B.Tech MAJOR PROJECT

ML Based Real-time Hand Gesture Recognition System For Computer Application Control

Supervisor
Sh. Rakesh Bairathi

Prepared By
Eshaan Gupta
Vaibhav Singh
Rupesh Kumar
Vaibhav Prakash Sharaf

Table of Contents

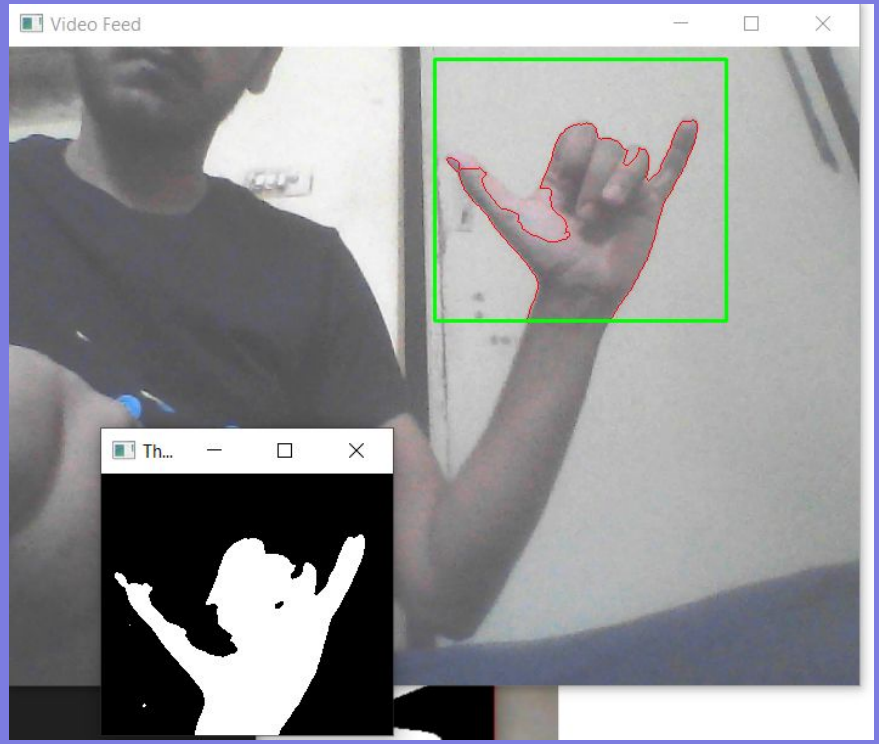
Idea

- The idea of gesture recognition is to develop a technique which can identify particular human gestures and use them to transfer information.
- The idea of developing hand gesture recognition procedures is to develop an interaction between human and computer and the recognized gestures are used to control the meaningful information.
- It is divided into two categories
    - Static gesture
    - Dynamic gesture.

Different Approach

- **Appearance based approaches**
  - The visual appearance of the input hand image is modeled using the feature extraction, which are then compared to the feature extracted of the stored image.
  - The general method of this approach is to detect skin colored regions in an image.
- **3D model based approaches**
  - In this approach for modeling and analysis of hand shape the 3D model is used. While doing the 2D projection some data is lost, So intended a depth parameter is added in the 3D model to make it more accurate.
- **Gloved based approaches**
  - Glove approach uses the sensors for capturing the hand position and motion.
  - Detection of the hand is done by the sensors on hand and the correct coordinates of location of palm and fingers is to be found using the sensors on the gloves
- **Marked Colored glove approach**
  - In this approach the gloves which are to be worn by the human hand are made marked by the colors to direct the process of tracking the hand and locating the palm and fingers, which will provide the exact geometric features resulting in formation of hand shape.

Differentiating foreground from background



SEPARATING FOREGROUND FROM BACKGROUND

- This image describes the semantic segmentation problem where the objective is to find different regions in an image and tag its corresponding labels. In this case, “sky”, “person”, “tree” and “grass”.

