CNN BASED LIVE CONVERSION OF SIGN LANGUAGES TO TEXT USING FINGER GESTURES

Alen p shyju

Department Of Computer Applications, Lovely Professional University, Punjab

ABSTRACT:

This study presents an advanced convolutional neural network (CNN) architecture tailored for real-time translation of gestural finger gestures into text. The technology leverages finger gestures as a medium for expressing the 26 letters of the alphabet in American Sign Language (ASL), with each gesture automatically translated into its corresponding text format. The CNN framework is meticulously designed to accurately capture and interpret the intricate nuances of finger movements, enabling seamless and efficient conversion to textual representations. Through the integration of cutting-edge neural network techniques, this system aims to enhance accessibility and communication for individuals utilizing gestural ASL, bridging gaps in language comprehension and fostering inclusivity in diverse linguistic environments.

Keywords

Convolutional neural network, Artificial Neural Network, Unsupervised Learning, Supervised Learning, Pooling, Reinforcement Learning, TensorFlow, Keras, OpenCV American Sign Language

1.INTRODUCTION

American Sign Language is a predominant sign language. Since the only disability that deaf and dumb people (hereinafter D&M) suffer from is related to communication and they cannot use spoken languages, the only way to communicate is through sign language. Communication is the process of exchanging thoughts and messages in various forms, such as words, signals, behaviours, and images. D&M people use their hands to express different gestures to express their ideas with other people. Gestures are messages exchanged nonverbally and these gestures are understood by vision. This non-verbal communication of deaf and dumb people is called sign language. A sign language is a language that uses gestures rather than sounds to convey meaning by combining the shape of the hands, the orientation and movement of the hands, arms or body, facial expressions, and lip patterns.

Minimizing the gap in verbal exchanges between D&M and non-D&M people becomes a desire to ensure effective conversation between everyone. Sign language translation is one of the fastest growing areas of research, allowing people with hearing impairments to communicate more naturally. A hand gesture recognition system offers deaf people the ability to talk to noisy people without the need for an interpreter. The system is designed for automated conversion of ASL to text and speech. The gestures we intend to train are those shown in the figure (1) below.

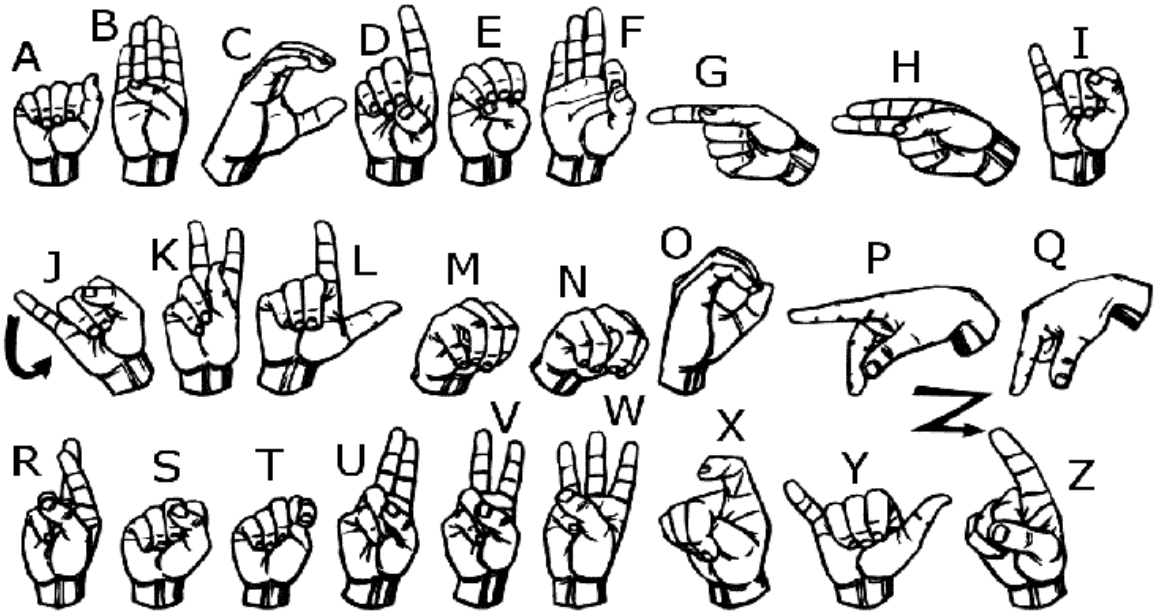

Figure 1

The 26 hand signs of the ASL Language.

