes for an alphabet each). Hence, eight different groups were created (based on similar hand gestures in ASL) to model rather than twenty-six classes.

   1. [A,E,M,N,S,T] - Class with all closed hand positions (like fist).
   2. [B,D,F,K,L,R,U,V,W] - Class with all flat open hand positions.
   3. [C,O] - Class with hands curved like a circle.
   4. [G,H] - Class with hand bent (gun shape.)
   5. [I] – Class with "I" by itself.
   6. [J,Y] - Class with hands closed and pinky open.
   7. [P,Q,Z] - Class with hands bent with two open fingers.
   8. [X] – Class with "X" by itself.



## 4. Prediction

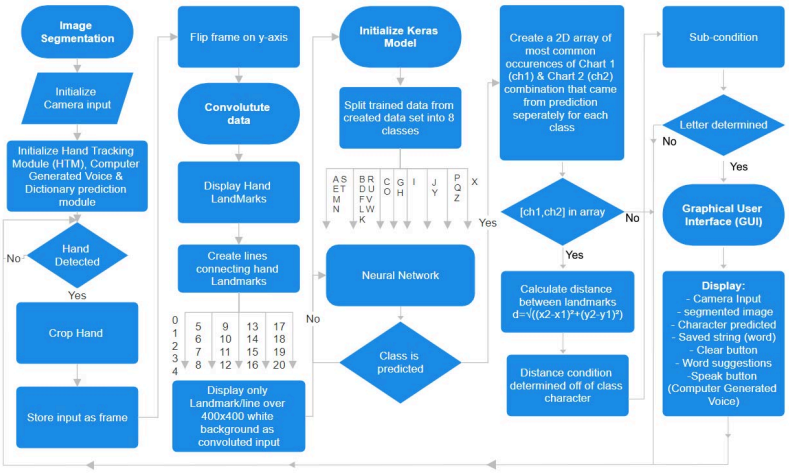
This stage is divided into a two-step process.

### A. Predicting the class

There are 8 classes which store the 26 characters. In order to predict the character one must first determine the class holding it. By calling the keras model, the train images are called upon and compared with the convoluted input which is stored as frames. Using the CNN created, 186 predicted readings for each letter were collected on a spreadsheet with two charts; Chart1 (ch1) and Chart 2 (ch2). Chart 1 held the most predicted class's index which would range from zero to seven while ch2 held the second most predicted class. Several 2 dimensional arrays were written for each of eight classes based on the most common occurrences of ch1 and ch2 combinations for each letter in their class. The CNN and ch1,ch2 combination gives out a 98.87% accurate prediction to determine the correct class.

### B. Predicting the character

After the class is determined, all that's left is to distinguish between the characters within the class itself, and present the highest confidence value as output. To further distinguish between characters, one may suggest using a CNN once again as it had such high accuracy for the 8 classes. However, when CNN is attempted, the computer provides incorrect and inaccurate outputs as it is very complicated to distinguish between a few letters in common. To solve this problem a different approach was taken. Rather then using a CNN to search within the class for the most accurate character, distance calculations were implemented.

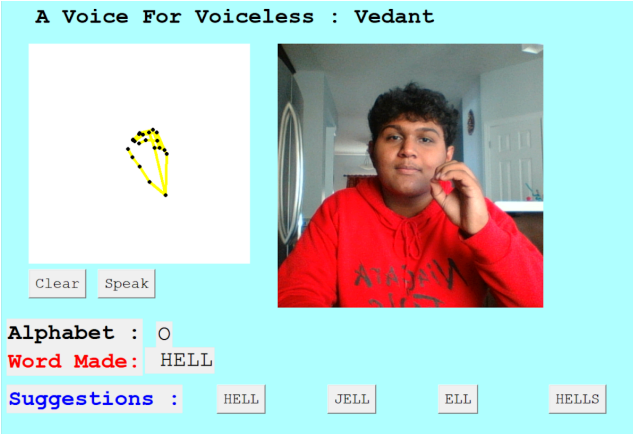



$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

To solve for distance from a point to point, Euclidean distance formula was used for which two landmark locations were calculated.

