# INDIAN SIGN LANGUAGE DIGIT DETECTION USING MACHINE LEARNING

*Submitted in partial fulfillment of the requirements*

*for the award of the degree of*

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

(2017 – 2021)

**Submitted By**

Abdul Latif Faqeeri

(1709370)

**Supervised By**

Er. Ruchika Bindal

Professor

Department of Computer Science,

Quest Group of Institution affiliated to

I.K Gujral Punjab Technical University

*(INDIA)*

# Declaration of Originality of Project Work

| | |
|---|---|
| Student Name | Abdul Latif Faqeeri |
| Program of study | Bachelor of Technology |
| Roll Number | 1709370 |
| Course Title | Computer Science and Engineering |
| Topic of Work | Indian Sign Language Digit Detection using Machine Learning |
| Semester | 8th |
| Supervisor Name | Er. Ruchika Bindal |

I declare herewith that this above-mentioned work (Project) titled **Indian Sign Language Digit Detection using Machine Learning** is my own original work.

Furthermore, I confirm that:

- This work consists of my own account of investigation. I have documented all methods, data and process truthfully and I have not manipulated any data.
- All data and findings in the work have not been falsified or embellished.
- I am aware that plagiarism, is considered to be an act of fraudulence and an offence and alleged plagiarism, at whatever stage of student's studies, whether before or after graduation, will be investigated and dealt with appropriately by the University. I have committed none of the forms of plagiarism.
- I have clearly referenced in accordance with department requirement, in both the text and bibliography or references, all sources (either from a print source, internet or any other sources) used in the work.
- I have provided the source of all tables, figures, and data etc. that are not my own work.
- This work has not been previously, or concurrently, used either for other courses or within other exam processes as an exam work. This work has not been published as a thesis/project report elsewhere.
- I have not sought or used the services of any professional agencies to produce this work.
- I appreciate that any false claim in respect of this work will result in disciplinary action in accordance with University or departmental regulations.
- I understand that my work may be electronically or otherwise checked for plagiarism by the use of plagiarism detection software and stored on a university/UGC server for eventual future comparison.

Signature …………………… Date ……………..........

# ACKNOWLEDEMENT

First of all, we are grateful to the Almighty Allah for establishing in to complete this thesis.

I wish to express my deep gratitude and sincere thanks to my guide Er. Ruchika Bindal

, Department of Computer Science, Quest Group of Institution affiliated to I.K Gujral Punjab Technical University for his encouragement

and for all the facilities that he has provided for this research work. Without his guidance this

research work could not become a reality. I take this opportunity to express my deep sense of gratitude for his invaluable guidance, constant encouragement which has sustained my efforts at all stages of this research work.

I can't forget to offer my sincere thanks to my parents and also to my friends who helped us to carry out this research work successfully.

# Abstract

This thesis investigates the use of machine learning for the detection and recognition of digits in Indian Sign Language (ISL). ISL digit recognition is crucial for enabling seamless communication between hearing-impaired individuals and the broader community. The primary challenge addressed in this work is the accurate identification and classification of hand gestures representing numerical digits in ISL.

This research proposes a machine learning-based framework for detecting and recognizing ISL digits from hand gesture images or video sequences. The study involves the development and optimization of a machine learning model that performs two key tasks:

1. **Gesture Detection and Feature Extraction**: The model first identifies and extracts relevant features from input data, such as hand shape, orientation, and movement patterns, to detect digit gestures.
2. **Gesture Classification**: These extracted features are then used to classify the gestures into their respective numerical categories using a decision tree algorithm, chosen for its interpretability and efficiency.

The research emphasizes achieving high accuracy in detecting digits across diverse conditions, including variations in lighting, background, and signer styles. The proposed system's effectiveness is evaluated through extensive experimentation using publicly available and custom ISL datasets.

The successful implementation of this system can significantly enhance communication tools and assistive technologies for individuals relying on ISL. Furthermore, this work lays a foundation for future advancements in ISL gesture recognition systems, fostering inclusivity and accessibility in digital communication and education platforms.

# Table of Contents

# Table of Figures

# 1. Introduction

Human communication is a rich tapestry woven from words, gestures, and actions. Sign language, a vital form of communication for deaf and hard-of-hearing individuals, relies heavily on hand gestures to convey meaning. Understanding and interpreting these gestures is crucial for facilitating communication and fostering inclusivity.

This thesis focuses on the application of machine learning to recognize and classify numerical digits in Indian Sign Language (ISL). Digit recognition in ISL is an important subset of gesture interpretation, with potential applications in communication aids, education platforms, and assistive technologies. However, challenges such as varying hand shapes, orientations, signer styles, and environmental conditions make automated ISL digit recognition a complex task.

The proposed research aims to bridge this gap by developing a robust machine learning-based system for ISL digit detection. The system will analyze visual data, such as images or video frames, to detect hand gestures representing numerical digits and classify them accurately. To achieve this, the study will utilize advanced feature extraction techniques and decision tree algorithms for gesture recognition. Decision trees are chosen for their interpretability and effectiveness in handling classification tasks. The significance of this research lies in its potential to enhance the accessibility of communication tools for the hearing-impaired. By enabling automated ISL digit recognition, the proposed system can contribute to bridging the communication divide and fostering inclusivity. Moreover, the insights gained from this work can serve as a foundation for further advancements in sign language gesture recognition systems, paving the way for broader applications in human-computer interaction and assistive technology.

This thesis will begin by examining the existing literature on sign language recognition and gesture detection, with a particular emphasis on ISL. It will then describe the architecture of the proposed machine learning model, the data preprocessing techniques employed, and the training methodologies used to optimize model performance. Finally, the research will detail the evaluation strategies and metrics applied to assess the system’s accuracy and robustness under varying conditions. Through this effort, the study seeks to advance the field of ISL digit recognition and contribute to the development of inclusive technologies.

# 2. Literature Survey

Human activity recognition (HAR) finds applications across diverse sectors such as healthcare, sports, and security due to its adaptability. Over time, many methodologies have been investigated to address the challenge of accurately identifying human activities from sensor data. These methodologies encompass a range of machine learning algorithms, including deep learning, support vector machines, and decision trees. The overarching goal of HAR is to improve quality of life by enabling customized and preemptive interventions based on human behavioral patterns.

In their 2019 study [1], Abdulhamit Subasi, Kholoud Khateeb, Tayeb Brahimi, and Akila Sarirete proposed utilizing Machine Learning techniques for Human Activity Recognition in a Smart Healthcare Environment. They applied machine learning techniques to monitor elderly individuals' activities in rehabilitation centers and track daily activities in smart homes. Additionally, their work encouraged physical exercises in rehabilitation centers for various patient groups, including children with motor disabilities and post-stroke patients, by integrating HAR.

In their 2019 paper [2], Homay Danaei Mehr and Huseyin Polat proposed Human Activity Detection with Deep Learning Approach in Smart Homes, where human activity recognition is broadly categorized into vision-based and sensor-based methods. Sensor-based methods are further subdivided into wearable, object-tagged, and dense sensing approaches.

In their 2014 work [3] Sungyoung Lee, Li Bin, Zhongqi Zhang, and Jin Wang introduced a method incorporating Human Activity Recognition (HAR) into surveillance systems deployed in public areas such as banks or airports to mitigate potential criminal activities and hazardous behaviors. The study's results validated the effectiveness of the proposed techniques in the early identification of interactions involving two or more individuals.

Human activity recognition (HAR), employing computer and machine vision technology, deciphers human motion. Although its potential remains largely untapped, it has garnered considerable interest for its practicality and adaptability. In their 2013 paper [4], Hoang Le Uyen Thuc, Shian-Ru Ke, Young-Jin Lee, Jenq-Neng Hwang, Jang-Hee Yoo, and Kyoung-Ho Choi conducted a review on Video-Based Human Activity Recognition which states Typically, any algorithm comprises four fundamental stages:

1. Data acquisition from external or wearable sensors for training purposes
2. Pre-processing involving noise reduction or windowing
3. Feature extraction encompassing structural and statistical features
4. Classification utilizing machine learning and deep learning models

In their 2020 survey [5], Djamila Romaissa Beddiar et al. introduced "Vision-based Human Activity Recognition," underlining machine learning's crucial role. Various strategies and technologies gather information, but machine learning algorithms infer actions.

In a 2015 study [6] by Salvatore Gaglio et al., titled "Human Activity Recognition Process Using 3-D Posture Data," standard algorithms like SVM, KNN, and Random Forest were utilized. Emphasizing feature selection's importance, a robust feature set enhances results.

In their 2015 review [7], Ong Chin Ann and Bee Theng Lau presented "Human Activity Recognition," focusing on the extensive research into human activity detection using sensor technologies over the past decade. Motion sensors, gyroscopes, biosensors, Accelerometers, pressure, proximity, and other types of sensors are commonly used for activity recognition. Additionally, some sensors utilize radio technology.

In a separate study, Ling Chen, Xiaoze Liu, Liangying Peng, and Menghan Wu (2020) [8] presented wearable devices that use deep learning to recognize complex human activities across different senses. Providing a survey of existing research employing a vision-based approach for activity recognition. This literature is divided into two primary categories: unimodal and multimodal approaches. Unimodal methods utilize data from a single modality and are further classified into space-time-based, stochastic, shape-based, and rule-based methods. On the other hand, Multimodal approaches utilize data from various sources and are further categorized into behavioral, affective, and social-networking methods.

