Machine Learning Project 2022-23

Hand Gesture Recognition System

Sem V Report Machine Learning
Of

Bachelor of Technology

In

Computer Science and Engineering (Data Science)

By
L002 Om Agrawal
L003 Vanshaj Ajmera
L027 Shardul Patki
L039 Soham Vasudeo
L040 Ritika Shetty

Under the supervision of

Dr. Shweta Loonkar

(Prof MPSTME NMIMS)

SVKM's NMIMS University
(Deemed-to-be University)

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING
Vile Parle (W), Mumbai-56
2022-23

CERTIFICATE

This is to certify that the project entitled "Hang GEsture Recognituion System", has been done by Mr, Om Agrawal, Vanshaj Ajmera, Shardul Patki, Soham Vasudeo and Ms. Ritika Shetty under my guidance and supervision for Machine Learning Project as part of curriculum of Bachelor of Technology in Computer Science Engineering (Data Science) of MPSTME, SVKM's NMIMS (Deemed-to-be University), Mumbai, India.

Name and signature of Mentor

_____

Dr. Shweta Loonkar

Date of Submission: 28/10/2022

Place: Mumbai

Abstract

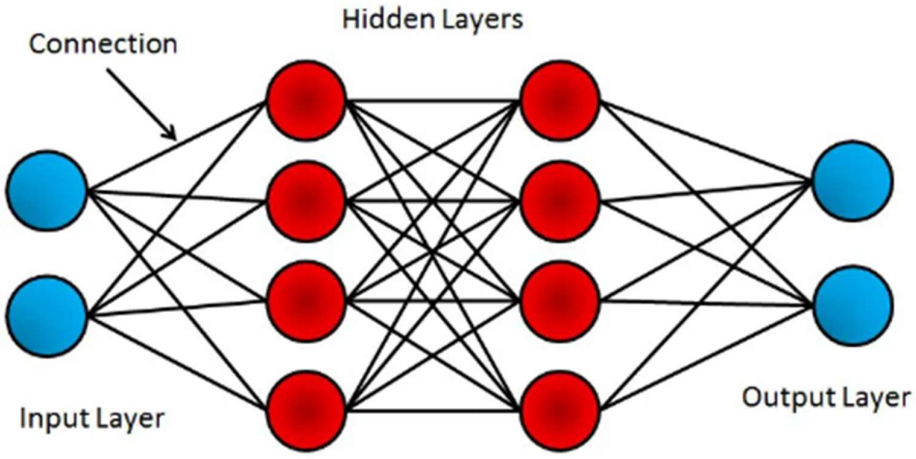
Man-machine interfaces based on hand gesture recognition have seen a lot of development recently. The majority of visual hand gesture recognition systems only function in confined environments because of the impact of lighting and complicated backgrounds. To detect skin colour regions like hands, an adaptive skin colour model based on face detection is used. Media pipe developed by Google was one of the key libraries at the heart of this project. We created a quick and easy motion history image-based approach to categorise dynamic hand movements. The classifiers for the hello, iloveyou, thanks, one and victory hand motions were trained on 30 frames each for each of the 30 samples used of training in each of these gestures. There were a total of 5 hand motions specified, including fist and wave movements. Most home appliances can be controlled using it, generally speaking. According to experimental findings, the processing time is 3.81 ms per frame, and the accuracy is an average of 96.4%. These illustrated the suggested system's viability. Using three primary modules—the camera and segmentation module, the detection module, and the feature extraction module—this study provides an overview of various approaches to hand gesture recognition. Depending on their benefits, many strategies can be utilised to obtain the desired goals. This publication also provides a summary of prior research, hand gesture method outcomes, and a comparison of gesture recognition techniques.

Introduction

Machine Learning, as the name implies, is all about computers learning on their own without being explicitly programmed or without direct human participation. This machine learning method begins with feeding them high-quality data, followed by training the computers by constructing various machine learning models. In terms of a formal definition of Machine Learning, we can state that a Machine Learning algorithm learns from experience E with respect to some type of task T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.

e.

Artificial neural networks (ANNs) are computer systems that handle a variety of tasks using vast amounts of data and are inspired by biological neural networks. Several algorithms are used to understand the correlations in each collection of data in order to acquire the best results from varying inputs. Multiple models are used to predict future events based on the data, and the network is trained to provide the desired results. The way the nodes are connected allows it to operate similarly to how the human brain does. Several correlations and hidden patterns in raw data are taken advantage of in order to cluster and categorise data.



Why do we need Neural Networks?

- It aids in the modelling of nonlinear and complicated real-world interactions.
- Because they are generalizable, they are utilized in pattern recognition.
- Text summarizing, signature identification, handwriting recognition, and many other applications are available.
- It is capable of modelling data with considerable volatility.

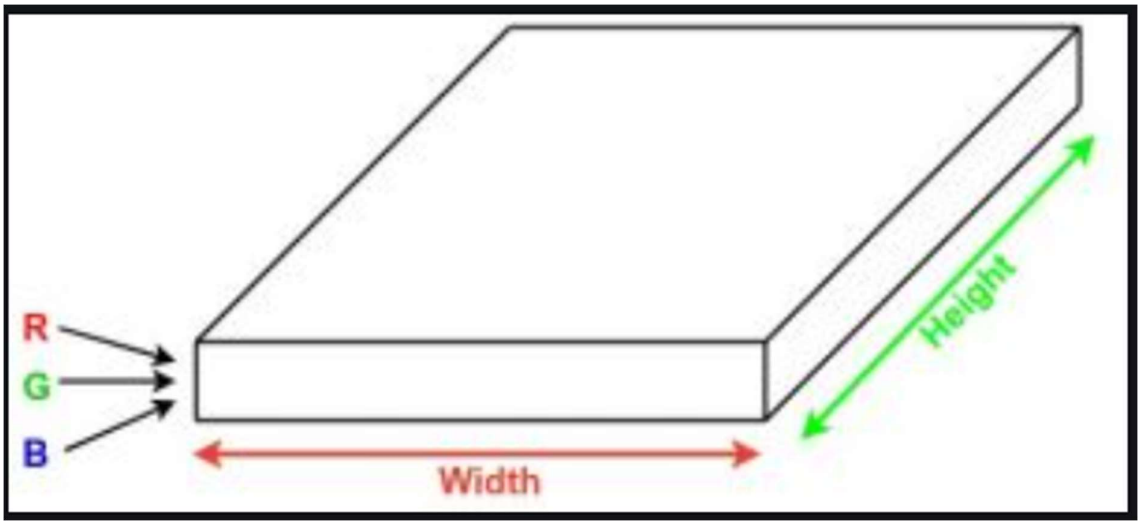
Given below are the advantages mentioned:

- Once taught, may work with inadequate information.
- Have the ability to tolerate flaws.
- Have a distributed memory system.
- Machine learning is a possibility.

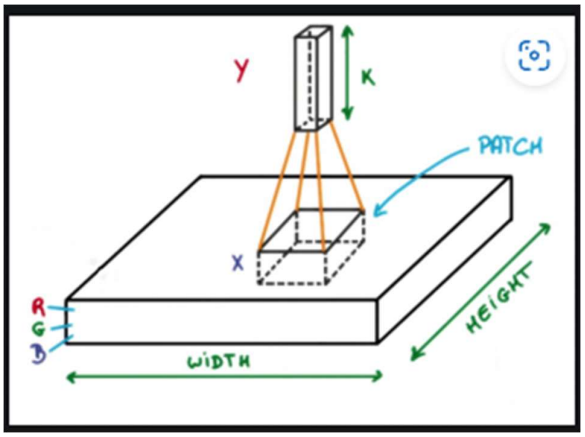
- Processing in parallel.
- Stores data over an entire network.
- It is possible to learn non-linear and complicated connections.
- Ability to generalize, i.e. the ability to infer hitherto undiscovered links from former relationships.

It has a huge future potential. New neural network-based technologies are always being created by researchers. Since everything is automated, they are highly effective at handling changes and are able to respond appropriately. Due to the widespread use of cutting-edge technology, engineers and experts in neural networks are highly sought after. Therefore, neural networks will still be a major employer in the coming years.

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (as images have red, green, and blue channels).



Now imagine taking a small patch of this image and running a small neural network on it, with say, k outputs, and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called Convolution. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.



Now let us talk about a bit of mathematics that is involved in the whole convolution process.

