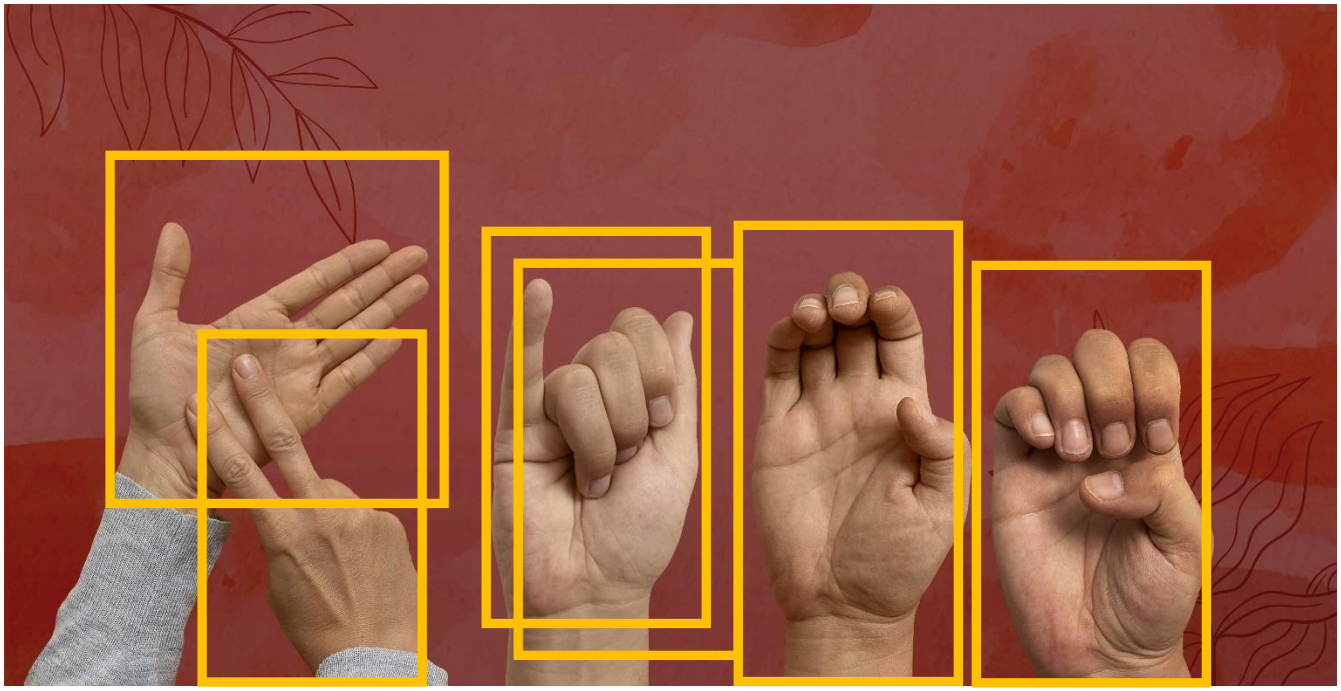
Translating Silence - Revolutionizing Communication with Neural Network-Powered Sign Language to Text

Translating Sign Language Tensorflow and Keras

Wajeeha Parker & Madiha Imran



Introduction

Language is the key to communication in our world, whether it is between humans, machines or a mix of both. In the realm of communication, sign language serves as a vital bridge for people who face challenges in expressing themselves through spoken words.

AI holds incredible potential in bridging those communication gaps. In this blog, we will walk you through the sign language detection system designed to seamlessly detect and convert hand and head gestures into texts.