2. LITERATURE SURVEY:

In recent years, there has been significant research conducted on hand gesture recognition.. With the help of literature survey, we realized that the basic steps in hand gesture recognition are: -

- Data acquisition
- Data pre-processing
- Feature extraction
- Gesture classification

2.1 Data acquisition:

The different approaches to acquire data about the hand gesture can be done in the following ways:

2.1.1 Use of Sensory Devices

Electromechanical devices are employed to precisely capture hand configuration and position. Various glove-based methods can be utilized to gather this information. However, this approach tends to be costly and less user-friendly.

2.1.2 Vision based approach:

Vision-based methods utilize a computer webcam as the input device to capture hand and/or finger information. By relying solely on a camera, Vision-Based methods facilitate natural interaction between humans and computers without the need for additional devices, thus reducing costs. These systems serve to complement human vision by simulating artificial vision systems implemented in software and/or hardware. The primary challenge of vision-based hand detection encompasses addressing the extensive variability in the appearance of the human hand due to a wide array of hand movements, diverse skin-color possibilities, as well as variations in viewpoints, scales, and camera capture speeds.

2.2 Data Pre-Processing

The hand detection approach combines threshold-based color detection with background subtraction [1]. An AdaBoost face detector can be utilized to distinguish between faces and hands, as both involve similar skin color.

Additionally, the required image for training can be extracted by applying a filter known as Gaussian Blur (or Gaussian smoothing) [2]. This filter can be conveniently applied using the OpenCV (Open Computer Vision) library.

2.3 Feature extraction for vision-based approach:

To extract the necessary images for training, instrumented gloves can be utilized, as mentioned in [3]. This approach helps reduce computation time for pre-processing and provides more precise and accurate data compared to applying filters on data obtained from video extraction.

In our attempts to perform hand segmentation using colour segmentation techniques, we encountered challenges due to the high dependence of skin colour and tone on lighting conditions. As a result, the segmentation outputs we obtained were not satisfactory. Furthermore, considering the large number of symbols to be trained for our project, many of which bear resemblance to each other (such as the gesture for the symbol 'V' and the digit '2'), we made the decision to maintain a stable, single-color background for the hand instead of segmenting it from a random background. This approach aims to yield improved results by eliminating the need to segment based on skin colour and enhance the accuracy of our large symbol set.

2.4 Gesture Classification:

In [4], Hidden Markov Models (HMM) are employed for the classification of gestures, focusing on the dynamic aspects of gestures. The gestures are derived from a sequence of video images by tracking skin-color blobs corresponding to the hand into a body-face space centred on the user's face.

The goal is to distinguish between two categories of gestures: deictic and symbolic. An image undergoes filtering using a rapid look-up indexing table. After the filtering process, skin-coloured pixels are clustered into blobs, which are statistical entities determined by the location (x, y) and the colorimetry (Y, U, V) of the skin-coloured pixels in order to identify uniform areas. In [5], a Naive Bayes Classifier is utilized as an effective and rapid method for static hand gesture recognition. This approach involves classifying different gestures based on geometric invariants obtained from image data post-segmentation. Unlike many other recognition methods, this approach is not reliant on skin colour. Gestures are extracted from each frame of the video against a static background. The initial step involves segmenting and labelling the objects of interest, followed by extracting geometric invariants. The subsequent step entails classifying gestures using a K-nearest neighbour algorithm aided by a distance

weighting algorithm (KNNDW) to provide suitable data for a locally weighted Naïve Bayes classifier.

3. METHODOLOGY:

The system utilizes a vision-based approach, where all signs are conveyed using bare hands, thereby eliminating the need for artificial devices for interaction.

