Zagdu Singh Charitable Trust's (Regd.)
THAKUR COLLEGE OF ENGINEERING & TECHNOLOGY
Autonomous College Affiliated to University of Mumbai
Approved by All India Council for Technical Education(AICTE) and Government of Maharashtra(GoM)
Conferred Autonomous Status by University Grants Commission (UGC) for 10 years w.e.f. A.Y 2019-20
Amongst Top 200 Colleges in the Country, Ranked 193rd in NIRF India Ranking 2019 in Engineering College category
• ISO 9001:2015 Certified • Programmes Accredited by National Board of Accreditation (NBA), New Delhi
• Institute Accredited by National Assessment and Accreditation Council (NAAC), Bangalore
Website : www.tcetmumbai.in
Estd. in 2001

A.Y 2020-21
Institutional Summer Internship

Track: Machine Learning Using Python

Project Report on
HAND GESTURE RECOGNITION AND CURSOR CONTROL & PAINT

Team Member Details:

Aniket Vishwakarma 50 SE COMP C
Riya Vishwakarma 53 SE COMP C
Neeraj Kumar Yadav 60 SE COMP C
Ujjwal Yadav 63 SE COMP C

Vikas Singh
Assistant Professor
Department of Computer Engineering

TOPIC INDEX

LIST OF FIGURES

LIST OF TABLES

Problem Description

It has been generations since we have been using hand gestures for communicating in human society. The shaking of hands, thumbs up and Thumbs down signs have been ever existing in the environment. It is believed that gestures are the easiest way of interaction with anyone. So then why not apply it to the machines that we are using. In this work, we are demonstrating, real- gesture. The initial setup includes a low-cost USB web camera that can be used for providing the input to the system. The complete process is divided into 4 steps which are frame-capturing, image-processing, region extraction, feature-matching. To the extreme, it can also be called as hardware because it uses a camera for tracking hands.

Aim and objective of research work include-

- For most laptop touchpad is not the most comfortable and convenient.
- Main objective pre-processing is to represent the data in such a way that it can be easily interpreted and processed by the system.
- Reduce cost of hardware.

It focuses on extracting the features over the human hands and then matching their features to recognize the movement of the hand.

Project essential feature-

- User friendly.
- Portable.
- Handle simple operation left- click dragging, minimizing.
- No hardware

The existing system consists of a mouse that can be either wireless or wired to control the cursor, know we can use hand gestures to monitor the system. The existing virtual mouse control system consists of the simple mouse operation using the colored tips for detection which are captured by web-cam, hence colored fingers acts as an object which the web-cam sense color like red, green, blue color to monitor the system, whereas could perform basic mouse operation like minimize, drag, scroll up , scroll down , left-click right-click using hand gestures without any colored finger because skin color recognition system is more flexible than the existing system.

In the existing system use static hand recognition like fingertip identification, hand shape, Number of fingers to defined action explicitly , which makes a system more complex to understand and difficult to use.

Literature Survey

Paper 1: "**A brief introduction to OpenCV**"

Summary: the paper is about computer vision to implement the vison to the computer

Paper 2: "Accurate, Robust, and Flexible Real-time Hand Tracking"

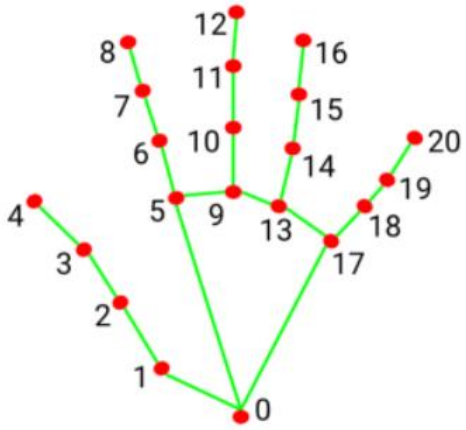
Summary: the the paper is all about the real time hand hand tracking in 3D model