But why do we need a detection system and especially when we have the interpreters around? Although sign languages are effective in communication, they are not commonly used or understood and create a communication barrier. Therefore, using an automated sign language translator that can translate sign language into written language would be a huge step in this direction. The hearing - and vocally - impaired people will benefit from an automatic sign language translator in their daily walk of life. This technology signifies a significant step toward an inclusive and interconnected society, ensuring that everyone, regardless of their abilities, can communicate freely and lead autonomous lives

The data for this project is sourced from the database of ASL (American Sign Language) sentences published by the National Center for sign language and Gesture Resources of the Boston University. It consists of 201 annotated video streams of ASL sentences. The videos are at 30 frames per second and the size of the frames is 312*242 pixels.

Reference Link: https://www-i6.informatik.rwth-aachen.de/aslr/database-rwth-boston-104.php

Now that the data and the input source are out of the way, we can discuss the more grueling part, i.e. the network architecture.

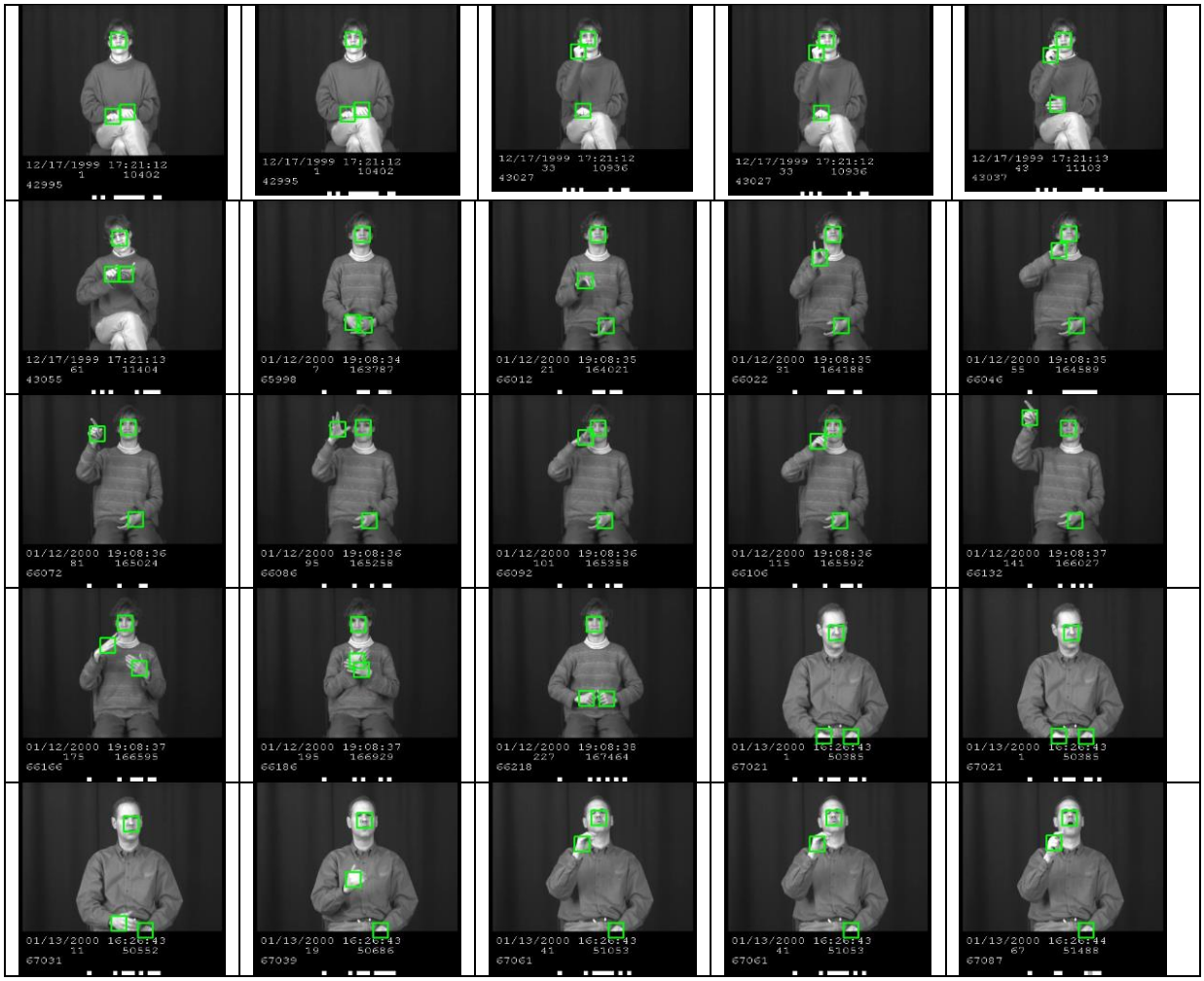


Image Reference: Cropped images of sign language from RWTH-AACHTEN database

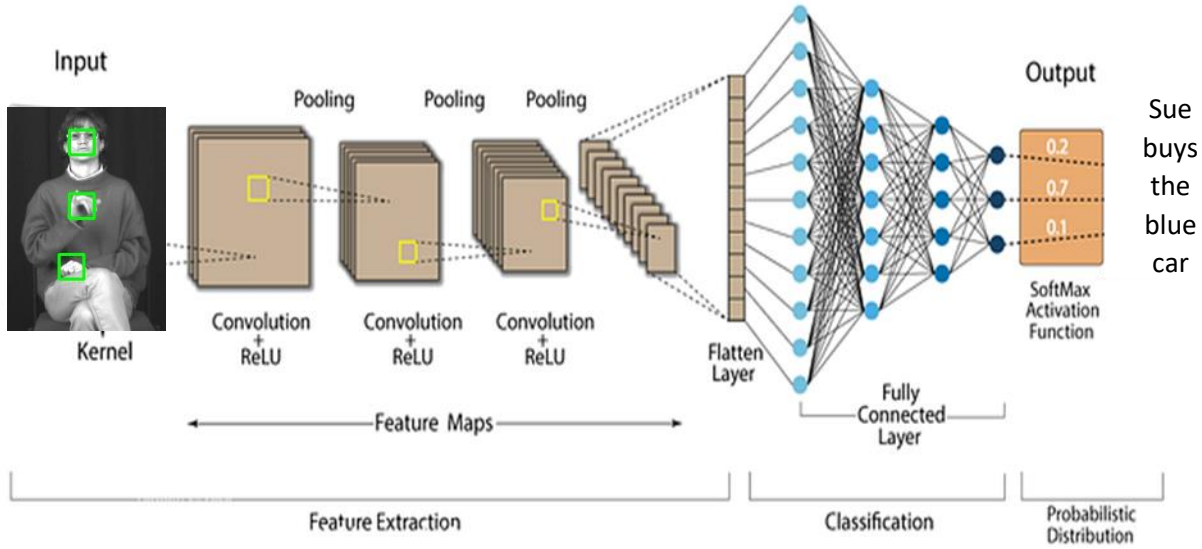
Network Architecture – Convolutional Neural Network

Convolutional Neural Network (CNN) are popularly known for its performance in image recognition and pixel data processing. Given its utility in image recognition, we decided to make use of the CNN for image detection.

So, first thing first, the input data preparation. We prep up the input data i.e. video sourced from the database and the annotation files for corpus data. Next is to use the data for training and to convert and shape all the frames (average 90 per video) dataset so it can be read by the algorithm. To initiate the process, we import the required packages mainly 'Tensorflow'– open-source Machine Learning library and 'Keras' – high level deep learning API.

Moving on to framing and hand detection, which reads the hand and head coordinate data provided with the dataset, makes a bounding box around each hand and head for each frame and saves the frame to the output folder as image.

The concept of a bounding box holds significance in various image classification and analysis tasks. This box serves as a critical element, allowing the model to concentrate specifically on the relevant section of the image required for its function. The absence of a bounding box can result in the algorithm identifying patterns in inappropriate areas, leading to inaccurate outcomes. To illustrate, during the training phase, the absence of a bounding box may prompt the model to associate features of an image with an incorrect label.