Convolution layers consist of a set of learnable filters (a patch in the above image). Every filter has small width and height and the same depth as that of the input volume (3 if the input layer is image input).
For example, if we must run convolution on an image with dimensions 34x34x3. The possible size of filters can be axax3, where 'a' can be 3, 5, 7, etc but small as compared to the image dimension.
During the forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have a value of 2 or 3 or even 4 for high-dimensional images) and compute the dot product between the weights of filters and patch from input volume.
As we slide our filters we will get a 2-D output for each filter and we will stack them together as a result, we will get output volume having a depth equal to the number of filters. The network will learn all the filters.

Why do deaf and mute people need such AI (Artificial Intelligence) systems? Why do we actually need thsi project and how will it help the socuiety at whole

PROBLEM:

- In our society we have people with disabilities.
- The technology is developing day by day but no significant developments are undertaken for the betterment of these people.
- About nine billion people in the world are deaf and dumb
- Communications between deaf-mute and a normal person have always been a challenging task.
- Sign language helps deaf and dumb people to communicate with other people.
- But not all people understand sign language.

We have aimed to develop a system due to security reasons. Nowadays, as we know there are various security techniques like pattern unlocks, passwords and advanced features like face and voice unlock in mobile devices. At the same time, the need for advancement to introduce a method. We aim towards building such a system that recognizes hand-drawn pattern input through the webcam of a portable device and compares patterns through template matching.

Therefore We aim to build a system that can help in communicating with normal people. In emergencies, it will be exceedingly difficult to convey their message and communicate so the solution to this problem will be to convert these gestures into text.

Through this project, we are trying to build a flexible system for physically impaired people that will ease their life. An attempt to create a sign language-to-text conversion, using information gestured by a physically impaired person which can be effectively conveyed to a normal person. The system will try to convert the sign language of the physically impaired person into text that can be interpreted by their own genre and by the rest of the world.

Literature Review

MediaPipe Hands uses an ML pipeline made up of several interconnected models: a model for detecting palms that uses the entire image and produces an orientated hand bounding box. a hand landmark model that runs on the palm detector-cropped picture region and outputs highly accurate 3D hand keypoints. The approach used in our MediaPipe Face Mesh solution, which combines a face detector with a face landmark model, is comparable to this one.

Giving the hand landmark model an appropriately cropped hand image significantly minimises the need for data augmentation (such as rotations, translations, and scaling) and instead enables the network to focus the majority of its resources on precise coordinate prediction. The crops in our pipeline can also be produced using the hand landmarks found in the preceding frame, and palm detection is only used to relocalize the hand when the landmark model is unable to do so.

A hand landmark tracking subgraph from the hand landmark module is used in the pipeline's implementation as a MediaPipe graph, and a dedicated hand renderer subgraph is used for rendering. A hand landmark subgraph from the same module and a palm detection subgraph from the palm detection module are both used internally by the hand landmark tracking subgraph.

Models for Palm Detection

In a manner similar to the face detection model in MediaPipe Face Mesh, we created a single-shot detector model intended for mobile real-time uses to detect initial hand locations. The issue of detecting hands is incredibly challenging because both our lite model and complete model must be able to recognise occluded and self-occluded hands while operating over a wide scale span (20x) relative to the image frame. It is somewhat challenging to identify hands correctly from their visual features alone, in contrast to faces, which exhibit high contrast patterns, such as in the region of the mouth and eyes. Accurate hand localisation is instead made possible by adding additional context, such as aspects of the arm, torso, or person.

Our approach employs many techniques to address the aforementioned problems. First, we train a palm detector rather than a hand detector because it is far easier to estimate the bounding boxes of stiff objects like fists and palms than it is to detect hands with articulated fingers. The non-maximum suppression technique also performs well in two-hand self-occlusion scenarios like handshakes since palms are smaller objects. The number of anchors can be reduced by a factor of 3–5 by modelling palms with square bounding boxes, or anchors in machine learning language. In order to provide greater scene context awareness, even for little objects, an encoder-decoder feature extractor is implemented (similar to the RetinaNet approach). Finally, to support a large number of anchors coming from the high scale variance, we reduce the focus loss during training.

With the aforementioned methods, we can detect palms with an average precision of 95.7%. With no decoder and a standard cross entropy loss, the baseline is just 86.22%.

Model Hand Landmark

After detecting the palm over the whole image, our subsequent hand landmark model uses regression, or direct coordinate prediction, to accomplish precise keypoint localization of 21 3D hand-knuckle coordinates inside the detected hand regions. The model acquires a reliable internal hand posture representation and is unaffected by self-occlusions or partially visible hands.

We manually added 21 3D coordinates to around 30K real-world photos to obtain ground truth data, as shown below (we take Z-value from image depth map, if it exists per corresponding coordinate). We additionally render a high-quality synthetic hand model over a variety of backgrounds and map it to the associated 3D coordinates in order to better cover the range of possible hand poses and provide additional supervision on the nature of hand geometry.

At its simplest, a neural network with some level of complexity, usually at least two layers, qualifies as a deep neural network (DNN), or deep net for short. Deep nets process data in complex ways by employing sophisticated math modeling.

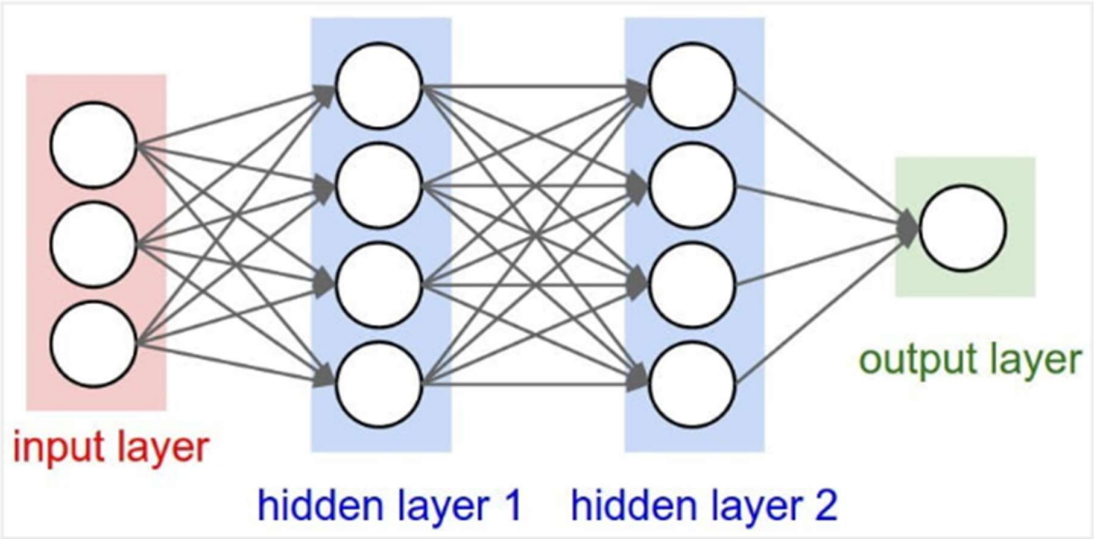
To utterly understand deep neural networks, however, it is best to see it as an evolution. A few items had to be built before deep nets existed.

First, machine learning had to get developed. ML (Machine Learning) is a framework to automate (through algorithms), statistical models, like a linear regression model, to get better at making predictions. A model is a single model that makes predictions about something. Those predictions are made with some accuracy. A model that learns—machine learning—takes all its bad predictions and tweaks the weights inside the model to create a model that makes fewer mistakes.

The learning portion of creating models spawned the development of artificial neural networks. ANNs utilize the hidden layer as a place to store and evaluate how significant one of the inputs is to the output. The hidden layer stores information regarding the input's importance, and it also makes associations between the importance of combinations of inputs.

Deep neural nets, then, capitalize on the ANN component. They say if that works so well at improving a model—because each node in the hidden layer makes both associations and grades the importance of the input to determine the output—then why not stack increasingly of these upon each other and benefit even more from the hidden layer?

So, the deep net has multiple hidden layers. 'Deep' refers to a model's layers being multiple layers deep.