In their 2020 review [9], Shugang Zhang et al. introduced "A Review on Human Activity Recognition Using Vision-Based Method." They classified HAR techniques into action-based, interaction-based, and motion-based categories. Action-based includes gesture recognition, posture recognition, behavior recognition, collapse detection, and routine activities.

Anurag Arnab, Mostafa Dehghani, George Heigold, Chen Sun, Mario Lucic and Cordelia Schmid in their paper [10] “Vivit: A Video Vision Transformer” presented completely transformer-based video classification approach which was inspired by ViT. Self-attention is the main operation in the model that is computed on sequence of spatio-temporal tokens derived from the input video

In their 2017 paper [11], A.H. de Souza junior, Liandro B. Marinho and P.P.Reboucas Filho introduced "A New Approach to Human Activity Recognition Using Machine Learning Techniques," emphasizing the role of machine learning in activity recognition. After collecting information through diverse techniques and technologies, machine learning algorithms are tasked with inferring or recognizing actions.

In their 2016 paper [12], S.U.Park, J.H.Park, M.A.Al-Masni, M.A.Al-Antari, Md.Z.Uddin and T.S. Kim introduced the use of depth cameras to identify human activities for health and social care services through a deep learning recurrent neural network. Popular machine learning algorithms like SVM, KNN, Random Forest, Naive Bayes, and HMM are often applied in this field. Before deploying a machine learning algorithm, it is essential to conduct feature selection, as a robust feature set can enhance performance. Human activity recognition (HAR) finds diverse applications, including in healthcare for elderly individuals, intelligent environments, security and surveillance, human-computer interaction, indoor navigation, and retail experiences.

In their 2017 publication [13], Adrian Nunez-Marcos, Gorka Azkune, and Ignacio Arganda-Carreras investigated "Vision-Based Fall Detection with Convolutional Neural Networks," highlighting the significance of motion-related actions for presence identification, particularly within security and surveillance settings. They observed that RFID technology leads in this domain, providing economical and exact solutions. Motion-related activities are further categorized into tracking, motion detection, and people enumeration.

In their 2009 paper [14], Yang Wang and Greg Mori presented "Human Action Recognition by Semi Latent Topic Models," highlighting the importance of having a well-prepared dataset aligned with the range of activities of interest for any problem statement and its proposed solution. However, they noted that available datasets often feature animated sequences or foreign faces, which pose challenges for processing in Indian environments. With this challenge in mind, the authors aimed to construct an authentic dataset comprising various household activities, such as standing, sitting, sneezing, and crying, focusing on Indian faces and body types. They aim to utilize various proposed methods to test and compare accuracy levels.

# 3. Research Gaps

- Not a proper automated mechanism of detection of numerical digits in ISL.
- Human action recognition has been explored extensively but unfortunately very little have been done on sign language detection using action detection methods.
- A very limited research papers are available on automation of digits detection from ISL videos.

# 4. Project objectives

The main objective of the project is:

a) To create a Dataset for Numerical Digits in Indian Sign Language.

b) To analyze sing language videos and further extract the gesture landmarks.

c) To preprocess the sign language dataset.

d) To train a machine learning based model for detection of digits in Indian Sign Language.

e) To evaluate the model on live stream videos.

# 5. Research Methodology

## 5.1. Research Problem

The research problem addressed in this thesis is the lack of an effective and automated method for recognizing and classifying numerical digits in Indian Sign Language (ISL) from raw visual data. Accurate digit recognition is essential for bridging the communication gap between hearing-impaired individuals and the broader community. However, existing solutions for ISL digit detection face several challenges, including variations in hand gestures, lighting conditions, signer styles, and background noise.

Indian Sign Language digit gestures, as a critical component of sign language, require precise interpretation to enable the development of communication tools, educational platforms, and assistive technologies. Current approaches often involve manual or semi-automated methods for recognizing ISL gestures, which are time-consuming and prone to inaccuracies.

To address these limitations, this thesis proposes a machine learning-based solution that can automate the detection and classification of ISL digit gestures from image or video data. The research is structured around two primary tasks:

1. **Gesture Detection and Feature Extraction**: Developing a machine learning model to analyze visual data, detect hand gestures, and extract relevant features such as hand shape, movement, and orientation.
2. **Gesture Classification**: Using the extracted features, the model will classify the detected gestures into their corresponding numerical digit categories using a decision tree algorithm.

The success of this approach would enable the development of tools and applications that support:

- **Assistive Technologies**: Enhancing communication tools for hearing-impaired individuals by automating ISL digit recognition.
- **Educational Platforms**: Facilitating the learning and teaching of ISL through interactive systems capable of understanding and interpreting digit gestures.

- **Accessibility Solutions**: Improving accessibility in digital communication by integrating ISL digit recognition into broader sign language translation frameworks.

In summary, the research problem centers on the absence of a robust and automated method for detecting and classifying ISL digits, which is a critical step toward enabling inclusive technologies and advancing sign language recognition systems. This thesis aims to address this gap by developing a machine learning-based framework that achieves high accuracy and efficiency in ISL digit recognition.

.

## 5.2 Analyzing the Dataset

The dataset used in this research focuses exclusively on Indian Sign Language (ISL) digit gestures, covering numerical labels from zero to ten (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10). For each digit class, there are 20 video samples, resulting in a total of 220 videos in the dataset. All videos feature different individuals performing the corresponding sign language gestures against a green background, ensuring a consistent environment for gesture recognition.

Certain digit classes exhibit imbalanced characteristics due to multiple distinct representations of the same digit:

- **Class 6** has four distinct representations.
- **Classes 8, 9, and 10** each have three distinct representations.

The distribution of these distinct representations is uneven, adding complexity to the classification task. For instance, the varying number of samples for each representation within a class challenges the model to accurately learn and generalize across these variations.

Despite these challenges, this dataset provides a diverse and valuable resource for training and evaluating the proposed machine learning model. The inclusion of multiple signers and variations in gesture representation reflects real-world scenarios, ensuring that the developed system can handle variability and achieve robust digit recognition. The green background minimizes interference from environmental factors, further aiding the focus on gesture detection and classification.

By addressing the imbalances and leveraging the variability in the dataset, this research aims to develop an accurate and efficient system for recognizing ISL digit gestures, paving the way for improved communication tools for the hearing-impaired community.

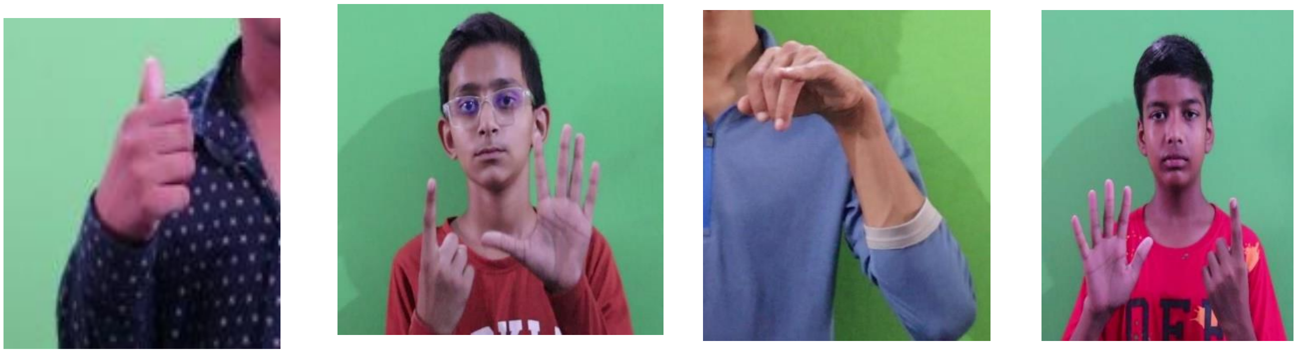


**Figure 1 - 4 different types of class 6 representation.**

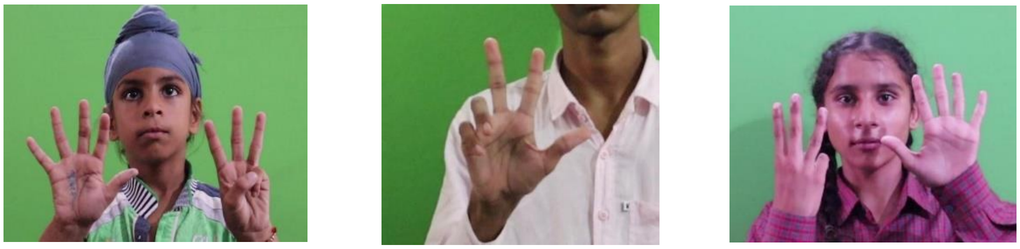


**Figure 2 - 3 different types of class 8 representations.**

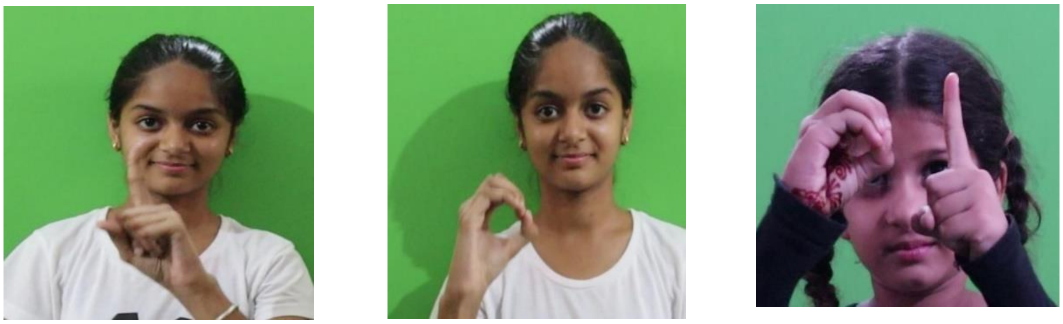


**Figure 3 - 3 different types of class 8 representation**

## 5.3 Data Preprocessing and landmarks Extraction

Data preprocessing refers to the process of cleaning, transforming, and organizing raw data into a format suitable for analysis and modeling. It is a critical step in the data analysis pipeline and involves several tasks to ensure that the data is accurate, consistent, and ready for further analysis.