The architecture comprises an input layer, two convolutional layers, two rectified linear units (ReLu), two stochastic pooling layers, one dense layer, and a SoftMax output layer. Filters are applied successively, refining the output and increasing detail. In the CNN design, convolutional layers with varying pooling sizes, an activation function, and a rectified linear unit managing non-linearity are employed.

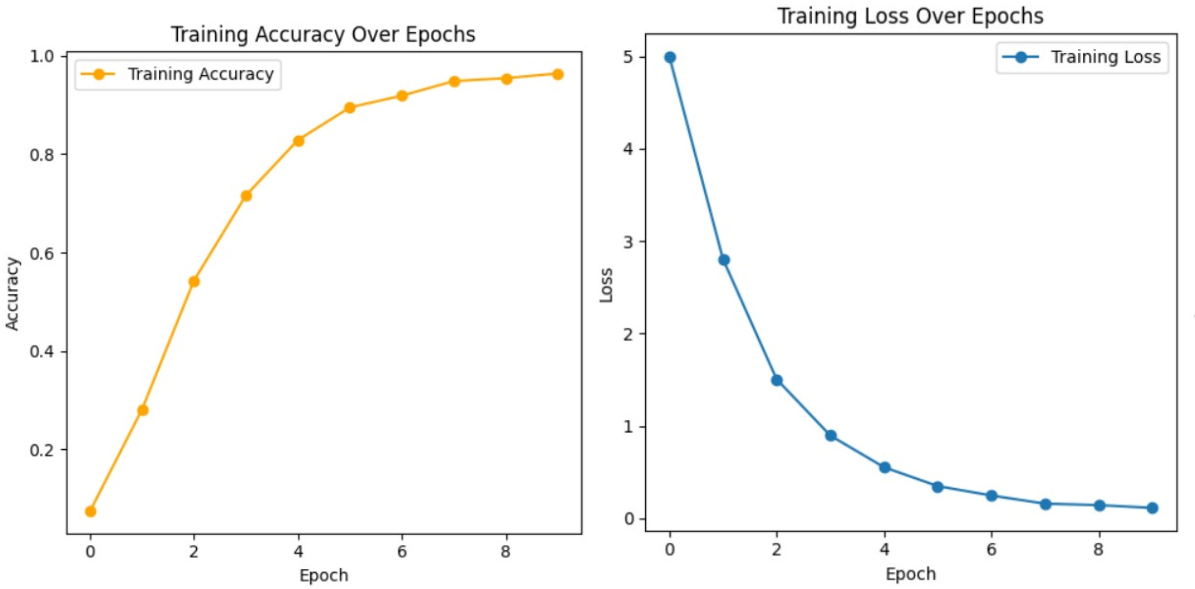
Each layer trains the CNN to recognize specific aspects of an input image. Complexity of filters increases with each layer until the CNN can successfully recognize the complete object in the final fully connected layer. The flattening layer further transforms the input into a vector before establishing connections with a set of fully connected layers.