SEPARATING FOREGROUND FROM BACKGROUND

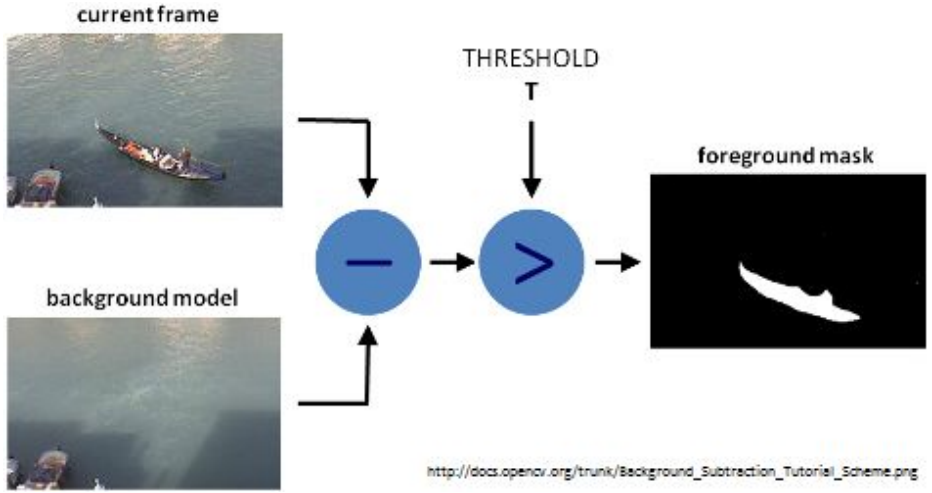
- We are going to recognize hand gestures from a video sequence. To recognize these gestures from a live video sequence, we first need to take out the hand region alone removing all the unwanted portions in the video sequence.
- Our first step to find the hand region from a video sequence involves three simple steps.
    1. Background Subtraction
    2. Motion Detection and Thresholding
    3. Contour Extraction

BACKGROUND SEPARATION

- First, we need an efficient method to separate foreground from background. To do this, we use the concept of running averages.
- We make our system to look over a particular scene for 30 frames. During this period, we compute the running average over the current frame and the previous frames.
- After figuring out the background, we bring in our hand and make the system understand that our hand is a new entry into the background, which means it becomes the foreground object.

BACKGROUND SEPARATION

- After figuring out the background model using running averages, we use the current frame which holds the foreground object (hand in our case) in addition to the background. We calculate the absolute difference between the background model (updated over time) and the current frame (which has our hand) to obtain a difference image that holds the newly added foreground object (which is our hand).



Motion Detection and Thresholding

- To detect the hand region from this difference image, we need to threshold the difference image, so that only our hand region becomes visible and all the other unwanted regions are painted as black. This is what Motion Detection is all about.
- Thresholding is the assignment of pixel intensities to 0's and 1's based a particular threshold level so that our object of interest alone is captured from an image.

Contour Extraction

- After thresholding the difference image, we find contours in the resulting image. The contour with the *largest area* is assumed to be our hand.
- Contour is the outline or boundary of an object located in an image.

Background elimination

Implementation

Running Average

- Function used to compute the `running average` between the background model and the current frame.
- This function takes in two arguments - `current frame` and `aWeight`.
- Running average is calculated using the formula given below:

$$dst(x, y) = (1 - a).\, dst(x, y) + a.\, src(x, y)$$

- $src(x, y)$ - Source image or input image (1 or 3 channel, 8-bit or 32-bit floating point)
- $dst(x, y)$ - Destination image or output image (same channel as source image, 32-bit or 64-bit floating point)
- a - Weight of the source image (input image)

Motion Detection and Thresholding

- We threshold the difference image to reveal only the hand region. Finally, we perform contour extraction over the thresholded image and take the contour with the largest area (which is our hand).
- We return the thresholded image as well as the segmented image as a tuple. The math behind thresholding is pretty simple. If x(n) represents the pixel intensity of an input image at a particular pixel coordinate, then threshold decides how nicely we are going to segment/threshold the image into a binary image.