3.1 Data Set Generation:

In our project, we attempted to locate existing datasets but were unable to find raw image datasets that met our criteria. The only datasets found were in the form of RGB values. So, I created my own dataset. The steps we took to generate our dataset are as follows:

I used the OpenCV (Open Computer Vision) library to create our dataset. Initially, we captured approximately 800 images of each ASL (American Sign Language) symbol for training purposes and approximately 200 images per symbol for testing purposes. Firstly, I captured each frame displayed by the webcam of our machine. Within each frame, I defined a Region of Interest (ROI) denoted by a blue bounded square figure [3].

Figure 3

Region of Interest (ROI) denoted by a blue bounded square.

Afterward, we apply a Gaussian Blur Filter to our image, which assists us in extracting various features. The Figure [4], following the application of the Gaussian Blur, appears as follows:

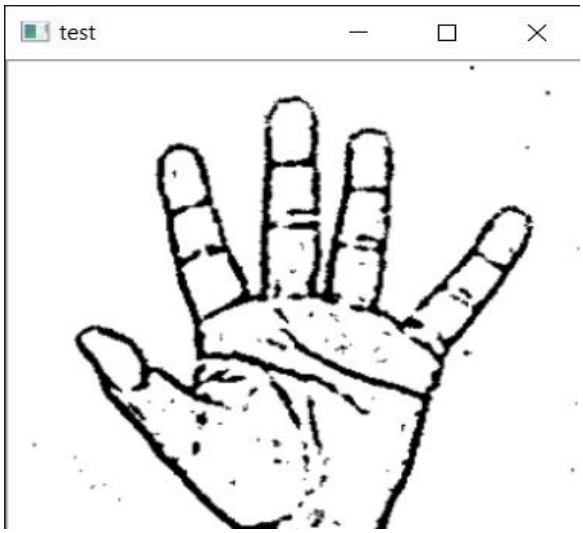

Figure 4

The application of the gaussian blur filter

3.2 Gesture Classification:

Our method employs two layers of algorithms to predict the user's final symbol.

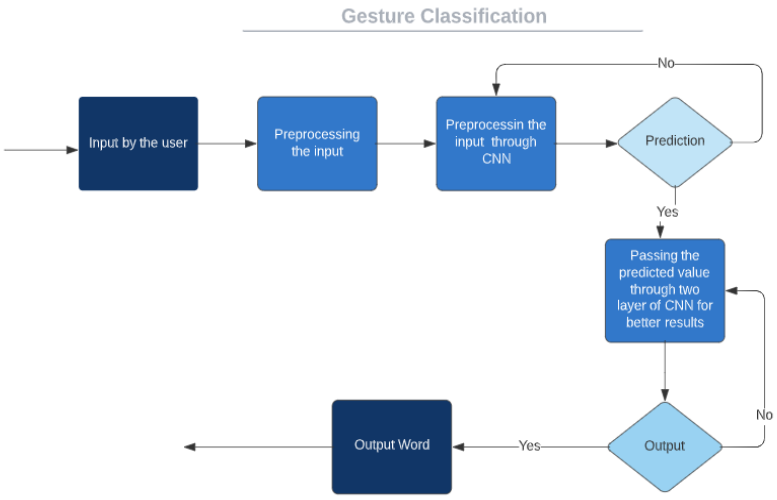

Figure 5

Gesture Classification

Algorithm Layer 1:

1. Apply a Gaussian Blur filter and threshold to the frame captured with OpenCV to obtain the processed image after feature extraction.
2. The processed image is fed into the CNN model for prediction. If a letter is detected for more than 50 frames, then the letter is recognized and considered for word formation.
3. The space between words is determined using the blank symbol.

Algorithm Layer 2:

1. We detect various sets of symbols that exhibit similar detection results.
2. Subsequently, we differentiate between those sets using classifiers specifically designed for each set.

Layer 1:

CNN Model:

1. 1st Convolution Layer: The input image has a resolution of 128x128 pixels. It undergoes processing in the first convolutional layer using 32 filter weights (3x3 pixels each), resulting in a 126x126 pixel image for each filter weight.

2. 1st Pooling Layer: The images are downsampled using max pooling of 2x2, reducing the image to 63x63 pixels by retaining the highest value in each 2x2 square of the array.

3. 2nd Convolution Layer: The output from the first pooling layer (63x63) serves as input to the second convolutional layer, which uses 32 filter weights (3x3 pixels each) and results in a 60x60 pixel image.