Some common tasks involved in data preprocessing include:

1. **Data Cleaning**: This involves identifying and handling missing or erroneous data, such as removing duplicate records, filling in missing values, or correcting errors.
2. **Data Transformation**: This includes transforming the data into a more suitable format for analysis, such as converting categorical variables into numerical ones, scaling numerical features to a common range, or normalizing data distributions.
3. **Data Reduction**: This involves reducing the dimensionality of the dataset by selecting relevant features or using techniques like principal component analysis (PCA) to reduce the number of variables while preserving important information.
4. **Data Integration**: In cases where data is collected from multiple sources, data integration involves combining different datasets into a single, unified dataset for analysis.
5. **Data Discretization**: This involves transforming continuous variables into discrete categories, which can make the data easier to interpret and analyze

Sign language mostly performed by hands and in the Numbers dataset only hands have been used for accomplishing numbers in the videos. For better performance of the model I have only located and detected the hands landmarks in pairs of (x, y) values while other human body such as head, shoulders are ignored. Google MediaPipe is the model which I have used for the locating and detection of hands landmarks in the data preprocessing phase which I have explained as below.

## 5.4 MediaPipe

Mediapipe is an open-source framework developed by Google that facilitates the development of machine learning (ML) pipelines for various perceptual tasks, including object detection, face detection, hand tracking, pose estimation, and gesture recognition. It provides a comprehensive set of tools and pre-trained models to simplify the process of building real-time multimedia processing pipelines.

One of the key features of Mediapipe is its modular architecture, which allows developers to construct complex ML pipelines by connecting pre-built components called "graphs." These graphs encapsulate specific tasks or algorithms, such as image processing, feature extraction, or model inference, making it easy to assemble and customize pipelines for different applications.

Mediapipe provides a collection of pre-trained models optimized for real-time performance on different hardware platforms, including CPUs, GPUs, and mobile devices. These models are trained on large datasets and fine-tuned for specific tasks, ensuring high accuracy and efficiency in real-world scenarios.

Additionally, Mediapipe offers a set of developer tools and libraries for building custom ML models, training pipelines, and visualizing data. It supports various programming languages, including Python and C++, making it accessible to a wide range of developers and researchers.

Overall, Mediapipe simplifies the development of real-time perceptual computing applications by providing a flexible framework, pre-trained models, and developer tools for building and deploying ML pipelines efficiently.

The MediaPipe Hand Landmarker task lets you detect the landmarks of the hands in an image. You can use this task to locate key points of hands and render visual effects on them. This task operates on image data with a machine learning (ML) model as static data or a continuous stream and outputs hand landmarks in image coordinates, hand landmarks in world coordinates and handedness (left/right hand) of multiple detected hands.

The hand landmark model bundle detects the key point localization of 21 hand-knuckle coordinates within the detected hand regions. The model was trained on approximately 30K real-world images, as well as several rendered synthetic hand models imposed over various backgrounds.

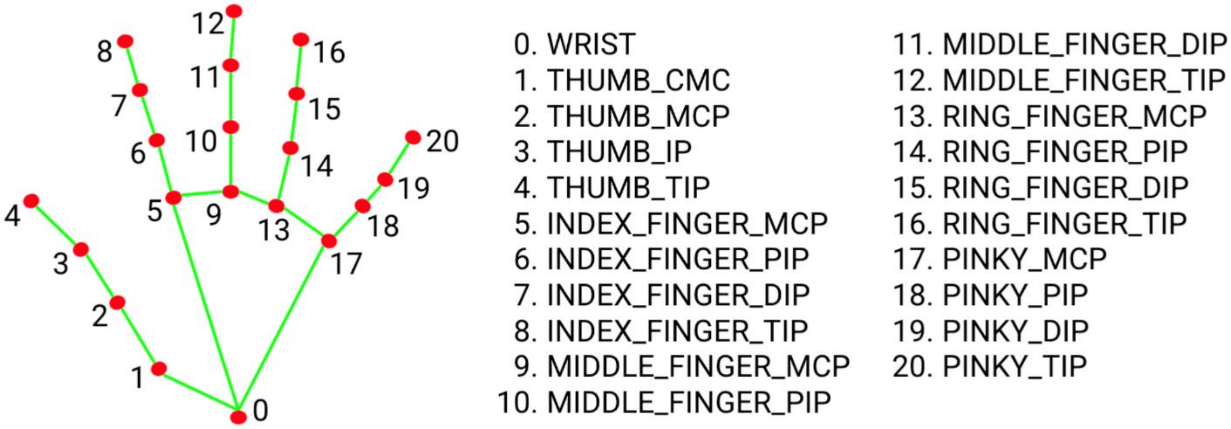


**Figure 4 – MediaPipe hand landmarks**

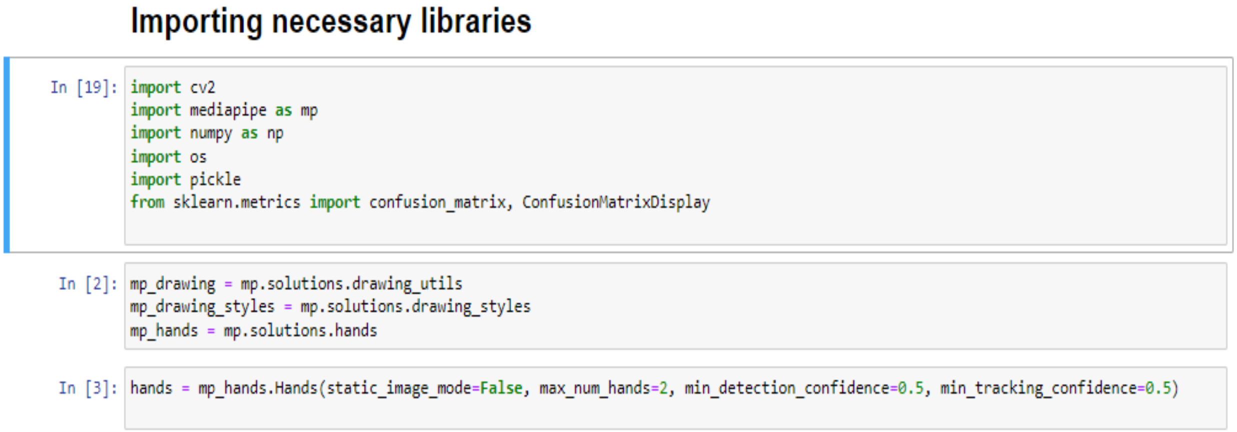
The hand landmarker model bundle contains a palm detection model and a hand landmarks detection model. The Palm detection model locates hands within the input image, and the hand landmarks detection model identifies specific hand landmarks on the cropped hand image defined by the palm detection model.

Since running the palm detection model is time consuming, when in video or live stream running mode, Hand Landmarker uses the bounding box defined by the hand landmarks model in one frame to localize the region of hands for subsequent frames. Hand Land marker only re-triggers the palm detection model if the hand landmarks model no longer identifies the presence of hands or fails to track the hands within the frame. This reduces the number of times Hand Land marker tiggers the palm detection model.

For data preprocessing and extraction of hands landmarks or key points I have followed the following two steps which are discussed as follow:

## 5.4.1 Detecting Frames include hands

The first step of data preprocessing is the recognizing of the frame which both or either of the hands are fully and completely present in that frame. For that, the below code goes through each video and MediaPipe will separate and store those frames in a separate directory.



