#### A

***PROJECT REPORT***

On

#### Real Time Hand Gesture Detection Using Tensorflow and Open CV

*Submitted in partial fulfillment of the requirements for the degree of*

#### BACHELOR OF TECHNOLOGY

**Session :- 2024**

#### Under Guidance of Submitted by

Mr. Aaditya Maheswari Nikhil Mail (20ETCCS079)

Assistant Professor Akshi Jain (20ETCCS006)

Dept. of CSE Charvi Gokhru (20ETCCS020 TINJRIT, Udaipur 8th Semester, CSE

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001 MAY - 2024**

I | P a g e



Department of Computer Science and Engineering Techno India NJR Institute of Technology, Udaipur-313001

# Certificate

This is to certify that project work titled Real Time Hand Gesture Detection Using Tensorflow and OpenCV by Nikhil Mali, was successfully carried out in the Department of Computer Science and Engineering, TINJRIT and the report is approved for submission in the partial fulfillment of the requirements for award of degree of Bachelor of Technology in Computer Science and Engineering.

Mr. Aaditya Maheshwari Dr. Rimpy Bishnoi

Assistant Professor CSE Head of Department

TINJRIT, Udaipur Dept. of CSE TINJRIT, Udaipur

Date...................... Date......................



Department of Computer Science and Engineering Techno India NJR Institute of Technology, Udaipur-313001

# Certificate

This is to certify that project work titled Real Time Hand Gesture Detection Using Tensorflow and OpenCV by Akshi Jain, was successfully carried out in the Department of Computer Science and Engineering, TINJRIT and the report is approved for submission in the partial fulfillment of the requirements for award of degree of Bachelor of Technology in Computer Science and Engineering.

Mr. Aaditya Maheshwari Dr. Rimpy Bishnoi

Assistant Professor CSE Head of Department

TINJRIT, Udaipur Dept. of CSE TINJRIT, Udaipur

Date...................... Date......................



Department of Computer Science and Engineering Techno India NJR Institute of Technology, Udaipur-313001

# Certificate

This is to certify that project work titled Real Time Hand Gesture Detection Using Tensorflow and OpenCV by Charvi Gokhru, was successfully carried out in the Department of Computer Science and Engineering, TINJRIT and the report is approved for submission in the partial fulfillment of the requirements for award of degree of Bachelor of Technology in Computer Science and Engineering.

Mr. Aaditya Maheshwari Dr. Rimpy Bishnoi

Assistant Professor CSE Head of Department

TINJRIT, Udaipur Dept. of CSE TINJRIT, Udaipur

Date...................... Date......................



Department of Computer Science and Engineering Techno India NJR Institute of Technology, Udaipur-313001

# Examiner Certificate

This is to certify that the following student **Nikhil Mali** of final year B.Tech. (Computer Science and Engineering), was examined for the project work titled ***Real Time Hand Gesture Detection Using Tensorflow and OpenCV*** during the academic year 2023 – 2024 at Techno India NJR Institute of Technology, Udaipur.

#### Remarks:

**Date:**

Signature Signature

(**Internal Examiner**) (**External Examiner**)

Name :- ……………………… Name :- ………………………

Designation:- ……………….. Designation:- ………………..

Department: - ………………. Department: - ……………….

Organization:- ……………… Organization:- ……………



Department of Computer Science and Engineering Techno India NJR Institute of Technology, Udaipur-313001

# Examiner Certificate

This is to certify that the following student **Akshi Jain** of final year B.Tech. (Computer Science and Engineering), was examined for the project work titled ***Real Time Hand Gesture Detection Using Tensorflow and OpenCV*** during the academic year 2023 – 2024 at Techno India NJR Institute of Technology, Udaipur.

#### Remarks:

**Date:**

Signature Signature

(**Internal Examiner**) (**External Examiner**)

Name :- ……………………… Name :- ………………………

Designation:- ……………….. Designation:- ………………..

Department: - ………………. Department: - ……………….