Paper 3: "Human hand gesture based system for mouse cursor control
"

Summary: this paper presents the mouse as our touchpad it detect the image and use background you have to use a background then it detect the image position and then perform the cursor control

Dataset Description

Media pipe is a framework used to build Machine Learning Pipelines. It works on many different solutions like **Face Detection, Hands, Object Detection, Hollister Mic, Fac Poses,**, etc.
Media Pipe Hand is a machine-learning employed high-fidelity hand and finger tracking solution. It detects 21 Landmark points as shown in Fig. are recorded from a hand in a single frame with the help of multiple models which are working simultaneously.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Mediapipe Hands consists of two different models working together namely Palm Detection Model in which a full image is identified and it draws a box around the hand, and Hand Landmark Model operates on this boxed image formed by Palm Detector and provides high fidelity 2D hand keypoint coordinates. (As shown in above fig.)

We have trained 26 Alphabets using the corresponding hand gestures as shown in Fig. Around 500 to 1000 labels are recorded using the webcam for each gesture label alphabet for training the media pipe model.

Data Preprocessing

1. To collect data press key “k” while running the ***app.py file*** which switches to listening mode as displayed in the figure below.
2. Then by pressing keys from 0 to 9, we can load each gesture for the hand gesture label.
3. After Finding coordinates, Keypoint Classifier undergoes 4 steps of Data Preprocessing steps: Land Mark Coordinates, relative coordinate conversion, Flattening to a 1-D array, and Normalized values.
4. Then the key points will be then added to “model/keypoint_classifier/keypoint.csv” as shown below.

1	0	0	0	-0.12605	-0.36975	-0.04202	-0.68908	0.084034	-0.88235	0.184874	-1	0.462185	-0.77311	0.151261	-0.78992	-0.0084	-0.63866	0	-0.58824	0.
2	0	0	0	-0.12389	-0.38053	-0.09735	-0.70796	-0.0177	-0.90265	0.070796	-1	0.40708	-0.80531	0.053097	-0.81416	-0.12389	-0.67257	-0.11504	-0.61947	0.
3	0	0	0	-0.14407	-0.36441	-0.09322	-0.69492	0.016949	-0.89831	0.127119	-1	0.40678	-0.80508	0.016949	-0.77966	-0.11017	-0.61017	-0.05932	-0.55932	0.
4	0	0	0	-0.15126	-0.36134	-0.10084	-0.68067	0	-0.88235	0.109244	-1	0.420168	-0.7479	0.016807	-0.76471	-0.12605	-0.60504	-0.08403	-0.54622	0.
5	0	0	0	-0.14516	-0.34677	-0.09677	-0.67742	0.008065	-0.87097	0.104839	-1	0.379032	-0.78226	0.048387	-0.79032	-0.09677	-0.62903	-0.06452	-0.57258	
6	0	0	0	-0.1453	-0.35897	-0.11111	-0.68376	-0.02564	-0.88034	0.068376	-1	0.358974	-0.81197	-0.01709	-0.78632	-0.1453	-0.62393	-0.11966	-0.57265	0.
7	0	0	0	-0.14545	-0.38182	-0.12727	-0.72727	-0.06364	-0.91818	0	-1	0.390909	-0.82727	0.027273	-0.88182	-0.14545	-0.71818	-0.16364	-0.63636	0.
8	0	0	0	-0.15385	-0.35043	-0.12821	-0.68376	-0.03419	-0.88889	0.068376	-1	0.358974	-0.79487	-0.01709	-0.77778	-0.16239	-0.60684	-0.12821	-0.55556	0.
9	0	0	0	-0.13913	-0.37391	-0.10435	-0.68696	-0.01739	-0.88696	0.06087	-1	0.4	-0.76522	0.008696	-0.77391	-0.13913	-0.61739	-0.11304	-0.54783	0.
10	0	0	0	-0.15254	-0.36441	-0.11864	-0.68644	-0.02542	-0.88983	0.059322	-1	0.372881	-0.78814	-0.05932	-0.77119	-0.16949	-0.59322	-0.09322	-0.54237	0.
11	0	0	0	-0.15	-0.36667	-0.11667	-0.68333	-0.025	-0.89167	0.066667	-1	0.391667	-0.775	0.016667	-0.80833	-0.13333	-0.63333	-0.09167	-0.56667	0.
12	0	0	0	-0.13821	-0.36585	-0.11382	-0.68293	-0.03252	-0.88618	0.04878	-1	0.357724	-0.77236	-0.01626	-0.79675	-0.14634	-0.64228	-0.11382	-0.56911	0.
13	0	0	0	-0.1453	-0.36752	-0.10256	-0.68376	-0.01709	-0.88034	0.059829	-1	0.393162	-0.76923	-0.01709	-0.78632	-0.15385	-0.62393	-0.11111	-0.57265	0.
14	0	0	0	-0.13793	-0.38793	-0.11207	-0.72414	-0.0431	-0.92241	0.034483	-1	0.387931	-0.7931	-0.02586	-0.83621	-0.16379	-0.67241	-0.12069	-0.59483	0.
15	0	0	0	-0.14655	-0.36207	-0.08621	-0.68103	0.034483	-0.87931	0.137931	-1	0.413793	-0.7931	0.043103	-0.75862	-0.10345	-0.59483	-0.09483	-0.5431	0.
16	0	0	0	-0.12931	-0.40517	-0.08621	-0.73276	0.008621	-0.92241	0.077586	-1	0.396552	-0.80172	0	-0.81897	-0.14655	-0.63793	-0.09483	-0.56897	0.
17	0	0	0	-0.15385	-0.35897	-0.11966	-0.69231	-0.01709	-0.88889	0.076923	-1	0.376068	-0.80342	0.017094	-0.82051	-0.12821	-0.64957	-0.10256	-0.5812	0.