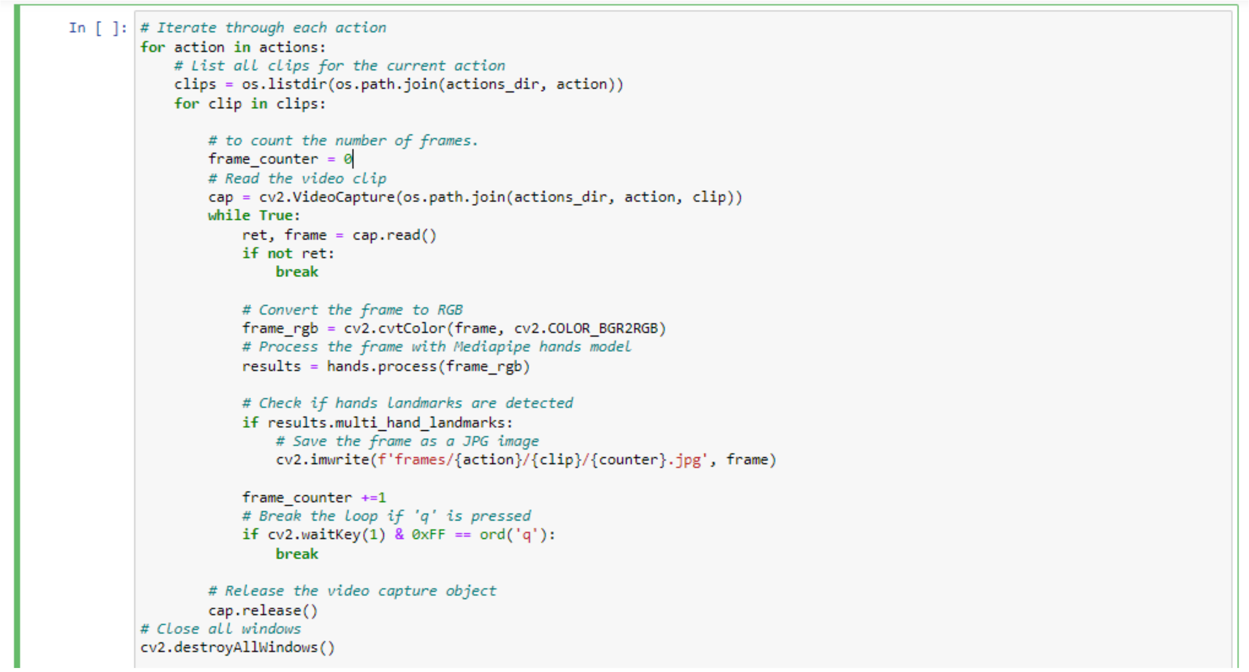
Importing necessary libraries

```
In [19]: import cv2
         import mediapipe as mp
         import numpy as np
         import os
         import pickle
         from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```
In [2]: mp_drawing = mp.solutions.drawing_utils
        mp_drawing_styles = mp.solutions.drawing_styles
        mp_hands = mp.solutions.hands
```

```
In [3]: hands = mp_hands.Hands(static_image_mode=False, max_num_hands=2, min_detection_confidence=0.5, min_tracking_confidence=0.5)
```

**Figure 5 – MediaPipe Hand Object Instantiation**

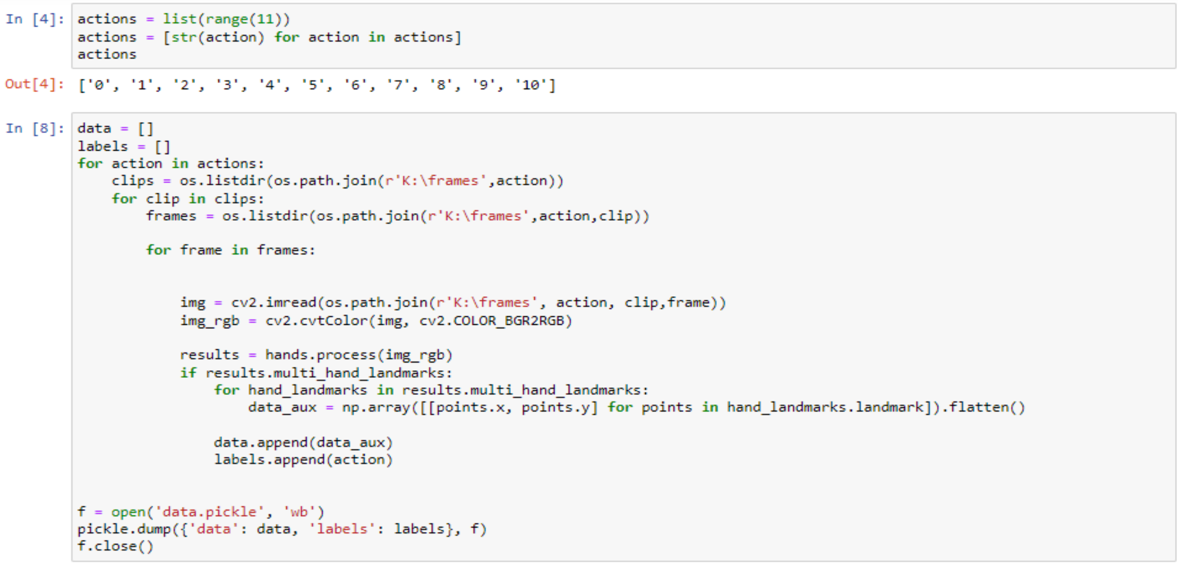


```
In [ ]: # Iterate through each action
        for action in actions:
            # List all clips for the current action
            clips = os.listdir(os.path.join(actions_dir, action))
            for clip in clips:

                # to count the number of frames.
                frame_counter = 0
                # Read the video clip
                cap = cv2.VideoCapture(os.path.join(actions_dir, action, clip))
                while True:
                    ret, frame = cap.read()
                    if not ret:
                        break

                    # Convert the frame to RGB
                    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                    # Process the frame with Mediapipe hands model
                    results = hands.process(frame_rgb)

                    # Check if hands landmarks are detected
                    if results.multi_hand_landmarks:
                        # Save the frame as a JPG image
                        cv2.imwrite(f'frames/{action}/{clip}/{counter}.jpg', frame)

                    frame_counter +=1
                    # Break the loop if 'q' is pressed
                    if cv2.waitKey(1) & 0xFF == ord('q'):
                        break

                # Release the video capture object
                cap.release()
        # Close all windows
        cv2.destroyAllWindows()
```

**Figure 6 – Code for detecting and separating frames with hand presence**

## 5.4.2 Hand Landmarks Extraction

Furthermore, after importing and instantiating the object, the following piece of code loops through each class and detects the hands landmarks in a NumPy array.



```
In [4]: actions = list(range(11))
        actions = [str(action) for action in actions]
        actions

Out[4]: ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10']

In [8]: data = []
        labels = []
        for action in actions:
            clips = os.listdir(os.path.join(r'K:\frames',action))
            for clip in clips:
                frames = os.listdir(os.path.join(r'K:\frames',action,clip))

                for frame in frames:


                    img = cv2.imread(os.path.join(r'K:\frames', action, clip,frame))
                    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

                    results = hands.process(img_rgb)
                    if results.multi_hand_landmarks:
                        for hand_landmarks in results.multi_hand_landmarks:
                            data_aux = np.array([[points.x, points.y] for points in hand_landmarks.landmark]).flatten()

                        data.append(data_aux)
                        labels.append(action)


        f = open('data.pickle', 'wb')
        pickle.dump({'data': data, 'labels': labels}, f)
        f.close()
```

**Figure 7 – Code for extraction of Hands Landmarks**

PEPE JEANS

TRACK
1984
SUPERDRY
DEPT
&FIELD

# 5.5 Machine Learning Algorithms

Machine learning algorithms are the core techniques used to build models that can learn from data and make predictions or decisions. There are various types of machine learning algorithms, each designed for different tasks and scenarios. Here are some of the commonly used machine learning algorithms:

1. **Supervised Learning Algorithms**:

- **Linear Regression**: Linear regression is a fundamental statistical technique used to model the linear relationship between a dependent variable and one or more independent variables. It is widely employed in various fields, including economics, finance, engineering, and social sciences, to analyze and make predictions based on data. The goal of linear regression is to find the best-fitting straight line that describes the relationship between the dependent variable (often denoted as y) and the independent variable(s) (often denoted as x)
- **Logistic Regression**: Logistic regression is a statistical technique used for binary classification problems, where the goal is to predict the probability of an instance belonging to one of two classes (e.g., yes/no, true/false, spam/not spam). It is a type of regression analysis that models the relationship between one or more independent variables and a binary dependent variable. In logistic regression, the dependent variable is a categorical variable that can take only two values, typically coded as 0 and 1. The independent variables can be continuous or categorical. The logistic regression model uses the logistic function (also known as the sigmoid function) to model the probability of the binary outcome.

- **Decision Trees**: Decision trees are a type of machine learning algorithm used for both classification and regression tasks. They are tree-like models where each internal node represents a decision or test on an attribute, each branch represents the outcome of that test, and each leaf node represents a class label (for classification) or a numerical value (for regression). The construction of a decision tree is based on recursively partitioning the data into smaller subsets, where at each node, the algorithm selects the attribute that best separates the classes or minimizes the variance in the target variable. This process continues until a stopping criterion is met, such as reaching a maximum depth or having a node with all instances belonging to the same class.

- **Support Vector Machines (SVMs)**: Support Vector Machines (SVMs) are powerful supervised machine learning algorithms used for classification and regression analysis. They are particularly well-suited for high-dimensional data and can effectively handle non-linear problems. The main idea behind SVMs is to find the optimal hyperplane that separates the classes with the maximum margin. The margin is the distance between the hyperplane and the closest data points from each class, known as support vectors. In the case of linearly separable data, the SVM finds the hyperplane that maximizes the margin between the two classes. For non-linear problems, SVMs use a technique called the kernel trick, which maps the data into a higher-dimensional feature space where it becomes linearly separable.

- **Naive Bayes**: Naive Bayes is a simple yet powerful classification algorithm based on the Bayes' theorem and the assumption of independence between features. It is called "naive" because it assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable.

- **K-Nearest Neighbors (KNN)**: K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm used for both classification and regression tasks. It is a type of lazy learning algorithm, meaning that it does not build a model until a new instance needs to be predicted. The core idea behind KNN is to classify a new instance based on the majority vote of its k nearest neighbors in the feature space.

- **Ensemble Methods**: Combine multiple models to improve accuracy and robustness, such as Random Forests, Gradient Boosting Machines, and AdaBoost.