Organization:- ……………… Organization:- ……………



Department of Computer Science and Engineering Techno India NJR Institute of Technology, Udaipur-313001

# Examiner Certificate

This is to certify that the following student **Charvi Gokhru** of final year B.Tech. (Computer Science and Engineering), was examined for the project work titled ***Real Time Hand Gesture Detection Using Tensorflow and OpenCV*** during the academic year 2023 – 2024 at Techno India NJR Institute of Technology, Udaipur.

#### Remarks:

**Date:**

Signature Signature

(**Internal Examiner**) (**External Examiner**)

Name :- ……………………… Name :- ………………………

Designation:- ……………….. Designation:- ………………..

Department: - ………………. Department: - ……………….

Organization:- ……………… Organization:-……………

# Preface

In human-computer interaction, communicating with devices seamlessly is crucial. While keyboards and mice have been reliable, there's a growing need for intuitive interfaces that understand natural gestures. Hand gesture detection marks a significant step in this direction, bridging human expression with technological response

.

This report explores real-time hand gesture detection using TensorFlow and OpenCV. In an era of AI and computer vision, understanding how these tools enable new forms of interaction is essential.

Beginning with foundational concepts, like neural networks and CNNs, readers grasp the theory behind intelligent systems interpreting gestures.Moving forward, practical implementation is detailed. Readers learn setting up TensorFlow and OpenCV, dataset acquisition, and preprocessing, laying the groundwork for a robust detection system.

The core focus shifts to code implementation, demonstrating the integration of TensorFlow and OpenCV for real-time detection. Each step, from data preprocessing to model training, is explained thoroughly. Beyond technical exploration, this journey celebrates human creativity meeting technological innovation. Enabling machines to understand gestures unlocks a future of richer human-computer interaction.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001

## ACKNOWLEDGMENT

We take this opportunity to record our sincere thanks to all who helped us to successfully complete this work. Firstly, We are grateful to our **supervisor Mr. Aaditya Maheswari** for his invaluable guidance and constant encouragement, support and most importantly for giving us the opportunity to carry out this work.

We would like to express our deepest sense of gratitude and humble regards to our

**Head of Department Dr. Rimpy Bishnoi** for giving invariable encouragement in our endeavors and providing necessary facilities for the same. Also a sincere thanks to all faculty members of CSE, TINJRIT for their help in the project directly or indirectly.

Finally, We would like to thank my friends for their support and discussions that have proved very valuable for us. We are indebted to our parents for providing constant support, love and encouragement. We thank them for the sacrifices they made so that we could grow up in a learning environment. They have always stood by us in everything we have done, providing constant support, encouragement and love

#### Nikhil Mail (20ETCCS079)

**Akshi Jain (20ETCCS006) Charvi Gokhru (20ETCCS020)**

# Abstract

The central objective of this project is to conceptualize, design, and implement a robust real-time hand gesture recognition system using TensorFlow and OpenCV, with a primary emphasis on achieving high accuracy, minimizing latency, and ensuring versatility for a myriad of applications, including human-computer interaction and sign language interpretation. Grounded in a rigorous and meticulous approach, the project encompasses a comprehensive array of tasks, spanning data preparation, model development, OpenCV integration, user interface design, performance optimization, thorough testing, meticulous documentation, and potential avenues for future enhancements.

At its essence, the system is meticulously crafted to provide an intuitive and user- friendly interface for real-time hand gesture recognition. Leveraging the formidable capabilities of TensorFlow for model training and OpenCV for real-time video processing, the project endeavors to accurately recognize and interpret hand gestures in real-time, thereby facilitating seamless communication for individuals with hearing impairments. By seamlessly integrating TensorFlow and OpenCV, the system aims to empower users with the ability to interact with digital interfaces and express themselves through sign language with unparalleled precision and efficacy. Ultimately, the project aspires to serve as a catalyst for positive societal change, fostering inclusivity, accessibility, and empowerment for individuals with hearing impairments, and paving the way towards a more inclusive and equitable future.