To sort within [A,E,M,N,S,T] class,

5. **GUI & Additional features**
   A GUI was created which would display the current image, skeleton image, predicted alphabet, exported string with the help of tkinter library. With the enchant and pyttsx3 libraries, dictionary suggestions, and text-to-voice were implanted. Lastly, “Clear” and “Speak” buttons.




## Observations

During the data-preprocessing and feature extraction phase, the plain hand images were first trained. The prediction/confidence values obtained were in between 62-73% and varied depending on the background and lighting, resulting in very low accuracy. To eliminate the background and improve accuracy, a gray filter was applied, which raised the prediction values to a constant 67-72% which may have been a low accuracy, but more consistent. After applying the gaussian blur filter above the grayscale pooling layer, the prediction value average increased to 77%. After further segmenting the image and creating a landmark layer and distinguishing for the character within the eight classes by conditional statements, the accuracy increased to 98%.

## Conclusion

In conclusion, a real-time ASL recognition system was successfully developed. Out of all three datasets, the binary hand-landmarks with white background were most successful with an accuracy of 97%. The use of CNN and pooling layers allowed for accurate hand gesture recognition. The project also collected a dataset of ASL character images for model training and implemented eight classes to improve prediction accuracy.

This project is a rich area for future research and development. One potential direction for improvement is to create a mobile app that runs the system on a smartphone. This will greatly increase accessibility, allowing users to communicate on-the-go. To achieve this, the system will need to be optimized for smartphones with limited processing-power and memory, while maintaining a high level of accuracy with cloud-storage. Another area for development is to incorporate ASL words. This can be achieved by collecting more training-data, developing more sophisticated algorithms and modeling for feature-extraction and classification.

**Bibliography and Resources**
https://docs.google.com/document/d/13hVrbgxSqe194aXSeVJyJg8O5A23EfzyiD8X7EmguUE/edit?usp=sharing

**References & Bibliography**
Shaw, Zed. *Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code*. Addison-Wesley, 2017.

Renotte, Nickolas, director. *Real Time Sign Language Detection with Tensorflow Object Detection and Python | Deep Learning SSD*. *YouTube*, YouTube, 5 Nov. 2020, www.youtube.com/watch?v=pDXdlXlaCco.
Renotte, Nicholas. *"Sign Language Detection using ACTION RECOGNITION with Python | LSTM Deep Learning Model."* Youtube, 19 Jun. 2021 https://www.youtube.com/watch?v=doDUihpj6ro.

Robotics and AI. , Murtaza's Workshop, director. *"Easy Hand Sign Detection | American Sign Language ASL | Computer Vision."* . *YouTube*, YouTube, 4 July 2022, www.youtube.com/watch?v=wa2ARoUUdU8.

YouTube. (2021). *Ai For Everyone Lesson 1: Introduction to Artificial Intelligence for Absolute Beginners*. *YouTube*. Retrieved April 26, 2023, from https://www.youtube.com/watch?v=gD_HWj_hvbo.

McWhorter, Paul, director. *AI for Everyone LESSON 2: Running Two Different Versions of Python on the Same Computer*. *YouTube*, YouTube, 27 July 2021, www.youtube.com/watch?v=I7GKSQeFGQs.

McWhorter, Paul, director. *AI for Everyone LESSON 3: Understanding and Using Python Virtual Environments*. *YouTube*, YouTube, 29 July 2021, www.youtube.com/watch?v=XCvsLMk4OHM.

McWhorter, Paul, director. *AI for Everyone LESSON 5: Installing OpenCV in Windows and Launching WEBCAM*. *YouTube*, YouTube, 3 Aug. 2021, www.youtube.com/watch?v=fGRe4bHRoVo.

McWhorter, Paul, director. *Managing Multiple Windows Size and Position in OpenCV*. *YouTube*, YouTube, 12 Aug. 2021, www.youtube.com/watch?v=VRRTglL9fN0.