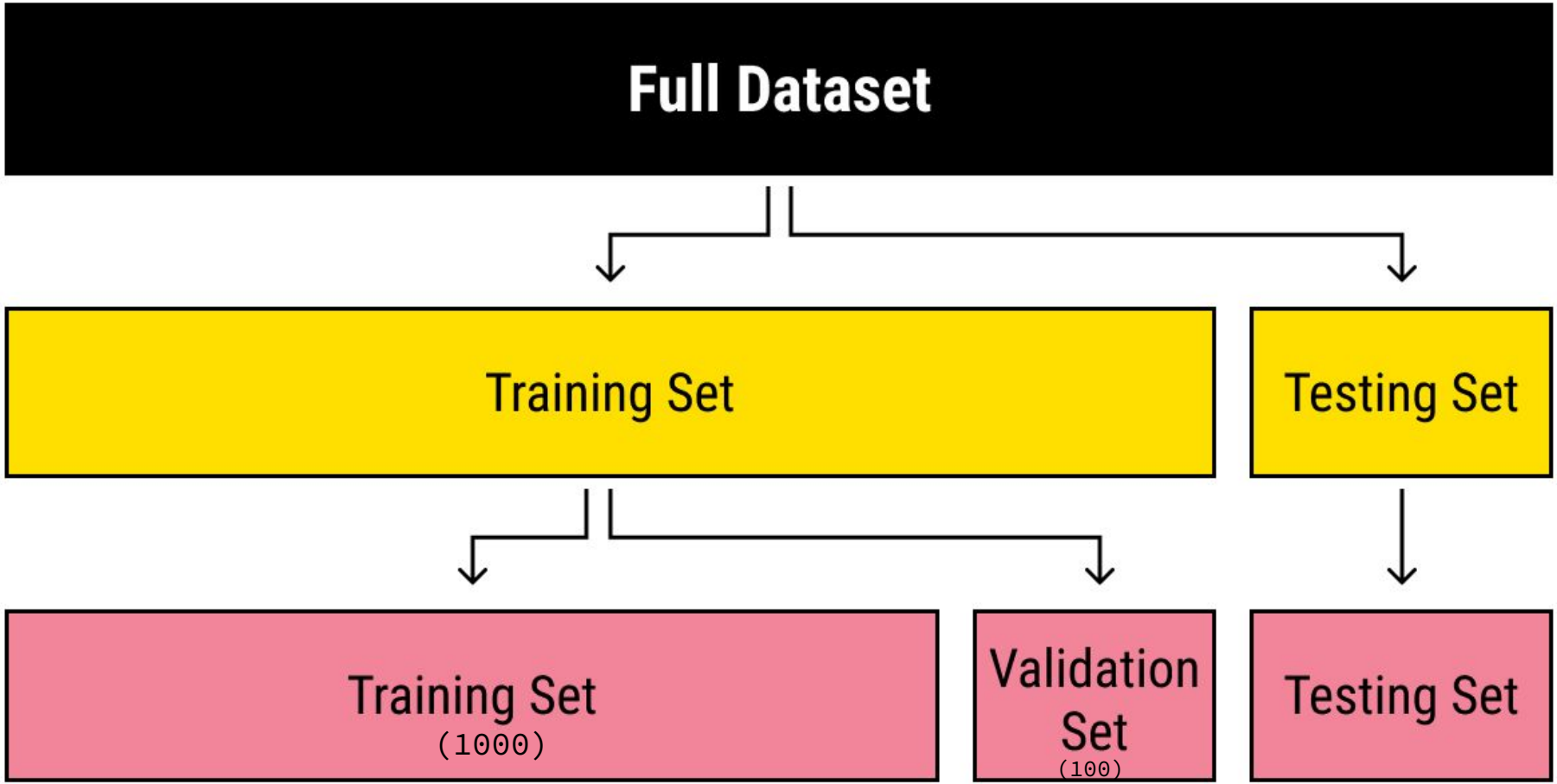
$$x(n) = \begin{cases} 1, & \text{if } n >= threshold \\ 0, & \text{if } n < threshold \end{cases}$$