# CONTENTS

|  |  |
| --- | --- |
| **Topic** | **Page No** |
| **Certificate** | II |
| **Examiner Certificate** | V |
| **Preface** | VIII |
| **Acknowledgment** | IX |
| **Abstract** | X |
| **Contents** | XI |
| **List of Figures** | XIII |
| **INTRODUCTION** | 01 |
| **CHAPTER - 1 Brief of Machine Learning** | 02 |
| 1.1 Introduction | 02 |
| 1.2 Types of Machine Learning | 02 |
| 1.3 Data | 03 |
| 1.4 Algorithms | 03 |
| 1.5 Models | 03 |
| 1.6 Neural Networks | 03 |
| 1.7 Training | 06 |
| 1.8 Tensorflow | 07 |
| 1.9 OpenCV | 08 |
| 1.10 Use Case | 09 |
| 1.11 Future Scope | 09 |
| **CHAPTER - 2 WORKING** | 12 |
| 2.1 Introduction | 12 |
| 2.2 Media Pipeline | 12 |

|  |  |
| --- | --- |
| 2.3 Pre-requisites | 13 |
| 2.4 Hardware’s and Software’s | 13 |
| 2.5 Appendices | 13 |
| 2.6 Working | 16 |
| 2.7 Summary | 25 |
| **CONCLUSION** | 26 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| 1.1 | Neural-Network Architecture | 04 |
| 2.1 | Media Pipe Key Points | 12 |
| 2.2 | Live Long | 18 |
| 2.3 | Okay | 18 |
| 2.4 | Thumbs Up | 19 |
| 2.5 | Fist | 19 |
| 2.6 | Rock | 20 |
| 2.7 | Stop | 20 |
| 2.8 | Thumbs Down | 21 |
| 2.9 | Call Me | 21 |
| 2.10 | Peace | 22 |
| 2.11 | Smile | 22 |

# INTRODUCTION

In a world where seamless communication is essential, individuals with hearing impairments often face challenges in expressing themselves effectively through traditional means. Sign language serves as a primary mode of communication for many in this community, yet barriers persist in bridging the gap between sign language and spoken language. To address this issue, the development of a real-time sign language interpretation system holds immense promise, offering a pathway to facilitate communication and enhance inclusivity.

The objective of this project is to design a sophisticated system for real-time sign language interpretation using a webcam, with a focus on accuracy, low-latency processing, and versatility for various applications, including human-computer interaction and sign language interpretation. Leveraging the capabilities of TensorFlow and OpenCV, the system aims to recognize hand gestures in real-time, enabling seamless communication for individuals with hearing impairments.

The scope of this project encompasses a comprehensive range of tasks, including data preparation, model development, OpenCV integration, user-friendly interface design, performance optimization, testing, documentation, and potential future enhancements. By addressing each aspect of the development process, we aim to create a robust and user- friendly solution that meets the diverse needs of its users.

At its core, this application is designed to leverage the power of TensorFlow for model training and OpenCV for real-time video processing. Through a combination of advanced machine learning algorithms and efficient image processing techniques, the system endeavors to accurately interpret hand gestures in real-time, providing users with an intuitive interface for seamless interaction.

In summary, the development of a real-time sign language interpretation system represents a significant step towards fostering inclusivity and empowering individuals with hearing impairments to communicate effectively in a rapidly evolving digital world. By harnessing the capabilities of TensorFlow and OpenCV, this project seeks to make meaningful strides towards breaking down barriers and creating a more inclusive society.

## CHAPTER-1: BRIEF OF MACHINE LEARNING

### Introduction

Machine learning and neural networks represent two cornerstones of modern artificial intelligence, revolutionizing various industries and domains. This report provides an in-depth exploration of these concepts, elucidating their principles, applications, and implications.