4. 2nd Pooling Layer: The resulting images are further downsampled using max pooling of 2x2, reducing the resolution to 30x30 images.

5. 1st Densely Connected Layer: The down sampled images are input to a fully connected layer with 128 neurons, and the output from the second layer is reshaped to an array of 30x30x32 = 28800 values. The input to this layer is an array of 28800 values, and the

output is fed to the 2nd Densely Connected Layer. A dropout layer with a value of 0.5 is utilized to prevent overfitting.

6. 2nd Densely Connected Layer: The output from the 1st Densely Connected Layer serves as input to a fully connected layer with 96 neurons.
7. Final Layer: The output of the 2nd Densely Connected Layer serves as input for the final layer, which will have the same number of neurons as the number of classes being classified (alphabets + blank symbol).

- Activation Function:

We have employed the ReLU (Rectified Linear Unit) in each of the layers (convolutional as well as fully connected neurons). ReLU calculates max (x, 0) for each input pixel, adding nonlinearity to the formula and aiding in learning more complex features. It helps in mitigating the vanishing gradient problem and expedites training by reducing computation time.

- Pooling Layer:

Max pooling is applied to the input image with a pool size of (2, 2) alongside the ReLU activation function. This reduces the number of parameters, thereby lowering the computation cost and mitigating overfitting.

- Dropout Layers:

To address overfitting, a dropout layer "drops out" a random set of activations in that layer by setting them to zero. This ensures that the network can provide the correct classification or output for a specific example even if some activations are dropped out.

- Optimizer:

The Adam optimizer is utilized for updating the model in response to the output of the loss function. The Adam optimizer combines the advantages of two stochastic gradient descent algorithms: adaptive gradient algorithm (ADA GRAD) and root mean square propagation (RMSProp).

Layer 2:

We employ two layers of algorithms to verify and predict symbols that are closely similar to each other, allowing us to detect the displayed symbol as accurately as possible. In our testing,

we observed that certain symbols were not being properly identified and were also being associated with other symbols:

1. For D: R and U

2. For U: D and R

3. For I: T, D, K, and I

4. For S: M and N

To address the above cases, we developed three distinct classifiers for classifying these sets:

1. {D, R, U}

2. {T, K, D, I}

3. {S, M, N}

3.3 Finger Spelling Sentence Formation Implementation:

1. When the count of a detected letter exceeds a specific value and no other letter is close to it within a certain threshold, we print the letter and add it to the current string (In our code, the value was set to 50 and the difference threshold to 20).

2. Otherwise, we clear the current dictionary containing the count of detections of the present symbol to prevent the possibility of an incorrect letter being predicted.

3. When the number of detected blank (plain background) exceeds a certain threshold and the current buffer is empty, no spaces are detected.

4. In other cases, it predicts the end of a word by printing a space, and the current buffer is appended to the sentence below.

3.4 AutoCorrect Feature:

We utilize a Python library called Hunspell suggest recommending correct alternatives for each (incorrect) input word. This allows us to display a set of words matching the current word, enabling the user to select a word to append it to the current sentence. This feature aids in reducing spelling mistakes and assists in predicting complex words.

3.5 Training and Testing:

We convert our input images from RGB to grayscale and apply Gaussian blur to eliminate unnecessary noise. Subsequently, we apply adaptive thresholding to extract our hand from the background and resize our images to 128 x 128.

After preprocessing, we feed the input images to our model for both training and testing, following all the aforementioned operations.

The prediction layer estimates the likelihood of the image falling under one of the classes. The output is normalized between 0 and 1, ensuring that the sum of each value in each class totals to 1. This normalization is achieved using the SoftMax function.

Initially, the output of the prediction layer may be somewhat distant from the actual value. To improve this, we have trained the networks using labelled data. Cross-entropy is utilized as a performance measurement in the classification. It is a continuous function that is positive at values which do not match the labelled value and is zero when it is equal to the labelled value. Therefore, we optimize the cross-entropy by minimizing it as close to zero as possible. In our network layer, we adjust the weights of our neural networks to achieve this, and TensorFlow has an inbuilt function to calculate the cross-entropy.