2. **Unsupervised Learning Algorithms:**

- **K-Means Clustering**: K-Means Clustering is one of the most popular and widely used unsupervised machine learning algorithms for cluster analysis. It is a centroid-based clustering technique that aims to partition a dataset into K distinct non-overlapping clusters, where each data point belongs to the cluster with the nearest mean (centroid).
- **Hierarchical Clustering**: Hierarchical clustering is a type of unsupervised machine learning algorithm that builds a hierarchy of clusters, either by merging smaller clusters into larger ones (agglomerative) or by dividing larger clusters into smaller ones (divisive). The result is a tree-like structure called a dendrogram, which represents the relationships between the clusters.
- **Principal Component Analysis (PCA)**: Principal Component Analysis (PCA) is a widely used dimensionality reduction technique in machine learning and data analysis. It is an unsupervised linear transformation method that aims to find a lower-dimensional representation of the data while preserving as much of the original variance as possible. The key idea behind PCA is to project the high-dimensional data onto a new coordinate system, where the axes (known as principal components) are ordered by the amount of variance they capture in the data. The first principal component captures the maximum possible variance, the second principal component captures the second-most variance, and so on.
- **t-SNE (t-Distributed Stochastic Neighbor Embedding)**: t-SNE (t-Distributed Stochastic Neighbor Embedding) is a non-linear dimensionality reduction technique that is particularly well-suited for visualizing high-dimensional data in a low-dimensional space, typically 2D or 3D. It is a powerful tool for exploring and understanding the underlying structure and patterns within complex datasets. The main idea behind t-SNE is to model the similarities between data points in the high-dimensional space and then preserve these similarities as faithfully as possible in the low-dimensional embedding. This is achieved by minimizing the divergence between two probability distributions: one that measures

pairwise similarities in the original high-dimensional space, and another that measures pairwise similarities in the low-dimensional embedding space.

- **Association Rule Mining**: Association rule mining is a popular unsupervised machine learning technique used to discover interesting relationships and patterns within large datasets. It is particularly useful for analyzing transaction data, such as market basket analysis in retail environments, but it can also be applied to other types of data. The goal of association rule mining is to identify frequent itemsets and derive association rules that describe the relationships between items in a dataset. An association rule is an expression of the form "If X, then Y," where X and Y are disjoint itemsets. The rule expresses the probability that a transaction containing X will also contain Y.

3. **Deep Learning Algorithms**:

1. **Artificial Neural Networks (ANNs)**: Artificial Neural Networks (ANNs) are a fundamental concept in the field of artificial intelligence and machine learning, inspired by the structure and function of the human brain. ANNs are composed of interconnected nodes, often referred to as neurons or units, organized in layers. The basic building block of an ANN is the perceptron, which takes multiple input signals, applies weights to them, sums them up, and then passes the result through an activation function to produce an output. Multiple perceptrons can be organized into layers, with each layer typically performing a different transformation of the input data.

2. **Convolutional Neural Networks (CNNs)**: Convolutional Neural Networks (CNNs) are a specialized type of artificial neural network designed specifically for processing structured grid data, such as images and videos. They are particularly powerful for tasks involving visual recognition, including image classification, object detection, segmentation, and more. CNNs have revolutionized computer vision and have become the backbone of many state-of-the-art algorithms in this field.

3. **Recurrent Neural Networks (RNNs)**: Recurrent Neural Networks (RNNs) are a class of artificial neural networks designed for sequential data processing, where the order of data points matters. Unlike feedforward neural networks, which process each input independently, RNNs maintain internal memory to process sequences of inputs while retaining information

about previous inputs. This makes them well-suited for tasks involving time series data, natural language processing, speech recognition, and more.

4. **Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU)**: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are specialized variants of recurrent neural networks (RNNs) designed to address the vanishing gradient problem and better capture long-range dependencies in sequential data.

   - **Long Short-Term Memory (LSTM)**: LSTM networks introduce a memory cell with three gates: the input gate, the forget gate, and the output gate. These gates regulate the flow of information into and out of the memory cell, allowing LSTMs to selectively remember or forget information over long sequences. The input gate controls the flow of new information into the cell, the forget gate controls which information to discard from the cell's memory, and the output gate controls the flow of information from the cell to the rest of the network. By dynamically updating their internal state, LSTMs can effectively capture long-term dependencies in sequential data and avoid the vanishing gradient problem.

   - **Gated Recurrent Unit (GRU)**: GRUs are a simpler variant of LSTMs with two gates: the update gate and the reset gate. GRUs combine the input and forget gates into a single update gate, which controls the flow of new information into the cell and the flow of information from the previous hidden state. The reset gate controls how much information from the past should be forgotten when computing the new candidate activation. GRUs have fewer parameters than LSTMs, making them computationally more efficient and easier to train in some cases. Despite their simpler architecture, GRUs have been shown to perform comparably to LSTMs in many tasks.

5. **Autoencoders**: Autoencoders are a type of artificial neural network used for unsupervised learning of efficient data representations. They are composed of an encoder and a decoder, which work together to learn a compressed representation (encoding) of the input data and then reconstruct the original input from this representation. The basic architecture of an autoencoder consists of three main components:

   - **Encoder**: The encoder takes the input data and maps it to a lower-dimensional latent space representation. This is achieved through a series of hidden layers that gradually reduce the dimensionality of the input data. The encoder's output represents a compressed version of the input data, capturing its essential features.

- **Decoder**: The decoder takes the compressed representation produced by the encoder and reconstructs the original input data. Similar to the encoder, the decoder consists of a series of hidden layers that gradually increase the dimensionality of the encoded representation until the original input dimensions are restored. The decoder's output should closely resemble the input data.

4. **Generative Adversarial Networks (GANs)**: Generative Adversarial Networks (GANs) are a class of artificial intelligence algorithms used in unsupervised machine learning, capable of generating new data samples from a given distribution. GANs consist of two neural networks, the generator and the discriminator, which are trained simultaneously through a competitive process. Here's how GANs work:

    - **Generator**: The generator takes random noise as input and generates synthetic data samples. Initially, the generator produces random noise, but as training progresses, it learns to generate data samples that resemble the real data distribution. The goal of the generator is to produce data samples that are indistinguishable from real data.

    - **Discriminator**: The discriminator, on the other hand, acts as a binary classifier that distinguishes between real and fake data samples. It is trained on both real data samples and synthetic samples generated by the generator. The discriminator's objective is to correctly classify real data as real and fake data as fake.

5. **Reinforcement Learning Algorithms**:

- **Q-Learning**: Q-Learning is a model-free reinforcement learning algorithm used to learn optimal actions to take in a given environment. It's particularly useful in situations where the agent doesn't have prior knowledge of the environment's dynamics and must learn through trial and error.

- **Deep Q-Networks (DQN)**: Deep Q-Networks (DQN) is a reinforcement learning algorithm that combines Q-Learning with deep neural networks, enabling it to handle high-dimensional state spaces and learn complex policies. DQN was introduced by DeepMind in 2013 and has since become one of the foundational algorithms in the field of deep reinforcement learning.

- **Policy Gradients**: Policy Gradient methods are a class of reinforcement learning algorithms used to directly optimize the policy function, which determines the agent's behavior in an environment. Unlike value-based methods such as Q-learning, which aim to learn the value function (e.g., Q-values) and then derive the policy from it, policy gradient methods directly parameterize the policy and update its parameters to maximize the expected cumulative reward.
- **Actor-Critic Methods**: Actor-Critic methods are a class of reinforcement learning algorithms that combine elements of both value-based and policy-based approaches. They consist of two main components: an actor, which learns the policy, and a critic, which learns the value function.

Since the random forest algorithm has been used to build the model for detection of sign language activity such as numbers from zero to ten. It would be helpful to describe it in more details.

# 5.6 What is random forest?

Random forest, a widely-utilized machine learning algorithm, is trademarked by Leo Breiman and Adele Cutler. This algorithm aggregates the outputs of multiple decision trees to produce a single result. Renowned for its simplicity and versatility, it proficiently addresses both classification and regression tasks, contributing to its widespread adoption across various domains.

## 5.6.1 Decision trees

Decision trees serve as the fundamental building blocks within the random forest model, offering insight into its operation. Essentially, a decision tree begins with a primary question, such as "Should I surf?" This question serves as the root node, leading to subsequent nodes that pose further inquiries, like "Is it a long period swell?" or "Is the wind blowing offshore?" These decision nodes facilitate data segmentation, guiding individuals toward a final verdict represented by the leaf node. Instances meeting the criteria follow the "Yes" branch, while others diverge. Decision trees strive to identify optimal splits for data subsets, often employing algorithms like

Classification and Regression Tree (CART) and metrics such as Gini impurity or information gain for evaluation.

While decision trees are susceptible to issues like bias and overfitting, the random forest algorithm mitigates these concerns by assembling multiple uncorrelated decision trees. This ensemble approach ensures more accurate predictions, making random forest particularly effective in classification problems, where class labels like "surf" and "don't surf" are determined

### 5.6.2 Ensemble methods

Ensemble learning techniques involve utilizing a collection of classifiers, such as decision trees, and combining their predictions to determine the most prevalent outcome. The two most prominent ensemble methods are bagging, also known as bootstrap aggregation, and boosting. Introduced by Leo Breiman in 1996, the bagging method entails selecting random samples of data from a training set with replacement, allowing individual data points to be chosen multiple times. Subsequently, these models are trained independently, and the average or majority of their predictions, depending on the task type—regression or classification—results in a more accurate estimation. This strategy is frequently employed to mitigate variance within noisy datasets.

## Random forest algorithm

The random forest algorithm extends the bagging method by incorporating both bagging and feature randomness, leading to the creation of an uncorrelated forest of decision trees. Feature randomness, also referred to as feature bagging or "the random subspace method," involves generating a random subset of features, thereby ensuring minimal correlation among decision trees. This distinction is pivotal compared to traditional decision trees, where all possible feature splits are considered. In contrast, random forests only select a subset of features, enhancing their diversity and robustness.