### Types of Machine Learning

#### Supervised Learning

* + - * In supervised learning, the algorithm is trained on a dataset consisting of input-output pairs, where the input is usually referred to as features and the output as labels or targets.
			* Common supervised learning tasks include regression, where the goal is to predict a continuous output variable, and classification, where the goal is to predict a discrete output category.
			* Algorithms used in supervised learning include linear regression, logistic regression, decision trees, support vector machines (SVM), and various types of neural networks.

### Unsupervised Learning

* Unsupervised learning involves training the algorithm on data without explicit labels or targets. The objective is to uncover hidden patterns, structures, or relationships within the data.
* Clustering is a common unsupervised learning task where the algorithm groups similar data points together into clusters based on their inherent properties.
* Dimensionality reduction techniques, such as principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE), aim to reduce the dimensionality of the data while preserving its essential characteristics.

### Reinforcement Learning

* Reinforcement learning is a learning paradigm where an agent learns to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties.
* The agent's goal is to learn a policy, which maps states of the environment to actions, in order to maximize cumulative rewards over time.
* Algorithms used in reinforcement learning include Q-learning, deep Q- networks (DQN), policy gradients, and actor-critic methods.

### Data

Machine learning algorithms rely on large volumes of data to learn patterns and make predictions. Datasets are typically divided into training, validation, and test sets to train and evaluate the performance of machine learning models.

### Algorithms

Machine learning algorithms can be categorized into supervised, unsupervised, and reinforcement learning. Supervised learning involves training models on labeled data, while unsupervised learning aims to discover patterns and structure in unlabeled data. Reinforcement learning focuses on training agents to take actions in an environment to maximize cumulative rewards.

### Models

Machine learning models represent the learned relationships between input data and output predictions. Common types of machine learning models include linear regression, decision trees, support vector machines, and deep neural networks.

### Neural Networks

Neural networks, a subset of machine learning and the core of deep learning algorithms, mimic the brain's structure and function. They consist of layers of nodes: an input layer, one or more hidden layers, and an output layer. Each node processes input signals, applies a mathematical function, and produces an output. During training, neural networks adjust

connection weights and biases to minimize prediction errors using backpropagation. Optimization algorithms like stochastic gradient descent aid in this process. Neural networks are versatile, used in image recognition, natural language processing, speech recognition, financial analysis, and more. Deep learning, employing deep neural networks with multiple hidden layers, has excelled in tasks requiring complex pattern recognition and data representation. Overall, neural networks' ability to learn from data, make predictions, and perform various tasks makes them a cornerstone of modern AI and machine learning systems.



**Fig1.1** Neural-Network Architecture

### Neurons

Neurons form the foundational units of computation within neural networks. Each neuron undertakes a process that involves receiving input signals, applying a transformation through an activation function, and subsequently generating an output signal. This output is then relayed as input to neurons situated in the subsequent layer, thus propagating information through the network. The network structure typically encompasses several layers: Input Layer: Receives external data input, with each neuron representing specific features or attributes. Hidden Layers: These layers process data by applying weights to inputs and

utilizing activation functions to learn intricate patterns. Output Layer: Produces the network's final output, which could be predictions, classifications, or other desired results. The inter- neuron connections are characterized by weights and biases, which are continually adjusted during training using optimization algorithms like gradient descent. This adjustment aims to minimize the disparity between predicted and actual outcomes in training data. Activation functions, such as sigmoid, tanh, ReLU, and softmax, govern a neuron's output based on the weighted sum of inputs. They play critical roles in different parts of the network, contributing to its overall functionality. Information propagation in neural networks involves forward propagation, where input data is sequentially processed to yield an output, and backpropagation, wherein prediction errors guide parameter updates for enhanced accuracy in subsequent predictions. In essence, neurons in neural networks are pivotal in processing input data, acquiring insights, and producing meaningful outputs, making neural networks indispensable tools across various machine learning applications.

### Layers

Neural networks consist of multiple layers of neurons, each serving a specific function. Input layers receive input data, hidden layers perform computations, and output layers produce predictions or classifications. Deep neural networks, characterized by multiple hidden layers, are capable of learning complex representations from data.