PreparING

DATASET

Identify gestures

We want our model to train at six different gestures that are


Full Dataset
Training Set
Testing Set
Training Set
(1000)
Validation Set
(100)
Testing Set

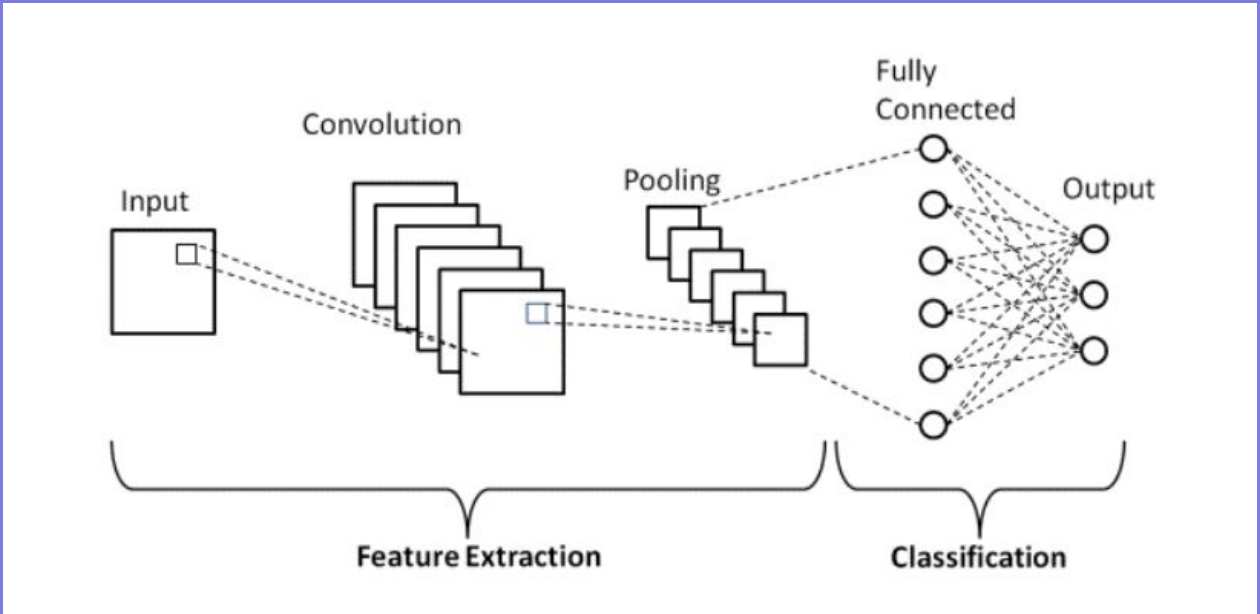
Preparation of dataset

- We have collected 1000 images for training the model.
- For each gesture 100 images are used for validification.
- Threshold taken : 0.5
- Images are resized to dimension
    - 89 X 100


```
C:\Users\ajeya\Desktop\gesture\Dataset>tree
Folder PATH listing for volume OS
Volume serial number is 1EF5-CDC3
C:.
├───FistImages
├───FistTest
├───LeftImages
├───LeftTest
├───PalmImages
├───PalmTest
├───RightImages
├───RightTest
├───SwingImages
├───SwingTest
├───ThumbImages
└───ThumbTest
```

File structure of dataset

TRAINING THE MODEL



TensorFlow

- TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.
- Object Detection using Tensorflow is a computer vision technique. As the name suggests, it helps us in detecting, locating, and tracing an object from an image or a video.

TensorFlow

TensorFlow Object Detection API

- The TensorFlow Object Detection API is an open-source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models.
- There are already pre-trained models in their framework which are referred to as Model Zoo.
- It includes a collection of pre-trained models trained on various datasets such as the
    - COCO (Common Objects in Context) dataset
- These various models have different architecture and thus provide different accuracies but there is a trade-off between speed of execution and the accuracy in placing bounding boxes.