Having identified the cross-entropy function, we optimize it using Gradient Descent, in fact, with the best gradient descent optimizer known as the Adam Optimizer.

4. RESULTS

We have achieved a 95.8% accuracy using only layer 1 of our algorithm. When combining layer 1 and layer 2, we achieve an accuracy of 98.0%, surpassing the accuracy of most current research papers on American Sign Language. Many of these papers focus on hand detection using devices like Kinect. For instance, in [7], a recognition system for Flemish sign language using convolutional neural networks and Kinect achieved a 2.5% error rate. In [8], a recognition model using hidden Markov model classifier and a vocabulary of 30 words achieved a 10.90% error rate. The average accuracy for 41 static gestures in Japanese sign language was 86% in reference [9]. Furthermore, using depth sensors, the project cited in reference [10] achieved an accuracy of 99.99% for observed signers and 83.58% and 85.49% for new signers, also employing a CNN for their recognition system.

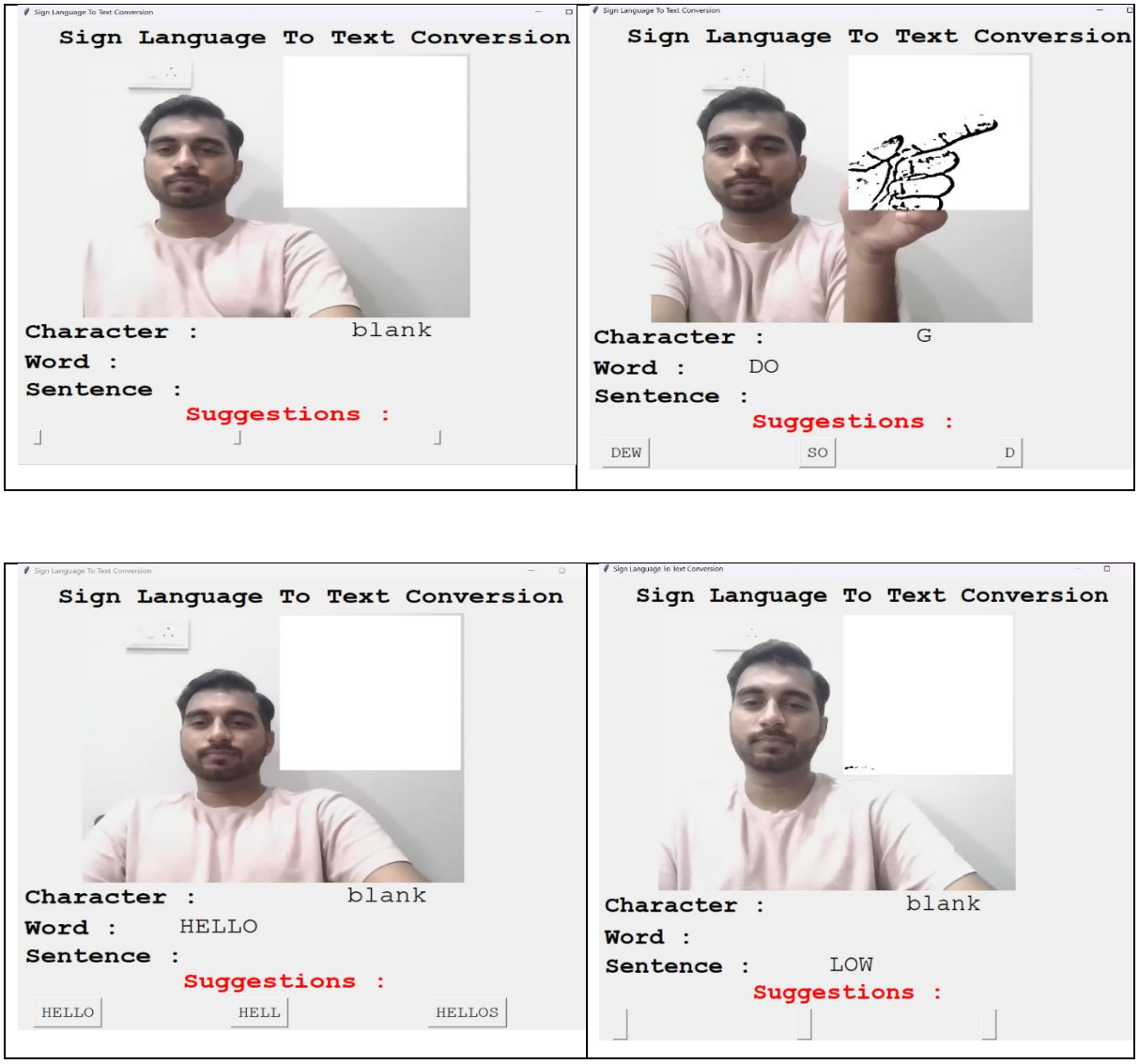

Figure 6

Output Obtained

It's important to note that our model does not utilize any background subtraction algorithm, unlike some of the models mentioned above. Therefore, implementing background subtraction in our project may lead to varying accuracies. Furthermore, while most of the above projects use Kinect devices, our main goal was to create a project that can be used with easily accessible resources. The Kinect sensor is not only less readily available, but also expensive for most users to purchase. Our model uses a normal laptop webcam, which is a significant advantage.

5. FUTURE SCOPE

We plan to improve accuracy in recognizing gestures against complex backgrounds by experimenting with different background subtraction algorithms. Additionally, we aim to enhance pre-processing to enable accurate gesture prediction in low-light conditions. We also intend to develop a web/mobile application for convenient access. While the current project focuses on American Sign Language (ASL), we aim to expand its capabilities to include other native sign languages with adequate datasets and training. Although the project currently focuses on finger spelling translation, sign languages are contextual and can convey objects or verbs. Recognizing this contextual signing will require advanced processing and natural language processing (NLP).

6. CONCLUSION