### Activation Functions

Activation functions are pivotal components within neural networks, introducing non- linearities that enable models to grasp intricate patterns and relationships within data. By applying these functions, neural networks can capture complex dependencies that linear transformations alone cannot represent effectively. Commonly used activation functions include:

Sigmoid Function: It maps input values to a range between 0 and 1, providing a smooth transition from low to high values. However, sigmoid functions suffer from vanishing gradients, especially for extreme input values, which can slow down learning in deep networks.

Hyperbolic Tangent (tanh) Function: Similar to the sigmoid function, tanh also squashes input values to a range (-1, 1). It addresses the vanishing gradient problem to some extent but still experiences saturation for large inputs.

ReLU (Rectified Linear Unit): This activation function sets negative inputs to zero and leaves positive inputs unchanged. ReLU has gained popularity due to its simplicity and effectiveness in training deep neural networks. However, it may suffer from the "dying ReLU" problem, where neurons can become inactive during training and stop learning.

Softmax Function: Primarily used in the output layer for multi-class classification tasks, softmax converts raw scores into probabilities, ensuring that the sum of output probabilities is always 1. It is particularly useful for generating probability distributions over multiple classes.

Choosing the appropriate activation function depends on the specific task and network architecture. For instance, ReLU is commonly used in hidden layers to introduce non- linearities and accelerate convergence, while softmax is suitable for generating class probabilities in classification tasks.

In conclusion, activation functions play a crucial role in neural network computations by introducing non-linearities that enable models to capture complex relationships in data, contributing significantly to the success of deep learning techniques across various domains.

### Training

Training neural networks involves an iterative process of fine-tuning the weights and biases of connections between neurons to minimize a loss function. This loss function quantifies the difference between predicted and actual outputs, guiding the network towards better performance. Backpropagation is a key technique used in this process, where prediction errors are propagated backward through the network to update model parameters.

Optimization algorithms like gradient descent are often employed alongside backpropagation to efficiently adjust the model's parameters during training. Gradient descent calculates the direction and magnitude of parameter updates based on the gradient of the loss function, gradually moving towards the optimal set of parameters that minimize the loss.

The iterative nature of training allows neural networks to learn from data, adjust their internal representations, and improve predictive accuracy over time. This process of learning and refinement is fundamental to the success of neural networks in various machine learning tasks.

### Tensorflow

TensorFlow is an open-source machine learning framework developed by Google, designed to facilitate the development and deployment of machine learning models. Here's a more detailed overview of TensorFlow

#### High-level APIs

* + - * TensorFlow provides high-level APIs, such as Keras, tf.keras, and Estimators, that simplify the process of model development, training, and deployment.
			* These APIs offer a user-friendly interface for building and training neural network models, enabling rapid prototyping and experimentation.

#### Distributed Training

* + - * TensorFlow supports distributed training across multiple devices and machines, allowing users to scale their models to large datasets and compute clusters.
			* Distributed training enables faster convergence and improved performance on large-scale tasks, leveraging resources efficiently.

#### Model Serving and Deployment

* + - * TensorFlow provides tools and frameworks for model serving and deployment in production environments.
			* TensorFlow Serving enables the deployment of trained models for serving predictions over the network, while TensorFlow Lite facilitates deployment on mobile and embedded devices.

### OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source library for computer vision tasks, providing a wide range of functions for image and video processing. Here's a more detailed overview of OpenCV:

#### Image Processing

* + - * OpenCV offers a rich set of functions for image processing, including filtering, thresholding, edge detection, and morphological operations.
			* These functions enable users to preprocess images before feeding them into machine learning models or performing further analysis.

#### Object Detection

* + - * OpenCV provides tools for object detection and recognition, including Haar cascades, Histogram of Oriented Gradients (HOG), and deep learning-based methods such as Single Shot MultiBox Detector (SSD) and You Only Look Once (YOLO).
			* These algorithms enable users to detect objects and faces in images and videos, facilitating various computer vision applications.