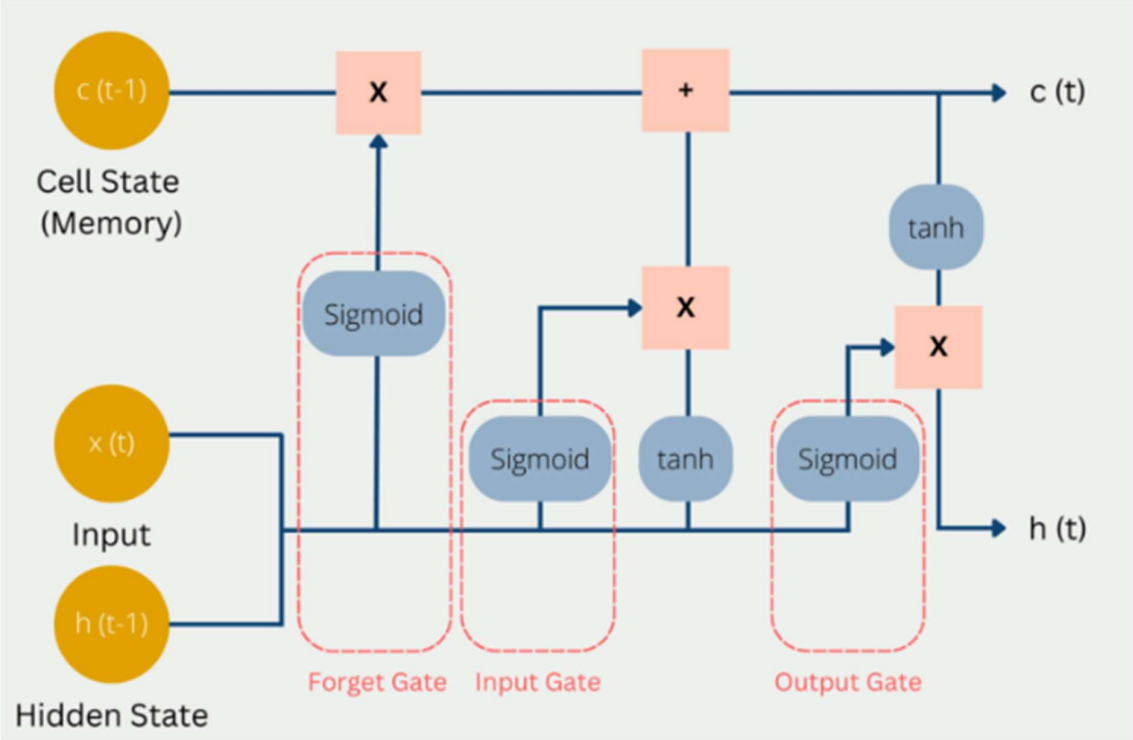


Two or more hidden layers comprise a Deep Neural Network

The Long Short-Term Memory (short: LSTM) model is a subtype of Recurrent Neural Networks (RNN). It is used to recognize patterns in data sequences, such as those that appear in sensor data, stock prices, or natural language. RNNs can do this because, in addition to the actual value, they also include its position in the sequence in the prediction.

How do Long Short-Term Memory Models work?
The problem with Recurrent Neural Networks is that they have a short-term memory to retain previous information in the current neuron. However, this ability decreases very quickly for longer sequences. As a remedy for this, the LSTM models were introduced to be able to retain past information even longer.

The problem with Recurrent Neural Networks is that they simply store the previous data in their "short-term memory." Once the memory in it runs out, it simply deletes the longest retained information and replaces it with new data. The LSTM model attempts to escape this problem by retaining only selected information in short-term memory. This short-term memory is stored in the so-called Cell State. In addition, there is also the hidden state, which we already know from normal neural networks.



These three values pass through the following gates on their way to a new Cell State and Hidden State:

1. In the so-called Forget Gate, it is decided which current and previous information is kept and which is thrown out. This includes the hidden status from the previous run and the status. These values are passed into a sigmoid function, which can only output values between 0 and 1. The value 0 means that all previous information is forgotten and 1 accordingly that all previous information is kept. The results from this are multiplied by the current Cell State, so that knowledge that is no longer needed is forgotten, as it is multiplied by 0 and thus dropped out.
2. In the Input Gate, it is decided how valuable the current input is to solve the task. For this purpose, the current input is multiplied by the hidden state and the weight matrix of the last run. All information that is important in the Input Gate is then added to the Cell State and forms the new Cell State c(t). This new Cell State is now the current state of the short-term memory and will be used in the next run.
3. In the Output Gate, the output of the LSTM model is then calculated in the Hidden State. Depending on the application, it can be, for example, a word that complements the meaning of the sentence. To do this, the sigmoid function decides what information can come through the output gate, and then the cell state is multiplied after it is activated with the tanh function.

MediaPipe is a Framework for building machine learning pipelines for processing time-series data like video, audio, etc. This cross-platform Framework works in Desktop/Server, Android, iOS, and embedded devices like Raspberry Pi and Jetson Nano.
Computer vision is a process by which we can understand the images and videos how they are stored and how we can manipulate and retrieve data from them. Computer Vision is the base or mostly used

for Artificial Intelligence. Computer-Vision is playing a key role in self-driving cars, robotics as well as in photo correction apps.

OpenCV(Open Source Computer Vision) is an image processing library created by intel and maintained by Willow Garage. OpenCV was released under BSD (Berkeley Source Distribution) license free for both academic and commercial.
It is the basic introduction to OpenCV we can continue the Applications and all the things in our upcoming articles.
Applications of OpenCV: There are lots of applications which are solved using OpenCV, some of them are listed below

- face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition
- OpenCV Functionality
- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu)

We are humans we can easily make it out that is the image of a person who is me. But if we ask the computer “is it my photo?.” The computer cannot say anything because the computer is not figuring out it all on its own.
The computer reads any image in a range of values between 0 and 255. For any color image, there are 3 primary channels -red, green, and blue.
Post reading the image, the image is passed through the model with the following compiler. The learning rate is maintained at a optimum to that the model runs accutarely.
Adadelta optimization is a stochastic gradient descent technique that addresses two issues by utilising an adaptive learning rate per dimension:

- The steady decline in learning rates during the course of training.
- The requirement of a manually chosen global learning rate.

Adadelta is a more powerful modification of Adagrad that adjusts learning rates based on a moving window of gradient updates rather than adding up all of the previous gradients. Adadelta gains new knowledge in this method even after numerous revisions. In contrast to Adagrad, Adadelta's original

version does not require you to define an initial learning rate. As with most other Keras optimizers, the initial learning rate can be set in this variation.[19]

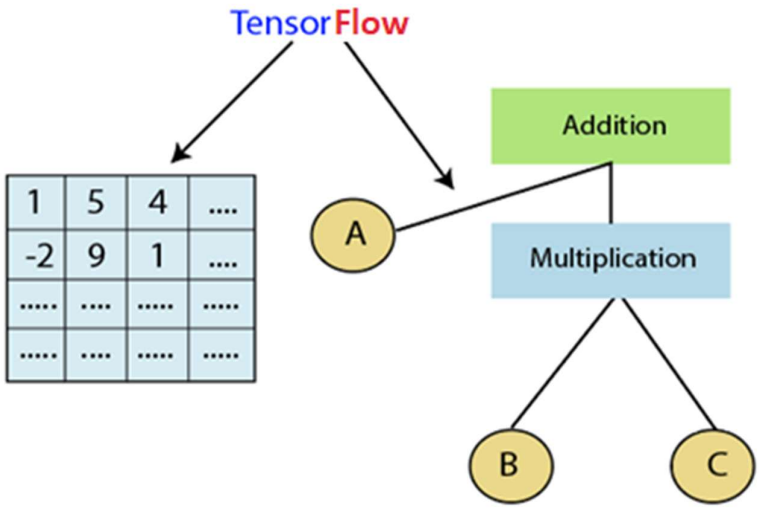
What is TensorFlow?

TensorFlow is a popular framework for machine learning and deep learning. It is a free and open-source library which is released on 9 November 2015 and developed by Google Brain Team. It is entirely based on Python programming language and is use for numerical computation and data flow, which makes machine learning faster and easier.

TensorFlow can train and run deep neural networks for image recognition, handwritten digit classification, recurrent neural network, word embedding, natural language processing, video detection, and many more. TensorFlow is run on multiple CPUs or GPUs and mobile operating systems.