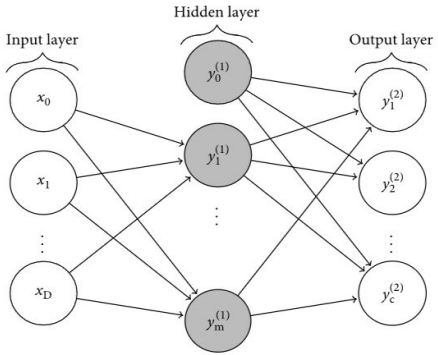
Artificial Neural Networks

Artificial neural networks are structures widely used for classification tasks.

The object to be classified is presented to the network through the activation of artificial neurons in the input layer. These activations are processed in the inner layers, and the result emerges as a pattern in the output layer.

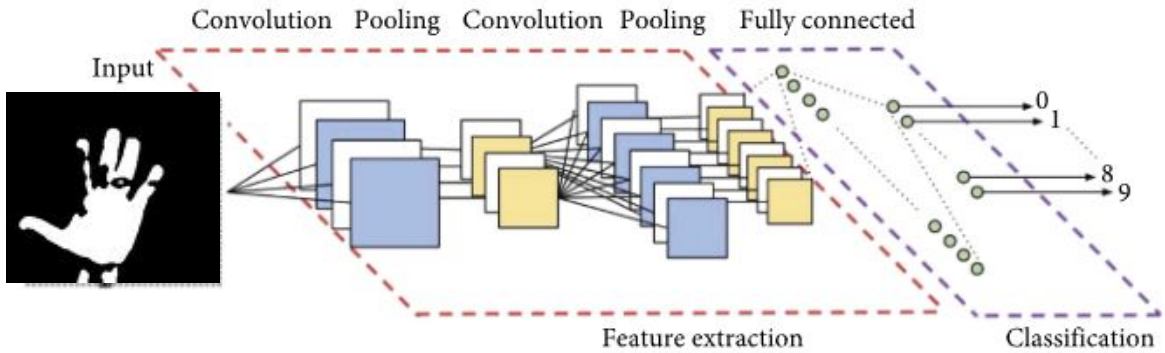
A network that has a single layer is known as a simple perceptron but is only capable of solving linearly separable problems.

In order to solve non linearly separable problems, it is necessary to use a multilayer perceptron (MLP) neural network. The MLP consists of an input layer, a number of hidden layers, and an output layer, as can be seen in Figure 1.



Convolutional Neural Network

- The convolutional neural network, or CNN for short, is a specialized type of neural network model designed for working with two-dimensional image data.
- There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers.



Components of CNN

- Convolutional Layer
- Pooling Layer
- Fully Connected Layer
- Dropout
- Activation Function

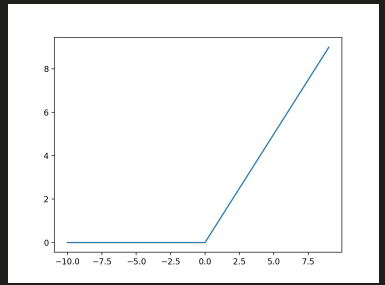
OUR MODEL

CNN Model

- The network contains **7** hidden convolution layers with **Relu** as the activation function and **1** Fully connected layer.
- The network is trained across **50** iterations with a batch size of **64**.
- We saw that 50 iterations trains the model well and there is no increase in validation accuracy along the lines so that should be enough.
- The model achieves an accuracy of **96.6%** on the validation dataset.
- The ratio of training set to validation set is **1000 : 100.**

```
# CNN Model
# tf.reset_default_graph()
convnet=input_data(shape=[None,89,100,1],name='input')
convnet=conv_2d(convnet,32,2,activation='relu')
convnet=max_pool_2d(convnet,2)
convnet=conv_2d(convnet,64,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,128,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,256,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,256,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,128,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=conv_2d(convnet,64,2,activation='relu')
convnet=max_pool_2d(convnet,2)

convnet=fully_connected(convnet,1000,activation='relu')
convnet=dropout(convnet,0.75)

# class size = 4
convnet=fully_connected(convnet,6,activation='softmax')

convnet=regression(convnet,optimizer='adam',learning_rate=0.001,loss='categorical_crossentropy',name='regression')
```