#### Real-Time Video Processing

* + - * OpenCV supports real-time video processing, allowing users to capture video streams from webcams or other video sources and perform operations such as object tracking, motion detection, and facial recognition in real-time.
			* Real-time video processing enables the development of interactive applications and systems for tasks such as surveillance, augmented reality, and human-computer interaction.

### Use Cases

#### Image and Speech Recognition

Convolutional neural networks (CNNs) are widely used for image classification, object detection, and facial recognition. Recurrent neural networks (RNNs) and transformers are employed for speech recognition and natural language processing tasks.

#### Healthcare

Machine learning models are utilized for medical image analysis, disease diagnosis, personalized treatment recommendations, and drug discovery.

#### Recommendation Systems

Machine learning algorithms drive recommendation systems used by e-commerce platforms, streaming services, and social media platforms to personalize content and enhance user experience.

#### Autonomous Vehicles

Neural networks power perception systems in autonomous vehicles, enabling tasks such as object detection, lane detection, and path planning.

### Future Scope

#### Advanced Gesture Recognition

Expand the capabilities of the system to recognize a wider range of hand gestures with higher accuracy. This could involve training the model on a larger and more diverse dataset, implementing advanced neural network architectures, or incorporating additional features such as hand pose estimation.

#### Real-time Feedback and Interaction

Integrate real-time feedback mechanisms to provide users with immediate responses to their gestures. This could involve visual or auditory cues to confirm successful recognition of gestures or provide guidance for corrective actions.

#### Gesture Customization and Personalization

Allow users to customize and personalize the system to recognize their own unique set of gestures. Implement user profiles or settings that adapt the recognition system based on individual preferences and usage patterns.

#### Multi-modal Interaction

Explore the integration of multiple modalities, such as voice commands or facial expressions, to enhance the overall user experience and enable more natural and intuitive interactions with digital interfaces.

#### Application-specific Extensions

Develop specialized extensions or modules tailored for specific applications or domains. For example, integrate the hand gesture recognition system into gaming applications, virtual reality environments, or assistive technologies for individuals with disabilities.

#### Cross-platform Compatibility

Ensure compatibility and seamless integration with a variety of platforms and devices, including desktop computers, mobile devices, and embedded systems. This could involve optimizing the system for different hardware configurations and operating systems.

#### Continuous Learning and Improvement

Implement mechanisms for continuous learning and improvement of the gesture recognition system over time. This could involve incorporating feedback loops from users to refine the model, as well as leveraging techniques such as online learning to adapt to changing environments and user preferences.

## CHAPTER-2: WORKING

### 2.1. Introduction

Gesture recognition is an active research field in Human-Computer Interaction technology. It has many applications in virtual environment control and sign language translation, robot control, or music creation. In this machine learning project on Hand Gesture Recognition, we are going to make a real-time Hand Gesture Recognizer using the MediaPipe framework and Tensorflow in OpenCV and Python.

### 2.2 MediaPipe

MediaPipe is a customizable machine learning solutions framework developed by Google. It is an open-source and cross-platform framework, and it is very lightweight. MediaPipe comes with some pre-trained ML solutions such as face detection, pose estimation, hand recognition, object detection, etc.

We‟ll first use MediaPipe to recognize the hand and the hand key points. MediaPipe returns a total of 21 key points for each detected hand.



Media Pipe Key Points (**Fig 2.1)**

These key points will be fed into a pre-trained gesture recognizer network to recognize the hand pose.

### Pre-requisites

* + - Python – 3.x
		- OpenCV – 4.5
			* Run “pip install opencv-python” to install OpenCV.
		- MediaPipe – 0.8.5
			* Run “pip install mediapipe” to install MediaPipe.
		- Tensorflow – 2.5.0
			* Run “pip install tensorflow” to install the tensorflow module.
		- Numpy – 1.19.3

### Hardware & Software

* + - High-resolution camera (webcam, smartphone).
		- GPU (optional, for accelerated processing).
		- Powerful CPU and ample RAM (8GB+).
		- Operating system (Ubuntu).
		- Python with TensorFlow and OpenCV.
		- CUDA and cuDNN (NVIDIA GPU acceleration).
		- IDE (PyCharm, Visual Studio Code).