The word TensorFlow is made of two words, i.e., Tensor and Flow

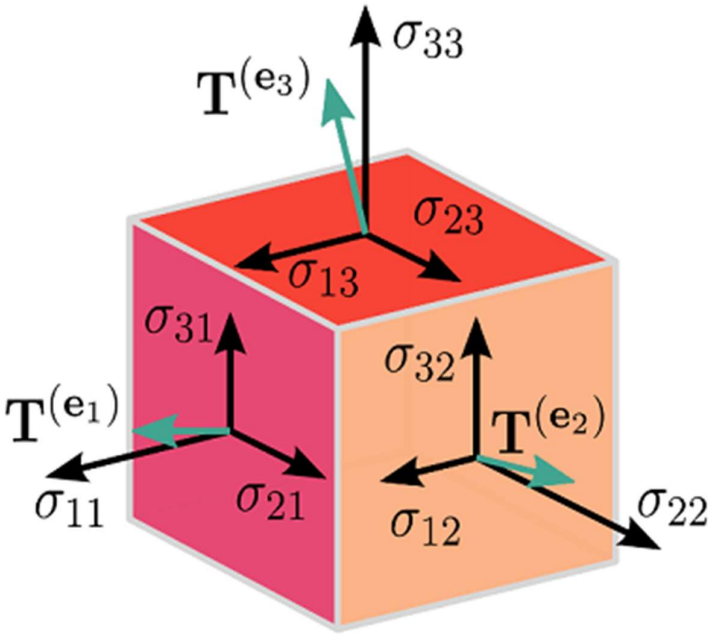
Tensor is a multidimensional array
Flow is used to define the flow of data in operation.
TensorFlow is used to define the flow of data in operation on a multidimensional array or Tensor.



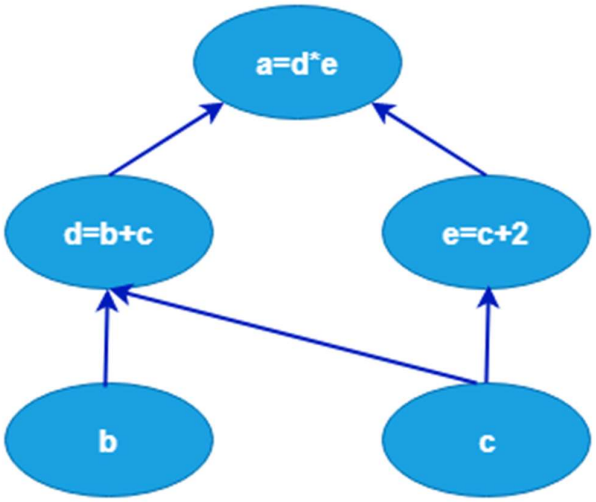
Components of TensorFlow:-

Tensor
The name TensorFlow is derived from its core framework, "Tensor." A tensor is a vector or a matrix of n-dimensional that represents all types of data. All values in a tensor hold similar data types with a known shape. The shape of the data is the dimension of the matrix or an array.

A tensor can be generated from the input data or the result of a computation. In TensorFlow, all operations are conducted inside a graph. The group is a set of calculations that takes place successively. Each transaction is called an op node connected.



Now, we can represent these operations graphically below:



Session

A session can execute the operation from the graph. To feed the graph with the value of a tensor, we need to open a session. Inside a session, we must run an operator to create an output.

It is mainly used for deep learning or machine learning problems such as Classification, Perception, Understanding, Discovering Prediction, and Creation.

Keras is a high-level, deep-learning API (Application Programming Interface) developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computations.

Keras is relatively easy to learn and work with because it provides a python frontend with an elevated level of abstraction while having the option of multiple back-ends for computation purposes. This makes Keras slower than other deep learning frameworks, but extremely beginner-friendly.

In general there are 3 main categories of object tracking-point tracking, kernel tracking and silhouette tracking. A comparative study of different object tracking methods has been listed in [2]. The various state of art tracking methods and techniques has been discussed in [1]. From this paper it is seen that Camshift method has real time performance and it can track accurately in presence of distracters and noise. Kalman filter method has error classification rate of 86% [2]. Speeded-Up-Robust-Features (SURF) is able to detect rotating object but it fails to detect highly scaled object. KLT is only for affine motion but not suitable for translational, rotational nor scaling motion [1]. There are other methods of tracking like robust multi-cue hand tracking achieved by velocity weighted features and color cue implemented by Zhigeng[3]. Liu [4] uses Viola-Jones method for hand detection. In [5], Tower method and skin color are used for hand gesture recognition tracking and control. From this literature survey we found out that Camshift method is best suited for our System since it is inexpensive i.e. it can be used with the in-built web camera and gives good result in terms of speed and accuracy.

After segmentation, we have to search the input image in a database. The image has to be described or represented by certain features. Shape is an important visual feature of an image. There are various methods used in Feature Extraction. [20] describes those methods: Contour shape techniques only exploit shape boundary information. Methods based on the Hausdorff distance are useful for locating objects in an image or sub-image matching. The retrieval performance of the Generic Fourier descriptor (GFD) demonstrates it is a desirable solution to generic shape representations. If storage is a concern, Fourier Descriptor (FD) can be considered.

There have been varied approaches of gesture recognition ranging from statistical modeling such as Principal component Analysis (PCA)which is used to extract the features from the hand gesture's images, Hidden Markov Model (HMM)[11] [12]which is a double stochastic process governed by a Markov chain and is used to model wide range of data, to approaches based on soft computing tools such as Artificial Neural Networks (ANN) [13] [14]useful for complex pattern recognition and classification tasks, Genetic Algorithms (GA)[15] [16] is used for feature selection problem. Techniques like Support Vector Machines (SVM)[18] employed for classification.

DataSet used for the Project:

The data set used in the process is of image recognition so images each of word of 30 frames has been used in the data there have been five actions for which the model has been trained for namely hello, I love you, thank you, one, and victory. The data set hence has been stored in a separate file or a folder adjacent to the model by using the OS functionality of the system. The data thus generated contains the numpy arrays of all these gestures that the model has to be trained for. The model does is able to detect these gestures up to some extent with a given probability and hence you are able to obtain efficiency of greater than 96% with a model accuracy of more than 87.5%. There is an additional function for adding new gestures into the system which enables us to add a new folder in the existing set of folders for the new gestures and provide it with additional 30 training images with 30 frames each. This helps us keep on improving the model so that the convolution neural network used has ample of data and is able to predict the required gestures successfully. The data thus can be referenced from these folders and can be used for training the model this enables us uh to have a separate storage for the training data rather than integrating it in the same Python notebook and thus as a future scope we can use cloud or other alternatives as storage devices to store large amounts of data and enable us to cater to a large variety of needs.

Methodology:

The process for creating the model involved the following steps:

- Import and Install Dependencies: Importing the dependencies and different libraries taht have been used in the algorithm.
- Key Points using Media-Pipe Holistic: This module helps us capture and model the different data points in the face and hands so that the system can tjhen store the training data.
- Extract keypoint Values: Based on the sensation of the different keypoints, that need to be converted to arrays so that they can be stored in the form of a ayyay so that they can be passed to the model.
- Setup Folders for Collection: This module is used to create folders for the multiple actions that will be stored in the model
- Collect keypoint values for Training and Testing: The Data set is now split to training and testing data. This enables us to furthur use the capabilities of the ML System
- Add a new Gesture: This module helps you add the new gesture and store the same in the folder
- Preprocess Data to create Lables and Features: THis helps us to differentiate between multiple sequences oif arrays.
- Build and Train LSTM Neural Network Model: Here we build and train the LSTM Model and choode the compluier and the learning rates for the algorithm.
- Make Predictions: Here we make predictions on the kind of data
- Save Weights
- Evaluation using Confusion matrix and Accuracy: This enables evaluation of the accuracy of the system with the use of Confusion Matrix
- Test Real-Time: This gives us the final results of the project and enables us to recognise the gestures.

Post the accomplishment of each of these steps, we were able to successfully create the running implementation of the hand gesture recoognition system.

Implementation:

The code implementations for the same have been pasted below.