5. **Dataset Columns**:-1st column denotes pressed number (used as class ID), 2nd and subsequent columns-Keypoint coordinates
6. After completion of these steps, 10 labels are created each time by pressing keys b/w 0 to 9.
7. As we are training 26 Alphabets we required to record and save all 26 labels. For this we undergo, 10 labels + 10 labels+6 labels respectively.
8. Now, we merge these separate files into a single file.

Choice of Model

Palm Detection Model

To detect initial hand locations, we designed a single-shot detector model optimized for mobile real-time uses in a manner similar to the face detection model in MediaPipe Face Mesh. Detecting hands is a decidedly complex task: our model has to work across a variety of hand sizes with a large scale span (~20x) relative to the image frame and be able to detect occluded and self-occluded hands. Whereas faces have high contrast patterns, e.g., in the eye and mouth region, the lack of such features in hands makes it comparatively difficult to detect them reliably from their visual features alone. Instead, providing additional context, like arm, body, or person features, aids accurate hand localization.

Our method addresses the above challenges using different strategies. First, we train a palm detector instead of a hand detector, since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. In addition, as palms are smaller objects, the non-maximum suppression algorithm works well even for two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using square bounding boxes (anchors in ML terminology) ignoring other aspect ratios, and therefore reducing the number of anchors by a factor of 3-5. Second, an encoder-decoder feature extractor is used for bigger scene context awareness even for small objects (similar to the RetinaNet approach). Lastly, we minimize the focal loss during training to support a large amount of anchors resulting from the high scale variance.

With the above techniques, we achieve an average precision of 95.7% in palm detection. Using a regular cross entropy loss and no decoder gives a baseline of just 86.22%.