Returning to the "should I surf?" analogy, the questions used to make predictions may vary among individuals. By accounting for the potential variability in the data through feature randomness, random forests mitigate the risks of overfitting, bias, and overall variance, resulting in more accurate predictions.

# How it works

Random forest algorithms operate based on three primary hyperparameters that must be configured prior to training: node size, the number of trees, and the number of features sampled. With these parameters set, the random forest classifier can be applied to address regression or classification challenges.

The random forest algorithm comprises a collection of decision trees, with each tree in the ensemble constructed from a data sample drawn from the training set using replacement, known as the bootstrap sample. Within this training sample, one-third is reserved as test data, referred to as the out-of-bag (oob) sample. Another layer of randomness is introduced through feature bagging, which diversifies the dataset and diminishes correlation among decision trees.

The prediction determination varies depending on the problem type. In regression tasks, the outputs of individual decision trees are averaged, while in classification tasks, a majority vote, selecting the most frequent categorical variable, determines the predicted class. Subsequently, the oob sample is utilized for cross-validation, ultimately refining the prediction.

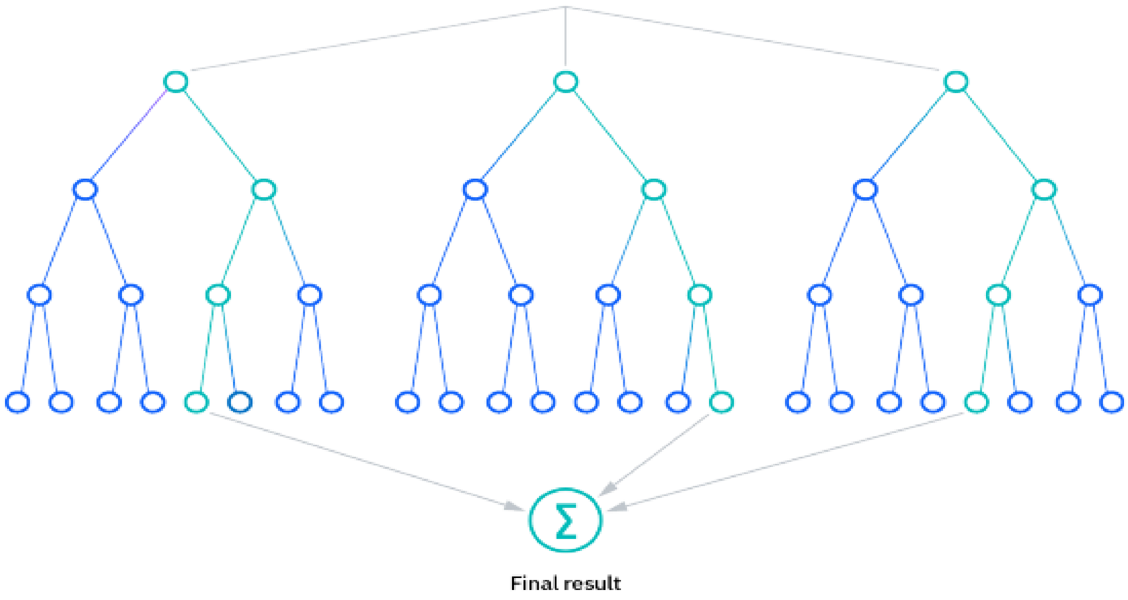
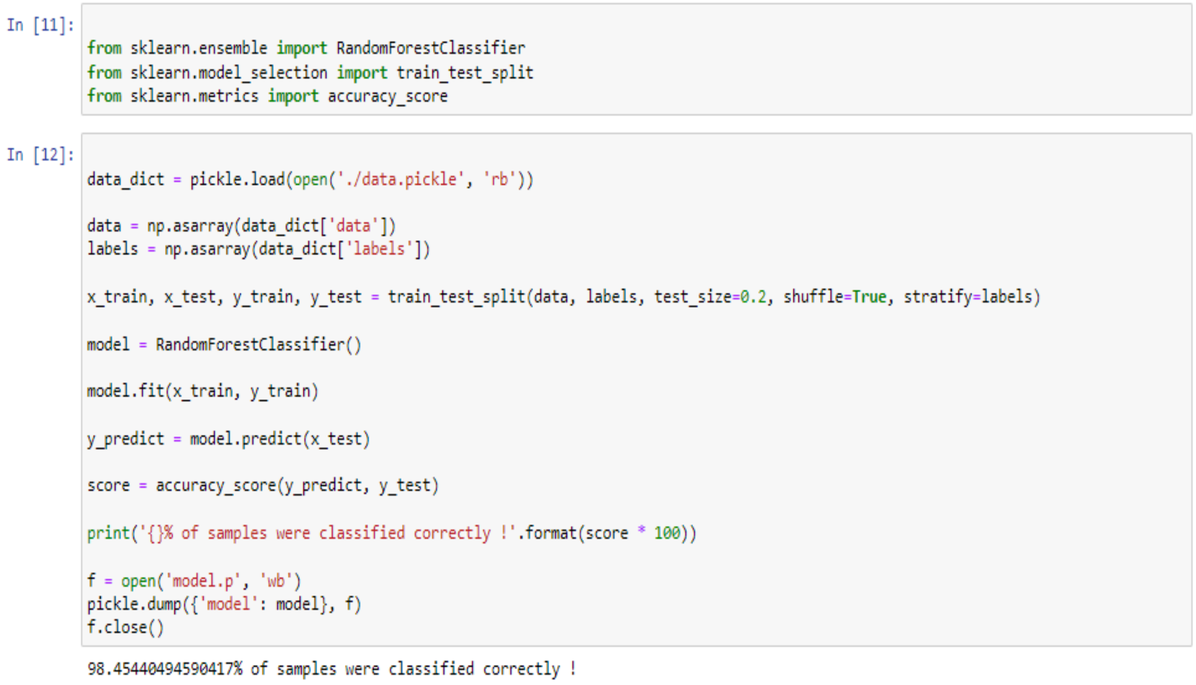


**Figure 8 – Random Forest Algorithm**



```
In [11]:
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
In [12]:
data_dict = pickle.load(open('./data.pickle', 'rb'))

data = np.asarray(data_dict['data'])
labels = np.asarray(data_dict['labels'])

x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, shuffle=True, stratify=labels)

model = RandomForestClassifier()

model.fit(x_train, y_train)

y_predict = model.predict(x_test)

score = accuracy_score(y_predict, y_test)

print('{}% of samples were classified correctly !'.format(score * 100))

f = open('model.p', 'wb')
pickle.dump({'model': model}, f)
f.close()

98.45440494590417% of samples were classified correctly !
```

**Figure 9 - Code for training random forest classifier**

The random forest classifier model was trained on 220 videos. Moreover, 30 frames were selected from each video. Each frame contains 21 landmark points for each hand (42 for both hands). Thus, the dataset fed to model was of shape (6468, 42). Furthermore, the model classifier accuracy is 98%.

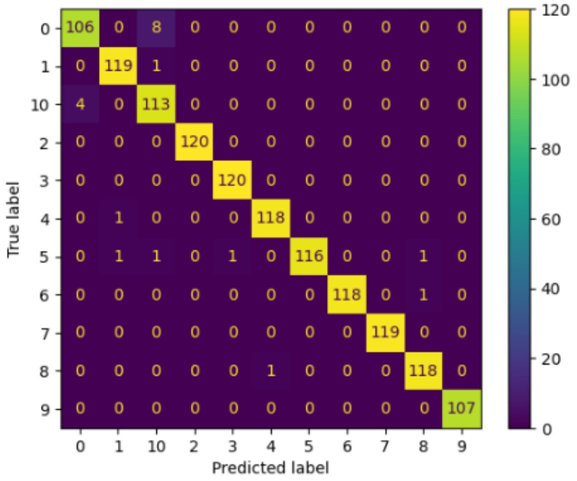
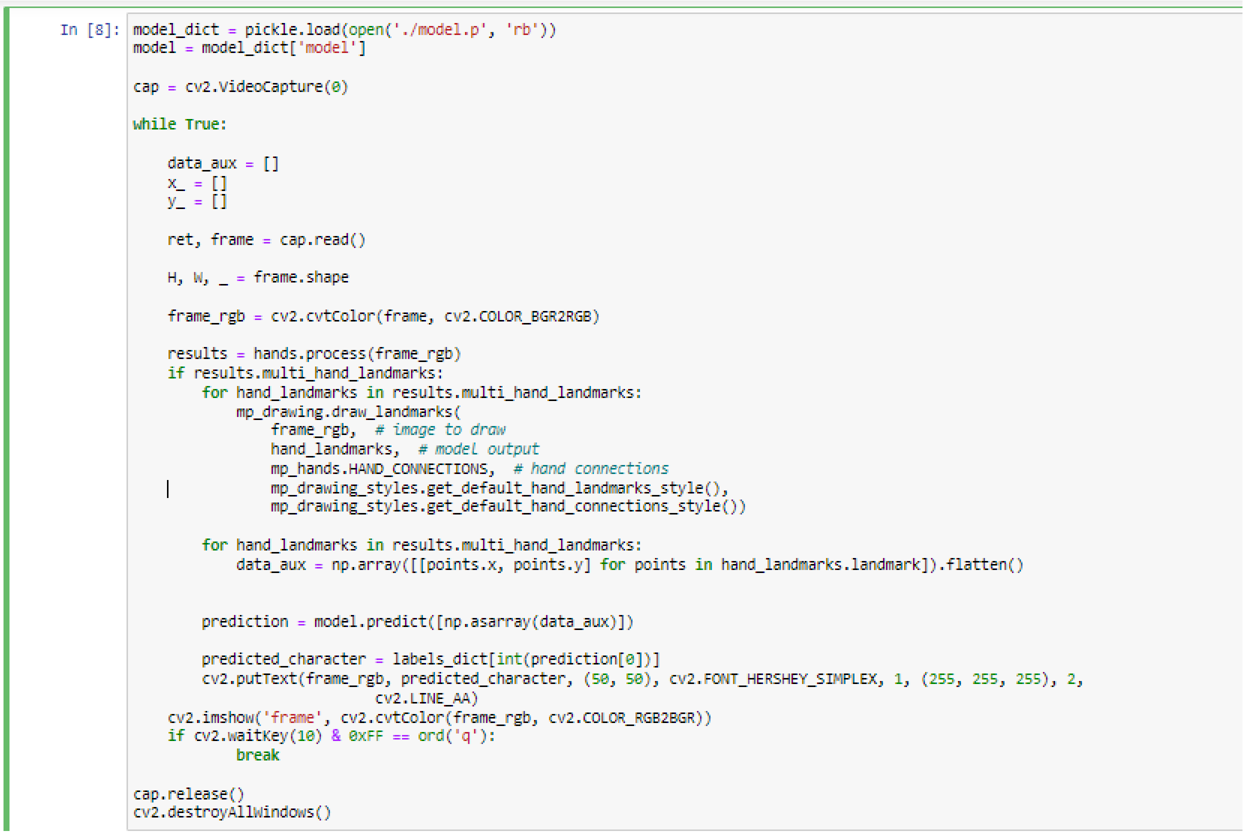