Import and Install Dependencies

In [1]:
```
import cv2
import numpy as np
import os
import matplotlib.pyplot as plt
import time
import mediapipe as mp
```

Key Points using Media-Pipe Holistic

In [2]:
```
mp_holistic = mp.solutions.holistic
mp_drawing = mp.solutions.drawing_utils
```

In [3]:
```
def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image.flags.writeable = False #image not editable or writable
    results = model.process(image)
    image.flags.writeable = True #image is writable and editable
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    return image, results
```

In [4]:
```
def draw_landmarks(image, results):
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_TESSELATION)
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS)
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS)
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS)
```

In [5]:
```
def updated_draw_landmarks(image, results):
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_TESSELATION, mp_drawing.DrawingSpec(color
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS)
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS)
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS)
```

In [64]:
```
cap = cv2.VideoCapture(1)
#set mediapipe Model
with mp_holistic.Holistic(min_detection_confidence = 0.5, min_tracking_confidence = 0.5) as holistic:
    while cap.isOpened():
        ret, frame = cap.read()

        image, results = mediapipe_detection(frame, holistic)

        updated_draw_landmarks(image,results)

        cv2.imshow('OpenCV Feed', image)
        print(results)
        if cv2.waitKey(10) & 0xFF==ord("q"):
            break

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

```
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
```

In [65]:
```
frame
```

Out[65]:
```
array([[[ 95,  81,  79],
        [ 94,  81,  80],
        [ 93,  81,  80],
        ...,
        [165, 158, 168],
        [162, 157, 166],
        [163, 158, 166]],

       [[ 94,  81,  79],
        [ 94,  81,  79],
        [ 93,  82,  79],
        ...,
        [163, 156, 167],
        [163, 157, 167],
        [161, 157, 166]],

       [[ 94,  84,  77],
        [ 94,  84,  77],
        [ 93,  84,  77],
        ...,
        [163, 156, 168],
        [164, 159, 170],
        [160, 157, 167]],

        [ 77,  95, 143]]], dtype=uint8)
```

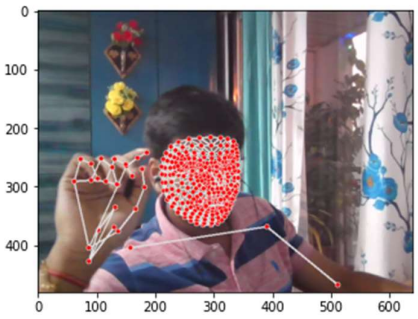
In [8]:
```
results
```

Out[8]:
```
mediapipe.python.solution_base.SolutionOutputs
```

In [66]:
```
draw_landmarks(frame,results)
```

In [67]:
```
plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

Out[67]:
```
<matplotlib.image.AxesImage at 0x2749366e820>
```



Extract keypoint Values

In [11]:
```
len(results.pose_landmarks.landmark)
```

Out[11]:
```
33
```

In [12]:
```
pose = []
for res in results.pose_landmarks.landmark:
    pose_array = np.array([res.x, res.y, res.z, res.visibility])
    pose.append(pose_array)
```

In [13]:
```
l_hand.shape
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-13-cfe195f79948> in <module>
----> 1 l_hand.shape

NameError: name 'l_hand' is not defined
```

In [19]:
```
pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.pose_landmarks.landmark] if results.pose_landmarks else
face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landmark] if results.face_landmarks else np.zeros(468*3)
```

In [20]:
```
r_hand = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.landmark] if results.right_hand_landmarks else r
```

In [21]:
```
l_hand = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.landmark] if results.left_hand_landmarks else np.
```

In [22]:
```
face.shape
```

Out[22]:
```
(1404,)
```

In [23]:
```
def extract_keypoints(results):
    pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.pose_landmarks.landmark] if results.pose_landmarks
    face = np.array([[res.x, res.y, res.z] for res in results.face_landmarks.landmark] if results.face_landmarks else np.zeros(46
    r_hand = np.array([[res.x, res.y, res.z] for res in results.right_hand_landmarks.landmark] if results.right_hand_landmarks el
    l_hand = np.array([[res.x, res.y, res.z] for res in results.left_hand_landmarks.landmark] if results.left_hand_landmarks else
    return np.concatenate([pose, face, r_hand, l_hand])
```

In [24]:
```
extract_keypoints(results)
```

Out[24]:
```
array([ 0.59445918,  0.55627567, -0.50982147, ...,  0.        ,
        0.        ,  0.        ])
```

Setup Folders for Collection

In [26]:
```
# Path for exported data, numpy arrays
DATA_PATH = os.path.join('MP_Data')

# Actions that we try to detect
actions = np.array(['hello', 'thanks', 'iloveyou', 'one','victory'])

# Thirty videos worth of data
no_sequences = 30

# Videos are going to be 30 frames in length
sequence_length = 30
```

In [ ]:
```

```

In [27]:
```
for action in actions:
    for sequence in range(no_sequences):
        try:
            os.makedirs(os.path.join(DATA_PATH, action, str(sequence)))
        except:
            pass
```

Collect keypoint values for Training and Testing

In [28]:
```
cap = cv2.VideoCapture(1)
#set mediapipe Model
with mp_holistic.Holistic(min_detection_confidence = 0.5, min_tracking_confidence = 0.5) as holistic:

    # for the actions
    for action in actions:
        #for the sequences aka videos
        for sequence in range(no_sequences):
            #for teh number of frames
            for frame_num in range(sequence_length):
                ret, frame = cap.read()

                image, results = mediapipe_detection(frame, holistic)

                updated_draw_landmarks(image,results)

                # NEW Apply wait logic
                if frame_num == 0:
                    cv2.putText(image, 'STARTING COLLECTION', (120,200),
                                cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255, 0), 4, cv2.LINE_AA)
                    cv2.putText(image, 'Collecting frames for {} Video Number {}'.format(action, sequence), (15,12),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                    # Show to screen
                    cv2.imshow('OpenCV Feed', image)
                    cv2.waitKey(2000)
                else:
                    cv2.putText(image, 'Collecting frames for {} Video Number {}'.format(action, sequence), (15,12),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                    # Show to screen
                    cv2.imshow('OpenCV Feed', image)

                 # NEW Export keypoints
                keypoints = extract_keypoints(results)
                npy_path = os.path.join(DATA_PATH, action, str(sequence), str(frame_num))
                np.save(npy_path, keypoints)

            #Break Gracefully
            if cv2.waitKey(10) & 0xFF==ord("q"):
                break

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

In [116]:
```
os.path
```

Out[116]:
```
<module 'ntpath' from 'C:\\Users\\AVACADO\\anaconda3\\lib\\ntpath.py'>
```

Add a new Gesture

In [30]:
```
#function to add new gesture to the database and train it at the same time
def add_gesture():
    new_ges = input("Enter the name of the new gesture: ")
    for sequence in range(no_sequences):
        os.makedirs(os.path.join(DATA_PATH, new_ges, str(sequence)))
        actions = np.append(actions, new_ges)
        cap = cv2.VideoCapture(1)
        #set mediapipe Model
        with mp_holistic.Holistic(min_detection_confidence = 0.5, min_tracking_confidence = 0.5) as holistic:
            #for the sequences aka videos
            for sequence in range(no_sequences):
                #for the number of frames
                for frame_num in range(sequence_length):
                    ret, frame = cap.read()

                    image, results = mediapipe_detection(frame, holistic)

                    updated_draw_landmarks(image,results)

                    # NEW Apply wait logic
                    if frame_num == 0:
                        cv2.putText(image, 'STARTING COLLECTION', (120,200),
                                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255, 0), 4, cv2.LINE_AA)
                        cv2.putText(image, 'Collecting frames for {} Video Number {}'.format(new_ges, sequence), (15,12),
                                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                        # Show to screen
                        cv2.imshow('OpenCV Feed', image)
                        cv2.waitKey(2000)
                    else:
                        cv2.putText(image, 'Collecting frames for {} Video Number {}'.format(new_ges, sequence), (15,12),
                                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                        # Show to screen
                        cv2.imshow('OpenCV Feed', image)

                    # NEW Export keypoints
                    keypoints = extract_keypoints(results)
                    npy_path = os.path.join(DATA_PATH, new_ges, str(sequence), str(frame_num))
                    np.save(npy_path, keypoints)
```

```
                    # NEW Export keypoints
                    keypoints = extract_keypoints(results)
                    npy_path = os.path.join(DATA_PATH, new_ges, str(sequence), str(frame_num))
                    np.save(npy_path, keypoints)

                #Break Gracefully
                if cv2.waitKey(10) & 0xFF==ord("q"):
                    break
        cap.release()
        cv2.destroyAllWindows()
```

[31]:
```
add_gesture()
```