McWhorter, Paul, director. *AI for Everyone LESSON 7: Understanding Pictures and Images as Data Arrays*. *YouTube*, YouTube, 17 Aug. 2021, www.youtube.com/watch?v=W43MpRroplA.

McWhorter, Paul, director. *AI for Everyone LESSON 9: Understanding Region of Interest (ROI) in OpenCV*. *YouTube*, YouTube, 31 Aug. 2021, www.youtube.com/watch?v=E46B7NPWK38.

McWhorter, Paul, director. *AI for Everyone LESSON 16: Face Recognition with OpenCV*. *YouTube*, YouTube, 19 Oct. 2021, www.youtube.com/watch?v=BHYZ1xkTGi8.

McWhorter, Paul, director. *AI for Everyone LESSON 18: Mediapipe for Hand Detection and Pose Estimation*. *YouTube*, YouTube, 2 Nov. 2021, www.youtube.com/watch?v=xHK-wv2JG18.

Gannon, Madeline. "Madelinegannon/Example-Mediapipe-UDP: Connecting Openframeworks to Google MediaPipe Machine Learning Framework over UDP." *GitHub*, www.github.com/madelinegannon/example-mediapipe-udp.

Grov, Ivan. "Hand-Gesture-Recognition-Mediapipe." *GitHub*, 28 Jan. 2021, www.github.com/kinivi/hand-gesture-recognition-mediapipe.

"Deaf Canadians 'at Risk' in Times of National Emergency | CBC News." *CBCnews*, CBC/Radio Canada, 27 Sept. 2018, www.cbc.ca/news/health/deaf-canadians-at-risk-in-times-of-national-emergency-1.4832321#:~:text=The%20Canadian%20Hearing%20Society%20estimates%20there%20are%203.15,deaf%2C%20including%20an%20estimated%2011%2C000%20who%20are%20deaf-blind.

Raval, Devansh. "Devansh-47/Sign-Language-to-Text-and-Speech-Conversion: This Is a Python Application Which Converts American Sign Language into Text and Speech Which Helps Dumb/Deaf People to Start Conversation with Normal People Who Dont Understand This Language." *GitHub*, 11 Nov. 2022, www.github.com/Devansh-47/Sign-Language-To-Text-and-Speech-Conversion

Nine, Neural, director. *Tkinter Beginner Course - Python GUI Development*. *YouTube*, YouTube, 29 Sept. 2021, www.youtube.com/watch?v=ibf5cx221hk.

Galli, Keith, director. *How to Program a GUI Application (with Python Tkinter)! YouTube*, YouTube, 1 Feb. 2019, www.youtube.com/watch?v=D8-snVfekto.

CodeHub, Sam, director. *How to Create Advance Text to Speech Convertor App Using Python Tkinter Framework & PYTTSX3 Library. YouTube*, YouTube, 22 Sept. 2022, www.youtube.com/watch?v=0ldMgo8r9Ck.

Dedic, Sanjin, director. *Python Enchant English Dictionary + Caesar Cipher Decryption*. *YouTube*, YouTube, 6 July 2017, www.youtube.com/watch?v=vdhCzy4Ibps.

**Bibliography Digital Images**

Guissous, Alla Eddine. "9: Example for the Max-Pooling and the Average-Pooling with a Filter ..." *ResearchGate*, Nov. 2019, www.researchgate.net/figure/Example-for-the-max-pooling-and-the-average-pooling-with-a-filter-size-of-22-and-a_fig15_337336341.

"American Sign Language (ASL) Alphabet (ABC) Poster." *Gerarda Flague Collection*, www.gerardaflaguecollection.com/products/american-sign-language-asl-alphabet-abc-poster.html

Google. (n.d.). MediaPipe Hand Landmark Model. Retrieved April 26, 2023, from https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

Saha, Sumit. "A Comprehensive Guide to Convolutional Neural Networks-the eli5 Way." *Medium*, Towards Data Science, 16 Nov. 2022, www.towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.