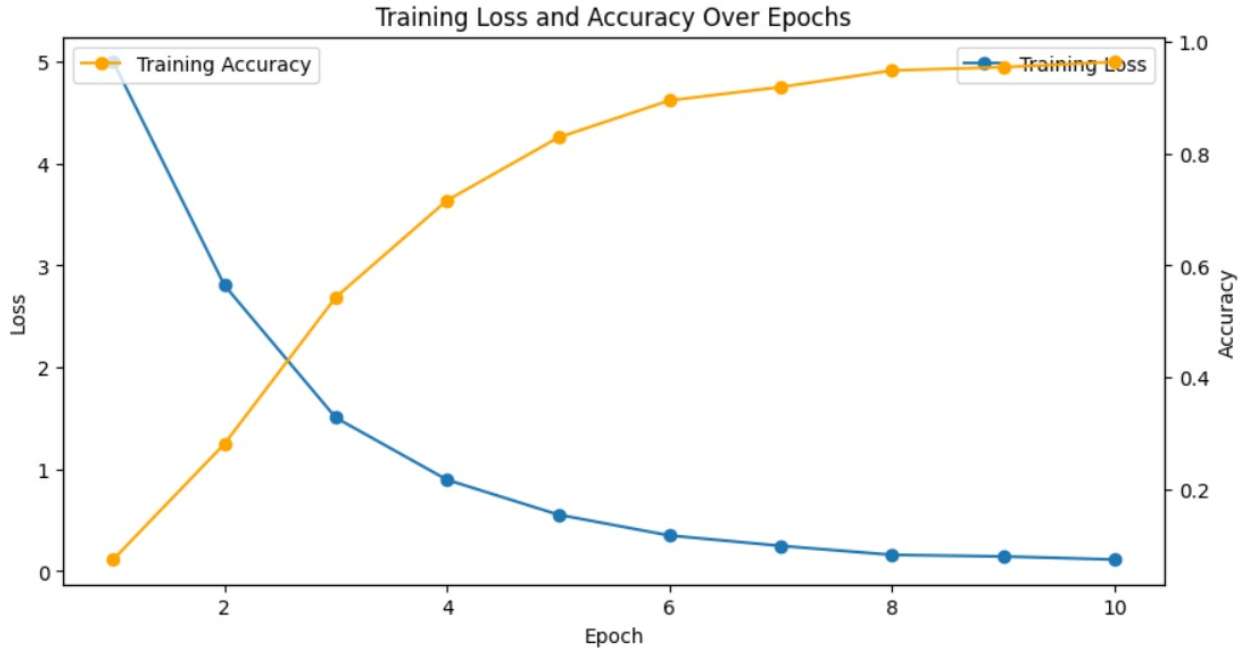
The output is given out in the form of text using the Softmax layer.

The network undergoes training to discern the unique characteristics of each hand gesture and out the output with each character' probability.

While deploying the model, we faced some challenges, some models report poor accuracy with pooling layers. Also, there are some discrepancies reported with similar hand and head movement leading to misclassification.

In the graphs below, it illustrates the epoch wise loss and accuracy in the training. As epochs advance, accuracy rises, and loss diminishes, affirming positive outcomes from the training process.





Performance Overview:

The proposed systems showcase training accuracy levels ranging from 84.68% to 96.37%. While the best performance was of one model which achieved the highest accuracy of 96.37%.

```
PS D:\IBA\Deep Learning\Project> python index.py
2024-01-06 16:50:09.542867: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performa
nce-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Enter video path: rwth-boston-104/videoBank/camera0/119_0.mpg
Video Path: rwth-boston-104/videoBank/camera0/119_0.mpg
Video Name: 119_0
Output Folder: output/hand-detection-frames\119_0
3/3 [==============================] - 0s 12ms/step
Predictions for the video:
Sue buys the blue car.
PS D:\IBA\Deep Learning\Project>
```

Looking ahead, we aim to deploy our current model for live video detection. Additionally, we plan to integrate language models, explore twirling functionalities, and incorporate LSTM and attention mechanisms to further enhance accuracy in our future endeavors.

For the project implementation and code, you can refer to the following github profile: https://github.com/WajeehaParker/Sign-Language-Translation/tree/master

---

References:

I. Speech Recognition Techniques for a Sign Language Recognition System
Philippe Dreuw, David Rybach, Thomas Deselaers, Morteza Zahedi, and Hermann Ney

II. A Review on Sign Language Recognition Using CNN
Meena Ugale(B), Odrin Rodrigues Anushka Shinde, Kaustubh Desle, and Shivam Yadav

III. Efficient Approximations to Model-based Joint Tracking and Recognition of Continuous Sign Language
Philippe Dreuw, Jens Forster, Thomas Deselaers, and Hermann Ney