```
Enter the name of the new gesture: two
```

Preprocess Data to create Lables and Features

In [29]:
```
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
```

In [30]:
```
label_map = {label:num for num, label in enumerate(actions)}
```

In [31]:
```
label_map
```

Out[31]:
```
{'hello': 0, 'thanks': 1, 'iloveyou': 2, 'one': 3, 'victory': 4}
```

In [32]:
```
sequences, lables = [], []
for action in actions:
    for sequence in range(no_sequences):
        window = []
        for frame_num in range(sequence_length):
            res = np.load(os.path.join(DATA_PATH, action, str(sequence), "{}.npy".format(frame_num)))
            window.append(res)
        sequences.append(window)
        lables.append(label_map[action])
```

In [33]:
```
np.array(sequences).shape
```

Out[33]:
```
(150, 30, 1662)
```

In [34]:
```
np.array(lables).shape
```

Out[34]:
```
(150,)
```

In [35]:
```
x = np.array(sequences)
```

In [36]:
```
x.shape
```

Out[36]:
```
(150, 30, 1662)
```

In [37]:
```
#converted to one hot encoded value of array width of 4
y = to_categorical(lables).astype(int)
```

In [38]:
```
y.shape
```

Out[38]:
```
(150, 5)
```

In [39]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.05, random_state=True)
```

In [40]:
```
y_test.shape
```

Out[40]:
```
(8, 5)
```

Build and Train LSTM Neural Network Model

In [41]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import TensorBoard
```

In [42]:
```
import keras
```

In [44]:
```
#Setup Directory to store Logs
log_dir = os.path.join('Logs')
tb_callback = TensorBoard(log_dir=log_dir)
```

In [45]:
```
model = Sequential()
#add 3 sets of LSTM Layers
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(30,1662)))
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=False, activation='relu')) # Last LSTM Layer so no need for sequences to be returned
#3 Layers of Densse NN Added
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(actions.shape[0], activation='softmax'))
```

In [46]:
```
x.shape
```

Out[46]:
```
(150, 30, 1662)
```

In [47]:
```
opt = keras.optimizers.Adadelta(learning_rate=0.01)
```

In [48]:
```
model.compile(optimizer= opt, loss='categorical_crossentropy', metrics=['categorical_accuracy'])
#Multi Class Classificaltion, hence loss has to be used as this one only
```

In [49]:
```
model.fit(x_train, y_train, epochs=2000, callbacks=[tb_callback])
# Callback to log
```

```
5/5 [==============================] - 1s 118ms/step - loss: 0.2455 - categorical_accuracy: 0.8803
Epoch 159/2000
5/5 [==============================] - 1s 115ms/step - loss: 0.1555 - categorical_accuracy: 0.9437
Epoch 160/2000
5/5 [==============================] - 1s 115ms/step - loss: 0.1594 - categorical_accuracy: 0.9648
Epoch 161/2000
5/5 [==============================] - 1s 140ms/step - loss: 0.1926 - categorical_accuracy: 0.9577
Epoch 162/2000
5/5 [==============================] - 1s 120ms/step - loss: 0.1775 - categorical_accuracy: 0.9366
Epoch 163/2000
5/5 [==============================] - 1s 111ms/step - loss: 0.1969 - categorical_accuracy: 0.9366
Epoch 164/2000
5/5 [==============================] - 1s 112ms/step - loss: 0.1495 - categorical_accuracy: 0.9648
Epoch 165/2000
1/5 [=====>........................] - ETA: 0s - loss: 0.0837 - categorical_accuracy: 0.9688
```

In [50]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 30, 64)            442112

 lstm_1 (LSTM)               (None, 30, 128)           98816

 lstm_2 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 5)                 165

=================================================================
Total params: 596,741
Trainable params: 596,741
Non-trainable params: 0
_________________________________________________________________
```

Make Predictions

In [51]:
```
res = model.predict(x_test)
```

```
1/1 [==============================] - 0s 449ms/step
```

In [54]:
```
actions[np.argmax(res[2])]
```

Out[54]:
```
'iloveyou'
```

In [55]:
```
actions[np.argmax(y_test[2])]
```

Out[55]:
```
'iloveyou'
```

Evaluate using Confusion matrix and Accuracy

In [57]:
```
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score
```

In [58]:
```
yhat = model.predict(x_test)
```

```
1/1 [==============================] - 0s 33ms/step
```

In [59]:
```
ytrue = np.argmax(y_test, axis=1).tolist()
yhat = np.argmax(yhat, axis=1).tolist()
```

In [60]:
```
multilabel_confusion_matrix(ytrue, yhat)
```

Out[60]:
```
array([[[6, 0],
        [0, 2]],

       [[6, 0],
        [0, 2]],

       [[7, 0],
        [0, 1]],

       [[6, 1],
        [0, 1]],

       [[6, 0],
        [1, 1]]], dtype=int64)
```

In [61]:
```
accuracy_score(ytrue, yhat)
```

Out[61]:
```
0.875
```

Test Real-Time

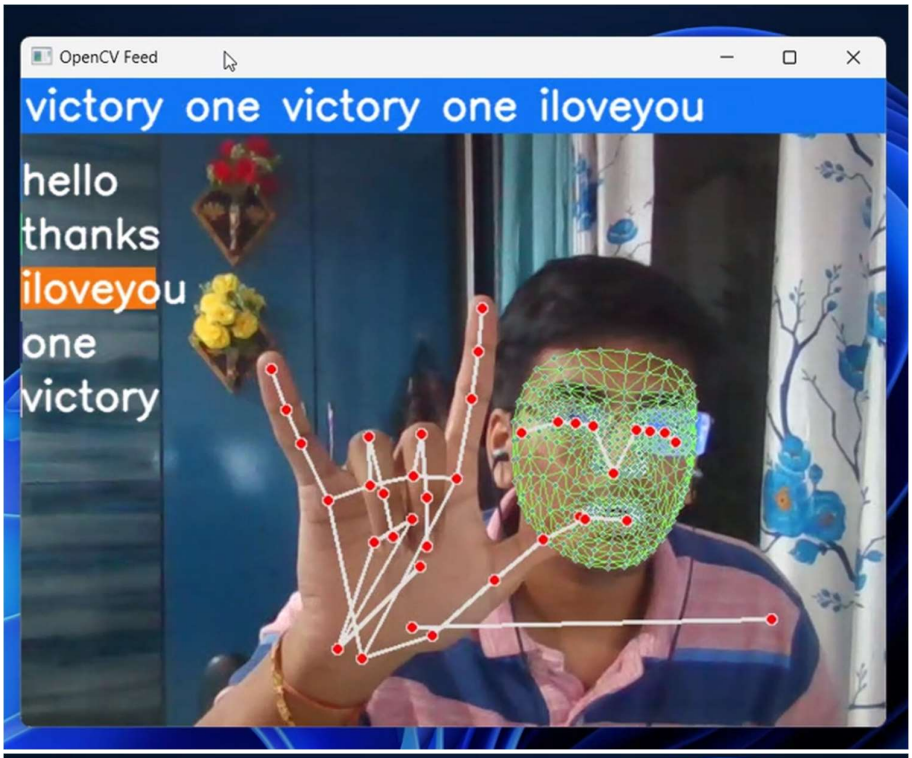
In [62]:
```
colors = [(245,117,16), (117,245,16), (16,117,245), (128, 0, 0), (204,204,255)]
def prob_viz(res, actions, input_frame, colors):
    output_frame = input_frame.copy()
    for num, prob in enumerate(res):
        cv2.rectangle(output_frame, (0,60+num*40), (int(prob*100), 90+num*40), colors[num], -1)
        cv2.putText(output_frame, actions[num], (0, 85+num*40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2, cv2.LINE_AA)

    return output_frame
```