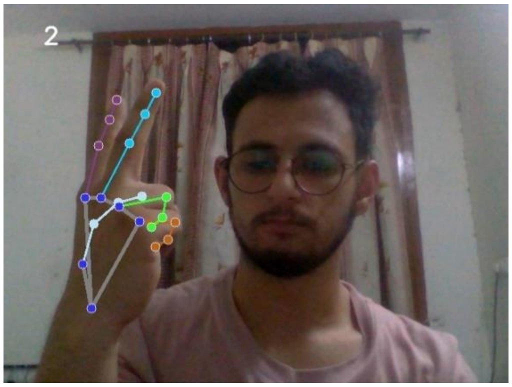
**Figure 10 – Confusion Matrix**

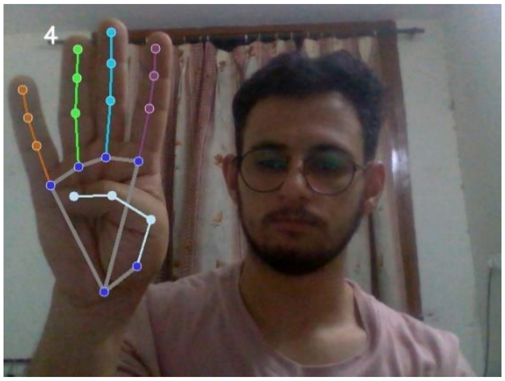
## 5.7 Testing on real live webcam video

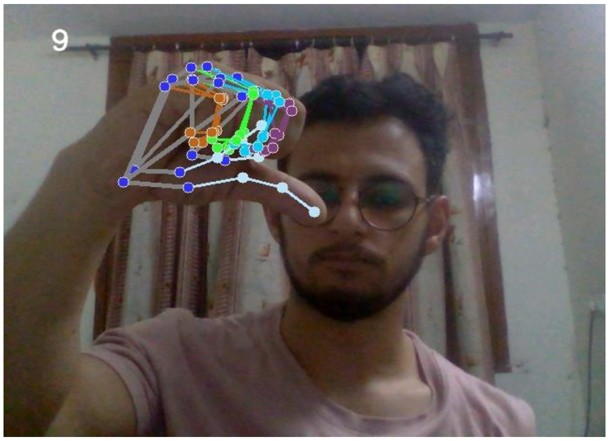


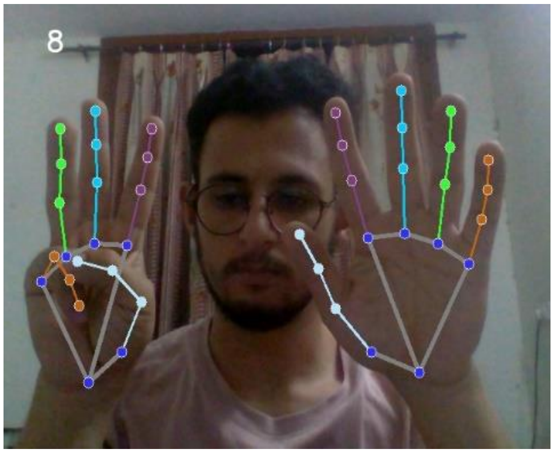
```
In [8]: model_dict = pickle.load(open('./model.p', 'rb'))
        model = model_dict['model']

        cap = cv2.VideoCapture(0)

        while True:

            data_aux = []
            x_ = []
            y_ = []

            ret, frame = cap.read()

            H, W, _ = frame.shape

            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

            results = hands.process(frame_rgb)
            if results.multi_hand_landmarks:
                for hand_landmarks in results.multi_hand_landmarks:
                    mp_drawing.draw_landmarks(
                        frame_rgb,  # image to draw
                        hand_landmarks,  # model output
                        mp_hands.HAND_CONNECTIONS,  # hand connections
                        mp_drawing_styles.get_default_hand_landmarks_style(),
                        mp_drawing_styles.get_default_hand_connections_style())

                for hand_landmarks in results.multi_hand_landmarks:
                    data_aux = np.array([[points.x, points.y] for points in hand_landmarks.landmark]).flatten()


                prediction = model.predict([np.asarray(data_aux)])

                predicted_character = labels_dict[int(prediction[0])]
                cv2.putText(frame_rgb, predicted_character, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2,
                            cv2.LINE_AA)
            cv2.imshow('frame', cv2.cvtColor(frame_rgb, cv2.COLOR_RGB2BGR))
            if cv2.waitKey(10) & 0xFF == ord('q'):
                    break

        cap.release()
        cv2.destroyAllWindows()
```

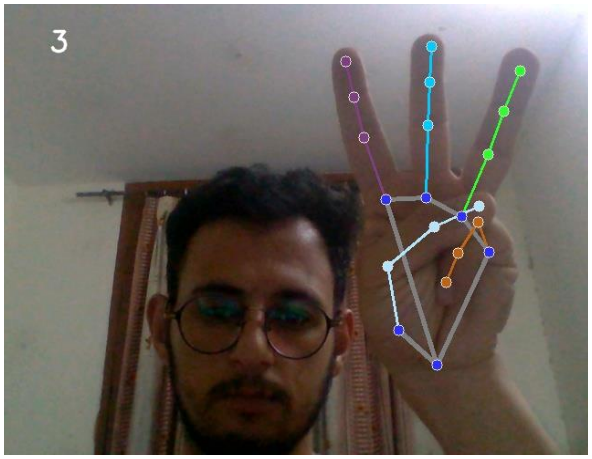
**Figure 11 – Code for testing model on live webcam**