### Appendices

# import necessary packages import cv2

import numpy as np import mediapipe as mp import tensorflow as tf

from tensorflow.keras.models import load\_model

# initialize mediapipe mpHands = mp.solutions.hands

hands = mpHands.Hands(max\_num\_hands=1, min\_detection\_confidence=0.7)

mpDraw = mp.solutions.drawing\_utils

# Load the gesture recognizer model model = load\_model('mp\_hand\_gesture')

# Load class names

f = open('gesture.names', 'r') classNames = f.read().split('\n') f.close()

print(classNames)

# Initialize the webcam cap = cv2.VideoCapture(0)

while True:

# Read each frame from the webcam

\_, frame = cap.read() x, y, c = frame.shape

# Flip the frame vertically frame = cv2.flip(frame, 1)

framergb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

# Get hand landmark prediction result = hands.process(framergb) className = ''

# post process the result

if result.multi\_hand\_landmarks: landmarks = []

for handslms in result.multi\_hand\_landmarks: for lm in handslms.landmark:

lmx = int(lm.x \* x) lmy = int(lm.y \* y)

landmarks.append([lmx, lmy])

# Drawing landmarks on frames mpDraw.draw\_landmarks(frame, handslms,

mpHands.HAND\_CONNECTIONS)

# Predict gesture

prediction = model.predict([landmarks]) # print(prediction)

classID = np.argmax(prediction) className = classNames[classID]

# show the prediction on the frame

cv2.putText(frame, className, (10, 50), cv2.FONT\_HERSHEY\_SIMPLEX,

1, (0,0,255), 2, cv2.LINE\_AA)

# Show the final output cv2.imshow("Output", frame)

if cv2.waitKey(1) == ord('q'): break

# release the webcam and destroy all active windows cap.release()

cv2.destroyAllWindows()

### Working

#### Import necessary packages

To build this Hand Gesture Recognition project, we‟ll need four packages. So first import these.

**# import necessary packages for hand gesture recognition project using Python OpenCV**

**import cv2**

**import numpy as np import mediapipe as mp import tensorflow as tf**

**from tensorflow.keras.models import load\_model**

### Initialize models

**# initialize mediapipe mpHands = mp.solutions.hands**

**hands = mpHands.Hands(max\_num\_hands=1, min\_detection\_confidence=0.7) mpDraw = mp.solutions.drawing\_utils**

Mp.solution.hands module performs the hand recognition algorithm. So we create the object and store it in mpHands.

Using mpHands.Hands method we configured the model. The first argument is max\_num\_hands, that means the maximum number of hand will be detected by the model in a single frame. MediaPipe can detect multiple hands in a single frame, but we‟ll detect only one hand at a time in this project.

Mp.solutions.drawing\_utils will draw the detected key points for us so that we don‟t have to draw them manually.

**# Load the gesture recognizer model model = load\_model('mp\_hand\_gesture') # Load class names**

**f = open('gesture.names', 'r') classNames = f.read().split('\n') f.close()**

**print(classNames)**

* Using the load\_model function we load the TensorFlow pre-trained model.
* Gesture.names file contains the name of the gesture classes. So first we open the file using python‟s inbuilt open function and then read the file.
* After that, we read the file using the read() function.

#### Output

[„okay‟, „peace‟, „thumbs up‟, „thumbs down‟, „call me‟, „stop‟, „rock‟, „live long‟, „fist‟,

„smile‟]

The model can recognize 10 different gestures.