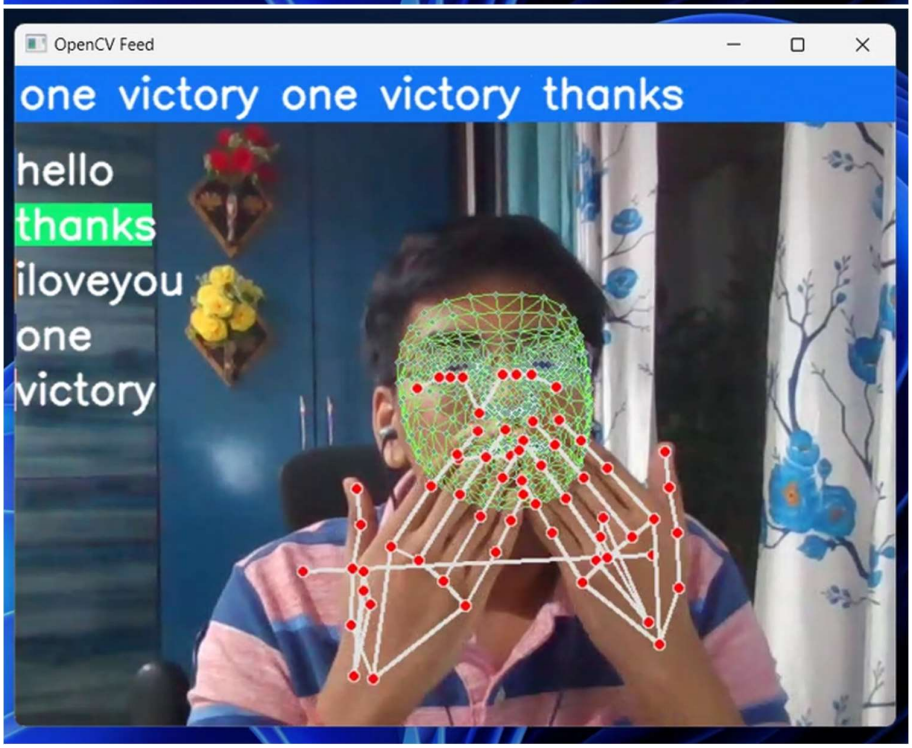
In [71]:
```
# 1. New detection variables
sequence = []
sentence = []
threshold = 0.8

cap = cv2.VideoCapture(1)
# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():

        # Read feed
        ret, frame = cap.read()

        # Make detections
        image, results = mediapipe_detection(frame, holistic)
        print(results)

        # Draw landmarks
        updated_draw_landmarks(image, results)

        # 2. Prediction logic
        keypoints = extract_keypoints(results)
#         sequence.insert(0,keypoints)
#         sequence = sequence[:30]
        sequence.append(keypoints)
        sequence = sequence[-30:]

        if len(sequence) == 30:
            res = model.predict(np.expand_dims(sequence, axis=0))[0]
            print(actions[np.argmax(res)])


        #3. Viz logic
            if res[np.argmax(res)] > threshold:
                if len(sentence) > 0:
                    if actions[np.argmax(res)] != sentence[-1]:
                        sentence.append(actions[np.argmax(res)])
                else:
                    sentence.append(actions[np.argmax(res)])

            if len(sentence) > 5:
                sentence = sentence[-5:]

            # Viz probabilities
            image = prob_viz(res, actions, image, colors)

        cv2.rectangle(image, (0,0), (640, 40), (245, 117, 16), -1)
        cv2.putText(image, ' '.join(sentence), (3,30),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

        # Show to screen
        cv2.imshow('OpenCV Feed', image)

        # Break gracefully
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

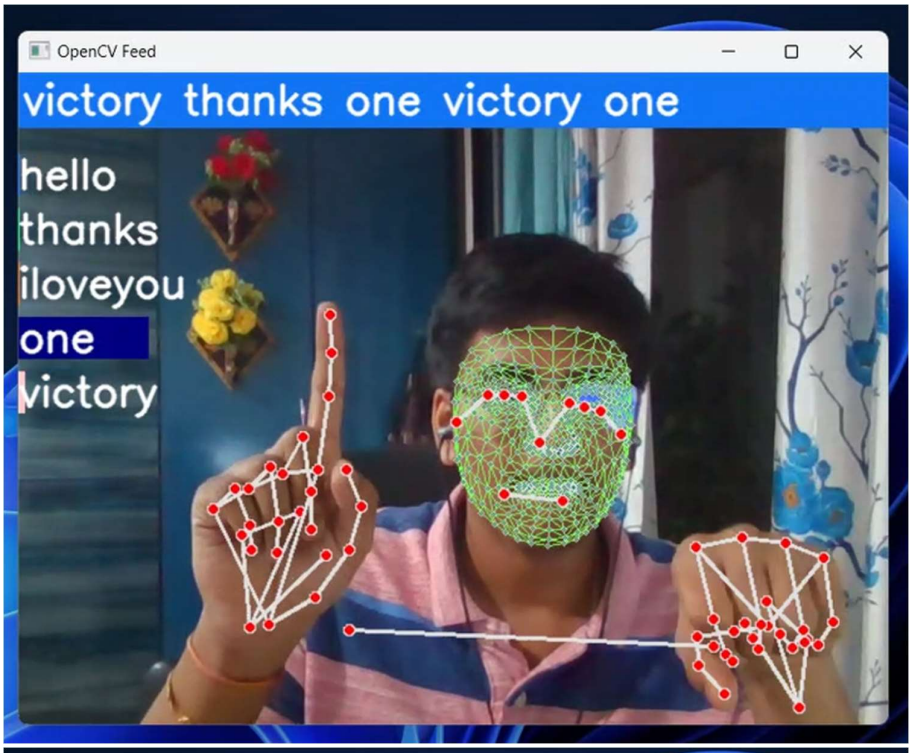
```
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
```

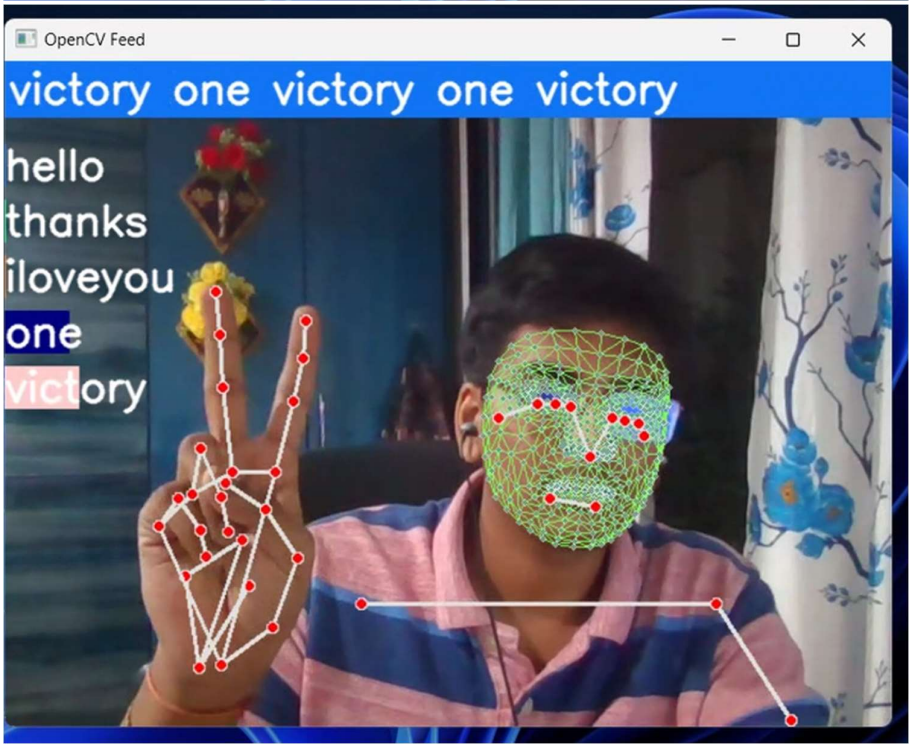
Results and Discussion:

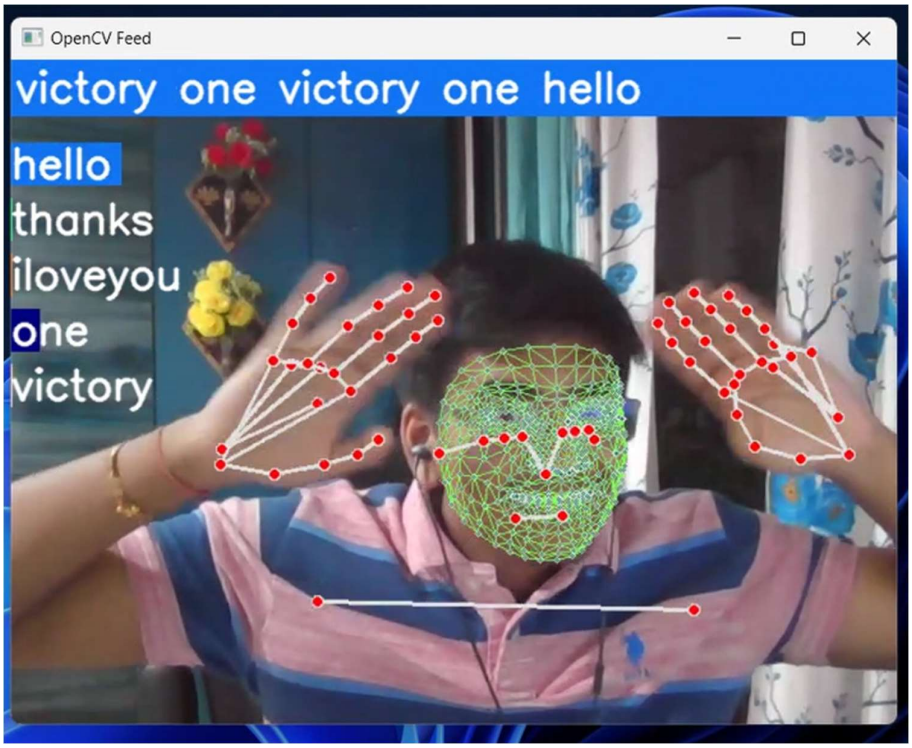
Based on the entire code and implementation of the projerct, it can be tested in real time to generate the required solutions. The predictions made by the algorithm were a lot accurate with recognition for both the left as well as the right hand gestures.






OpenCV Feed
victory thanks one victory one
hello
thanks
iloveyou
one
victory


OpenCV Feed
victory one victory one victory
hello
thanks
iloveyou
one
victory



Like these, the gestures have been detected successfully and the probabilities of each of these has been displayed with the help of the bar adjescent to it. This is just a tip of the iceberg to the capabilities of the system.

Conclusion

The LSTM approach, and an ANN algorithm are used in this research to present a real-time hand gesture detection system. The suggested gesture recognition system supports a variety of hand motions on a platform that is based on vision. Both single handed and double handed gestures can be made using the method.

Through this project we are trying to build flexible system for the physically impaired people that will ease their life. An attempt to create a sign language to text conversion wireless lightweight system, through the use of information gestured by physically impaired person which can be effectively conveyed to a normal person. The system will try to convert the sign language of the physically impaired person into text that can interpreted by their own genre and also to the rest of the world. The interaction between a human and a computer can be fluid and effective when using gestures as an input modality. Aside from being silent, requiring little eye attention, and sometimes feeling more natural than, say, pushing a key on a keyboard, gesture control has few advantages over more conventional modalities. This paper introduced a unique vector-only gesture recognition method. The arrays of frames are used in this paper to identify hand motions. The Sign Sequence and Template Matching Algorithm with ANN and LSTM are the foundations on which the Gesture Recognition Algorithm was constructed. The hand gesture recognition system will be employed in human-machine interfaces in the future, such as controlling flying machines and other electrical devices with the wave of a hand.

References

1. Alper Yilmaz, Omar Javed, Mubarak Shah, "Object Tracking: A Survey", ACM Computing Surveys, Vol. 38, No. 4, Article 13, December 2006.
2. Fariborz Mahmoudi, Mehdi Parviz, "Visual Hand Tracking Algorithms," Proceedings of the Geometric Modeling and Imaging - New Trends (GMAI'06), 2006.