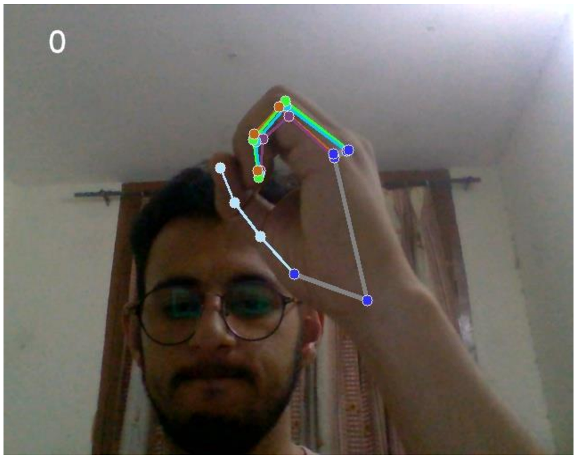


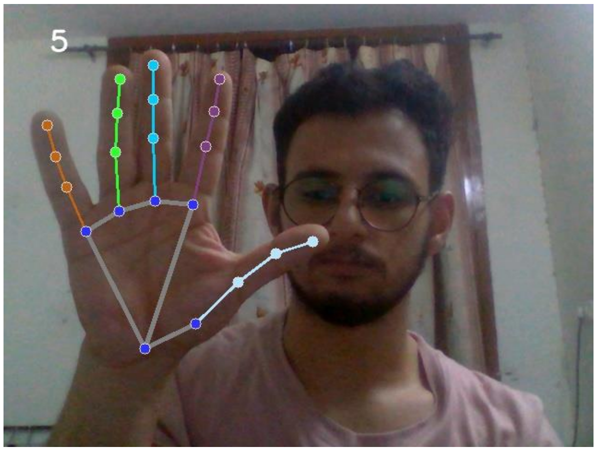


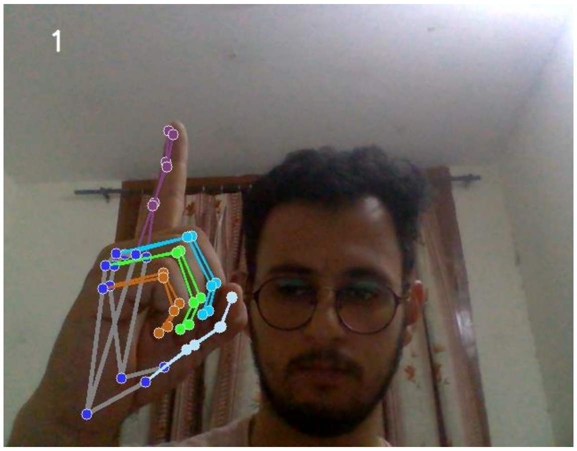

9


8


3


0


5


1

# 6 Discussion & Results

- This research focused on detecting numerical digits in Indian Sign Language (ISL) using machine learning techniques.
- The dataset utilized consisted exclusively of 220 video samples representing digits from 0 to 10. Each digit class initially had 20 video samples, though imbalances were introduced due to multiple distinct representations for some classes, specifically digits 6, 8, 9, and 10.
- Approximately 50% of the digit classes (6, 8, 9, and 10) exhibited more than one sign language representation. Furthermore, the distribution of videos across these distinct representations was uneven, creating challenges for accurate classification and contributing to imbalances in the dataset.
- Preprocessing was conducted using Google's Media Pipe, an open-source framework for extracting hand landmark key points.
- Each frame of the video dataset was processed to extract hand landmark features, which were then converted into NumPy arrays for input into the machine learning model.
- A Random Forest classifier was employed for digit recognition, achieving an overall classification accuracy of 98%.
- Despite this high accuracy, the model occasionally struggled to differentiate between digits with similar or multiple distinct representations (e.g., 6, 8, 9, and 10). This was largely due to the limited and imbalanced representation within the training dataset.
- The trained model has the potential for real-time application, as it can be integrated with live webcam feeds for instant digit recognition in ISL.
- By adding more training data or augmenting the data, the model will become stronger and more robust in prediction of videos, specially those that the model is weak.

# 7 Future Scope

The current implementation of the machine learning model for recognizing Indian Sign Language (ISL) digits demonstrates promising results. However, there are several opportunities for further research and development to broaden its scope, enhance its functionality, and expand its applications:

1. **Incorporate Full ISL Dataset**: The present work focuses exclusively on digit recognition. Future research can extend the model to recognize a wider range of ISL gestures, including alphabets, nouns, verbs, and adjectives. This would pave the way for a comprehensive system capable of supporting full-fledged ISL communication.

2. **Address Imbalanced Datasets**: To improve the model's accuracy for digit classes with multiple representations (e.g., 6, 8, 9, and 10), future work should focus on addressing dataset imbalances. Techniques such as data augmentation, synthetic data generation, and targeted sampling can enhance the model's robustness in handling complex and varied representations.

3. **Develop Real-Time Applications**: Building on the current model, a real-time digit recognition system could be developed for integration with live video feeds. This would enable applications such as live communication aids for hearing-impaired individuals and real-time translation of ISL digits into text or spoken language.

4. **Generalization Across Backgrounds and Environments**: Training the model on videos with diverse backgrounds and lighting conditions will improve its performance in real-world scenarios. This enhancement would make the system more versatile and reliable outside controlled environments.

5. **Integration with Natural Language Processing (NLP):** Combining the digit recognition model with NLP systems could enable two-way communication by converting ISL gestures into text or speech and vice versa. This integration would significantly improve accessibility for deaf and hard-of-hearing individuals.

6. **Support for Multiple Sign Languages**: Expanding the system to recognize and process gestures from other sign languages (e.g., American Sign Language, British Sign Language)

would promote inclusivity and extend its global reach, catering to diverse linguistic communities.

7. **Augmented and Virtual Reality Applications**: Incorporating augmented reality (AR) and virtual reality (VR) could open new avenues for immersive learning and assistive technologies. AR systems could overlay real-time ISL translations in physical environments, while VR environments could enable virtual sign language practice and communication.

8. **Personalized Sign Language Models**: Developing personalized models that adapt to individual users' unique gesture styles and preferences could lead to more natural interactions and higher user satisfaction. This would be especially useful in healthcare or education settings.

9. **Healthcare and Assistive Technologies**: The system could be adapted for use in healthcare settings, enabling communication with patients who use ISL. It could also be integrated into assistive devices or smart home systems to enhance accessibility and independence for individuals with disabilities.

10. **Collaboration Across Disciplines**: Interdisciplinary collaboration among researchers in machine learning, linguistics, education, and healthcare will be crucial to drive innovation. Partnerships with ISL experts and the deaf community will ensure the system's effectiveness and cultural relevance

The future scope of this project is vast and holds the potential to revolutionize the way we communicate and understand sign language. Interdisciplinary collaborations among researchers, engineers, and domain experts will be essential to realize the full potential of this technology and make it accessible and beneficial to society.

# 8 Conclusion

This thesis has explored the application of machine learning techniques for the detection of Indian Sign Language (ISL) digits, focusing on the Numbers subset of the dataset. The dataset comprised 220 videos representing digits from 0 to 10, with each digit class containing 20 videos. However, some classes, such as 6, 8, 9, and 10, featured multiple distinct sign representations, leading to an imbalanced dataset.

Using the Google Media Pipe framework for preprocessing and the Random Forest Classifier for digit classification, the proposed system achieved an impressive accuracy of 98%. Despite this success, challenges were observed in accurately predicting certain digits due to their similar representations and the dataset's imbalanced nature. The system's robustness was demonstrated by its ability to classify ISL digits in live video streams, regardless of the background, which was green during training. This highlights the model's potential for real-world applications, such as live ISL digit detection for communication aids.

While this research primarily focused on the Numbers subset, future efforts could extend the model to encompass other ISL gesture categories, such as alphabets and verbs, enabling a comprehensive solution for ISL recognition. Advanced deep learning techniques, data augmentation, and larger datasets could further improve the system's performance, especially for digits with multiple representations. Potential applications of this system include real-time ISL translation, integration with natural language processing (NLP) for seamless communication, and expansion to support other sign languages globally. Additionally, future research could explore integration with augmented reality (AR) and virtual reality (VR) technologies, creating immersive and educational environments for ISL learning and communication.

The findings of this thesis underscore the potential of machine learning to address challenges in sign language recognition, promoting accessibility and inclusivity for individuals reliant on ISL. By laying the groundwork for more sophisticated and comprehensive systems, this research contributes to the broader goal of fostering effective communication and accessibility for the deaf and hard-of-hearing communities.

# References

[1] Abdulhamit Subasi, Kholoud Khateeb, Tayeb Brahimi and Akila Sarirete, “Human activity recognition using machine learning methods in a smart healthcare environment” Version of Record 15, Nov 2019.

[2] Homay Danaei Mehr and Husyein Polat, “Human Activity Recognition in Smart Home with Deep Learning Approach”, 2019.

[3] Sungyoung Lee, Jin Wang, Zhongqi Zhang, Li Bin and “An Enhanced Fall Detection System for Elderly Person Monitoring using Consumer Home Networks “IEEE Transactions on Consumer Electronics 60(1):23-29 DOI:10.1109/TCE.2014.6780921, Feb 2014

[4] Shian-Ru Ke, Hoang Le Uyen Thuc, Young-Jin Lee, Jenq-Neng Hwang, Jang-Hee Yoo, Kyoung-Ho Choi ,“A Review on Video-Based Human Activity Recognition” Computers 2013, 2(2), 88-131; https://doi.org/10.3390/computers2020088 Jun 2013

[5] Djamila Romaissa Beddiar, Brahim Nini, Mohammad Sabokrou and Abdenour Hadid “Vision-based human activity recognition: a survey” 79, pages30509–30555 (2020), Aug 2015

[6] Salvatore Gaglio, Giuseppe Lo Re and Marco Morana “Human Activity Recognition Process Using 3-D Posture Data” IEEE Transactions on Human-Machine Systems ( Volume: 45, Issue: 5) Oct 2015

[7] Ong Chin Ann and Bee Theng Lau “Human activity recognition: A review” DOI: 10.1109/ICCSCE.2014.7072750 Conference: Proceedings - 4th IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2014, Mar 2015

[8] Ling Chen, Xiaoze Liu, Liangying Peng and Menghan Wu “Deep learning based multimodal complex human activity recognition using wearable devices” 51, pages4029– 4042 Nov 2020

[9] Shugang Zhang, Zhiqiang Wei, Jie Nie, Lei Huang, Shuang Wang and Zhen Li “A Review on Human Activity Recognition Using Vision-Based Method” Volume 2017 Article ID 3090343 , Aug 2017

[10] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, Cordelia Schmid “ViViT: A Video Vision Transformer” Nov 2021.

[11] Liandro B. Marinho, A.H. de souza junior and P.P.Reboucas Filho, “A New Approach to Human Activity Recognition Using Machine Learning Techniques” AISC,volume 557, Feb 2017

Neil Robertson and Ian Reid, “A general method for human activity recognition in video” Sep 2006.

[12] S.U.Park, J.H.Park, M.A.Al-Masni, M.A.Al-Antari, Md.Z.Uddin, T.S.Kim “A Depth Camera-based Human Activity Recognition via Deep Learning Recurrent Neural Network for Health and Social Care Services” Oct 2016

[13] Adrian Nunez-Marcos, Gorka Azkune and Ignacio Arganda-Carreras “Vision-Based Fall Detection with Convolutional Neural Networks” Wireless Communications and Mobile Computing 2017(1):1-16 DOI:10.1155/2017/9474806, Dec 2017

[14] Yang Wang and Greg Mori “Human Action Recognition by Semi Latent Topic Models” IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 31, Issue: 10), Oct 2009