### Live Long (Fig. 2.2)



**Okay (Fig. 2.3)**



### Thumbs Up (Fig. 2.4)



**Fist (Fig. 2.5)**



### Rock (Fig. 2.6)



**Stop (Fig. 2.7)**



### Thumbs Down (Fig. 2.8)



**Call Me (Fig. 2.9)**



### Peace (Fig. 2.10)



**Smile (Fig. 2.11)**

### Read frames from a webcam

# Initialize the webcam for Hand Gesture Recognition Python project cap = cv2.VideoCapture(0)

while True:

# Read each frame from the webcam

\_, frame = cap.read()

``x , y, c = frame.shape

# Flip the frame vertically frame = cv2.flip(frame, 1) # Show the final output cv2.imshow("Output", frame)

if cv2.waitKey(1) == ord('q'):

break

# release the webcam and destroy all active windows cap.release()

cv2.destroyAllWindows()

* + We create a VideoCapture object and pass an argument „0‟. It is the camera ID of the system. In this case, we have 1 webcam connected with the system. If you have multiple webcams then change the argument according to your camera ID. Otherwise, leave it default.
	+ The cap.read() function reads each frame from the webcam.
	+ cv2.flip() function flips the frame.
	+ cv2.imshow() shows frame on a new openCV window.
	+ The cv2.waitKey() function keeps the window open until the key „q‟ is pressed.

### Detect hand keypoints

* + MediaPipe works with RGB images but OpenCV reads images in BGR format. So, using cv2.cvtCOLOR() function we convert the frame to RGB format.
	+ The process function takes an RGB frame and returns a result class.
	+ Then we check if any hand is detected or not, using result.multi\_hand\_landmarks method.
	+ After that, we loop through each detection and store the coordinate on a list called landmarks.
	+ Here image height (y) and image width(x) are multiplied with the result because the model returns a normalized result. This means each value in the result is between 0 and 1.
	+ And finally using mpDraw.draw\_landmarks() function we draw all the landmarks in the frame.

### 2.6.5 Recognize hand gestures

**# Predict gesture in Hand Gesture Recognition project prediction = model.predict([landmarks])**

**print(prediction)**

**classID = np.argmax(prediction) className = classNames[classID]**

**# show the prediction on the frame**

**cv2.putText(frame, className, (10, 50), cv2.FONT\_HERSHEY\_SIMPLEX,**

**1, (0,0,255), 2, cv2.LINE\_AA)**

* The model.predict() function takes a list of landmarks and returns an array contains 10 prediction classes for each landmark.

The output looks like this-

[[2.0691623e-18 1.9585415e-27 9.9990010e-01 9.7559416e-05

1.6617223e-06 1.0814080e-18 1.1070732e-27 4.4744065e-16 6.6466129e-07

4.9615162e-21]]

* Np.argmax() returns the index of the maximum value in the list.
* After getting the index we can simply take the class name from the classNames list.
* Then using the cv2.putText function we show the detected gesture into the frame.

### Summary

In summary, the Hand Gesture Recognition project involved building a hand gesture recognizer using OpenCV and Python. The project utilized MediaPipe for hand detection and TensorFlow for gesture recognition, highlighting the integration of computer vision and deep learning technologies. Key aspects of the project included learning about neural networks, file handling, and common image processing techniques. By combining these elements, we developed a robust system capable of accurately detecting and interpreting hand gestures, showcasing the practical application of AI and machine learning in real-world projects.

## CONCLUSION

Real-time Hand Gesture Recognition using TensorFlow and OpenCV represents a significant advancement in human-computer interaction, blending deep learning and computer vision technologies. By integrating TensorFlow's machine learning capabilities and OpenCV's image processing functionalities, this approach enables intuitive and immersive computing experiences, fostering a more interconnected digital future.

Neural networks epitomize the convergence of biological insight and technological innovation, unlocking unprecedented potential across diverse domains. Their adaptive learning mechanisms and multi-layered architecture facilitate complex pattern recognition and classification. As neural networks continue to evolve, they promise to revolutionize industries and drive societal transformation, guiding humanity towards a future illuminated by intelligent exploration.