Activation Function

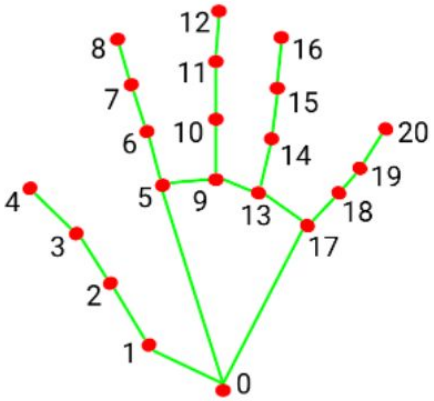


Pooling Layer

Dropout

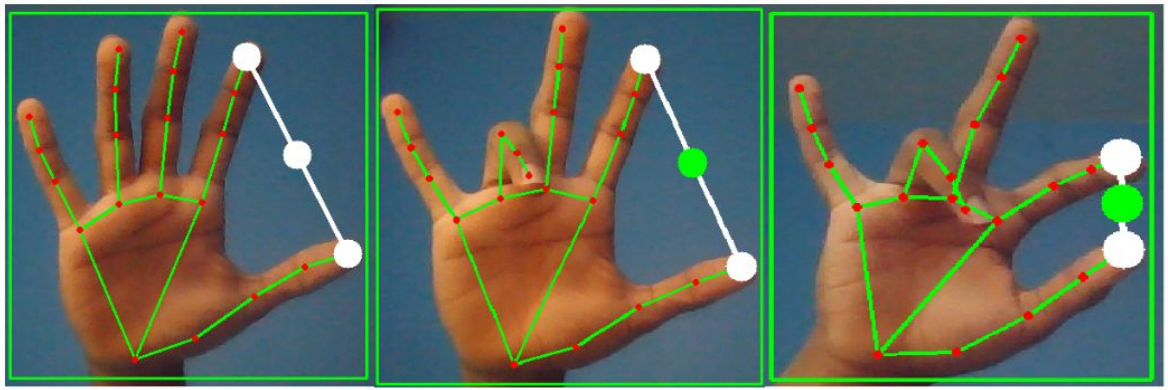
Volume Control Using Gesture

Hand Detection



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

Demo



Application

- Smart Cars
- Healthcare
- Virtual-reality
- Consumer electronics

Thank You

Bibliography

[1] TensorFlow 2.8.0. TensorFlow February 2, 2022 release GitHub documentation.

[2] .Kobylarz, Jhonatan; Bird, Jordan J.; Faria, Diego R.; Ribeiro, Eduardo Parente; Ekárt, Anikó (2020-03-07). "Thumbs up, thumbs down: non-verbal human-robot interaction through real-time EMG classification via inductive and supervised transductive transfer learning". *Journal of Ambient Intelligence and Humanized Computing*. Springer Science and Business Media LLC.

[3] Object Detection using Tensorflow Colab demonstration using a TF-Hub module trained to perform object detection.

[4] Wei Liu1 , Dragomir Anguelov , Dumitru Erhan SSD: Single Shot MultiBox Detector. Method for detecting objects in images using a single deep neural network

[5] Hand Gesture Recognition based on Shape Parameters

[6] Meenakshi Panwar and Pawan Singh Mehra , “Hand Gesture Recognition for Human Computer Interaction”, in Proceedings of IEEE International Conference on Image Information Processing(ICIIP 2011), Waknaghat, India, November 2011.

[7] Amornched Jinda-apiraksa, Warong Pongstiensak, and Toshiaki Kondo, ”A Simple Shape-Based Approach to Hand Gesture Recognition”, in Proceedings of IEEE International Conference on Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON), Pathumthani, Thailand , pages 851-855, May 2010

[8] A. Jinda-Apiraksa, W. Pongstiensak, and T. Kondo, “Shape-Based Finger Pattern Recognition using Compactness and Radial Distance,” The 3rd International Conference on Embedded Systems and Intelligent Technology(ICESIT 2010), Chiang Mai, Thailand, February 2010.