This report presents the development of a practical real-time vision-based American Sign Language recognition system for individuals who are deaf and mute, specifically focusing on ASL alphabets. We have attained a final accuracy of 98.0% on our dataset. Our prediction accuracy has been enhanced through the implementation of two additional layers of algorithms, allowing us to verify and predict symbols that closely resemble each other. This enhancement enables us to detect nearly all symbols, provided that they are displayed appropriately, without background noise, and with sufficient lighting.

7. REFERENCES

[1] Yang, J., & Xu, Y. (1994). *Hidden markov model for gesture recognition* (p. 0027). Carnegie Mellon University, the Robotics Institute.

[2]https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html

[3] Bazaz, R. K., Mishra, K., Jaiman, M., Stanislaus, V., & Dhamnaskar, P. (2023, October). Real Time Conversion Of Sign Language To Text and Speech (For Marathi and English). In *2023 International Conference on Advanced Computing Technologies and Applications (ICACTA)* (pp. 1-5). IEEE.

[4] Kadous, M. W. (1996, October). Machine recognition of Auslan signs using PowerGloves: Towards large-lexicon recognition of sign language. In *Proceedings of the Workshop on the Integration of Gesture in Language and Speech* (Vol. 165, pp. 165-174). Wilmington: DE.

[5] aeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural Networks-Part-2/

[6] Pigou, L., Dieleman, S., Kindermans, P. J., & Schrauwen, B. (2015). Sign language recognition using convolutional neural networks. In *Computer Vision-ECCV 2014 Workshops: Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part I 13* (pp. 572-578). Springer International Publishing.

[7] http://www-i6.informatik.rwth-aachen.de/~dreuw/database.php

[8] Kang, B., Tripathi, S., & Nguyen, T. Q. (2015, November). Real-time sign language fingerspelling recognition using convolutional neural networks from depth map. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* (pp. 136-140). IEEE.

[11] Number System Recognition (https://github.com/chasinginfinity/number-sign-recognition)

[9] Mukai, N., Harada, N., & Chang, Y. (2017, June). Japanese fingerspelling recognition based on classification tree and machine learning. In *2017 Nicograph International (NicoInt)* (pp. 19-24). IEEE.

[10] Zaki, M. M., & Shaheen, S. I. (2011). Sign language recognition using a combination of new vision based features. *Pattern Recognition Letters*, *32*(4), 572-577.

[12] https://opencv.org/

[13] https://en.wikipedia.org/wiki/TensorFlow