3. Zhigeng Pan, Yang Li, Mingmin Zhang, Chao Sun, Kangde Guo, Xing Tang, Steven Zhiying Zhou, "A Real-time Multi-cue Hand Tracking Algorithm Based on Computer Vision," IEEE Virtual Reality 2010, 20-24 March, Waltham, Massachusetts, USA.
4. Liu Yun, Zhang Peng, "An Automatic Hand Gesture Recognition System Based on Viola-Jones Method and SVMs". 2009 Second International Workshop on Computer Science and Engineering.
5. Pham The Bao, Nguyen Thanh Binh, "A New Approach To Hand Tracking and Gesture Recognition by a new feature type and HMM", 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery.
6. Deshmukh K.S., Shinde G. N, "An Adaptive Color Image Segmentation", Electronic Letters on Computer Vision and Image Analysis 5(4):12-23, 2005.
7. Frank Y. Shih, Shouxian Cheng, "Automatic seeded region growing for color image segmentation", Image and Vision Computing 23(2005) 877-886.
8. Guoliang Yang, Huan Li, Li Zhang and Yue Cao,"Research On a Skin Color Detection Algorithm Based on Self-adaptive Skin Color Model", 2010 International Conference On Communications And Intelligence Information Security.
9. Noel O'Connor, Sorin Sav, Tomasz Adamek, Vasileios Mezaris, Ioannis Kompatsiaris, Tsz Ying Lui, Ebroul Izquierdo, Christian Ferran Bennström, Josep R Casas," Region and Object Segmentation Algorithms in the Qimera Segmentation Platform".
10. Ryzard S. Choras, "Hand Shape and Hand Gesture Recognition", 2009 IEEE Symposium on Industrial Electronics and Applications (ISIEA 2009), October 4-6, 2009.
11. Nguyen Dang Binh, Toshiaki Ejima, "Real-Time Hand Gesture Recognition using Pseudo 3-D Hidden Markov Model", Proc. 5th IEEE Int. Conf. on Cognitive Informatics (ICCI'06), 2006.
12. Assoc Prof Abd Manan Ahmad, Dr Abdullah Bade, et al. "Using Principal Component Analysis And Hidden Markov Model For Hand Recognition Systems", 2009 International Conference on Information and Multimedia Technology.
13. Stefan Oniga, Alin Tisan, Daniel Mic, Attila Buchman and Andrei Vida-Ratiu, "Hand Postures Recognition System Using Artificial Neural Networks Implemented in FPGA", 30th ISSE 2007.
14. Md. Rezwanul Ahsan, Muhammad Ibn Ibrahimy, Othman O. Khalifa, "Electromygraphy (EMG) Signal based Hand Gesture Recognition using Artificial Neural Network (ANN)", 2011 4th International Conference on Mechatronics (ICOM), 17-19 May 2011.
15. Ho-Duck Kim, Chang-Hyun Park, Hyun-Chang Yang, and Kwee-Bo Sim," Genetic Algorithm Based Feature Selection Method Development for Pattern Recognition", SICE-ICASE International Joint Conference 2006 Oct. 18-2 1, 2006.
16. Angel Dacal-Nieto1,2, Esteban Vázquez-Fernández1,3, Arno Formella2, Fernando Martin3, Soledad Torres-Guijarro1, Higinio González-Jorge1,"A genetic algorithm approach for feature selection in potatoes classification by computer vision", 2009 IEEE.
17. Majida Ali Abed, Ahmad Nasser Ismail and Zubadi Matiz Hazi," Pattern recognition Using Genetic Algorithm", International Journal of Computer and Electrical Engineering, Vol. 2, No. 3, June, 2010
18. Yu Yuan and Kenneth Barner, "Hybrid Feature Selection for Gesture Recognition using Support Vector Machines", ICASSP 2008.

19. Team, Keras. “Keras Documentation: Adadelta.” *Adadelta*, keras.io/api/optimizers/adadelta. Accessed 28 Oct. 2022.
20. Abdullah, King. “Introduction to MediaPipe.” LearnOpenCV, 1 March 2022, https://learnopencv.com/introduction-to-mediapipe/.
21. Guide, Step. “What is Keras and Why it so Popular in 2021.” Simplilearn, 18 September 2021, https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-keras#what_is_keras.
22. Johnson, Jonathan. “What's a Deep Neural Network? Deep Nets Explained – BMC Software | Blogs.” BMC Software, 27 July 2020, https://www.bmc.com/blogs/deep-neural-network/.
23. “Long Short-Term Memory Networks (LSTM)- simply explained!” Data Basecamp, 4 June 2022, https://databasecamp.de/en/ml/lstms.
24. “Machine learning & OpenCV.” Intellipaat, 23 April 2021, https://intellipaat.com/blog/tutorial/artificial-intelligence-tutorial/machine-learning-opencv/.
25. “OpenCV - Overview.” GeeksforGeeks, https://www.geeksforgeeks.org/opencv-overview/.
26. “tf.keras.optimizers.Adadelta.” TensorFlow, https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adadelta.
27. “What is Tensorflow | TensorFlow Introduction.” Javatpoint, https://www.javatpoint.com/tensorflow-introduction.