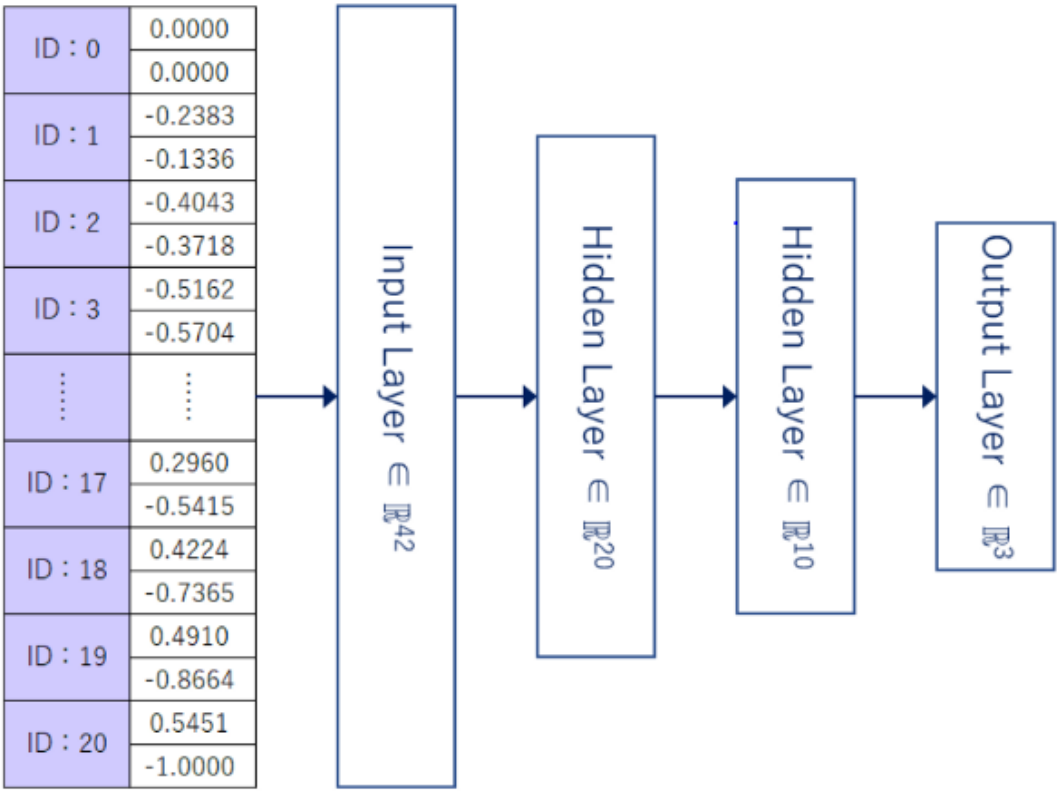
Hand Landmark Model

After the palm detection over the whole image our subsequent hand landmark model performs precise keypoint localization of 21 3D hand-knuckle coordinates inside the detected hand regions via regression, that is direct coordinate prediction. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions.

To obtain ground truth data, we have manually annotated ~30K real-world images with 21 3D coordinates, as shown below (we take Z-value from image depth map, if it exists per corresponding coordinate). To better cover the possible hand poses and provide additional supervision on the nature of hand geometry, we also render a high-quality synthetic hand model over various backgrounds and map it to the corresponding 3D coordinates.

Model training and testing

The model structure for training the key points can be found in "alpha_train.ipynb" in Jupyter Notebook and execute from top to bottom. We used 75% of the data for training and the rest 25% is allocated for testing purpose.
The image of the model prepared in "alpha_train.ipynb" is as follows.



```
#Model Initialization
model = tf.keras.models.Sequential([
    tf.keras.layers.Input((21 * 2, )),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(40, activation='relu'),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(30, activation='relu'),
    tf.keras.layers.Dense(NUM_CLASSES, activation='softmax')
])

#Compiling the Model

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])

#Model Fitting

model.fit(
    X_train,
    y_train,
    epochs=1000,
    batch_size=128,
    validation_data=(X_test, y_test),
    callbacks=[cp_callback, es_callback]
)
```

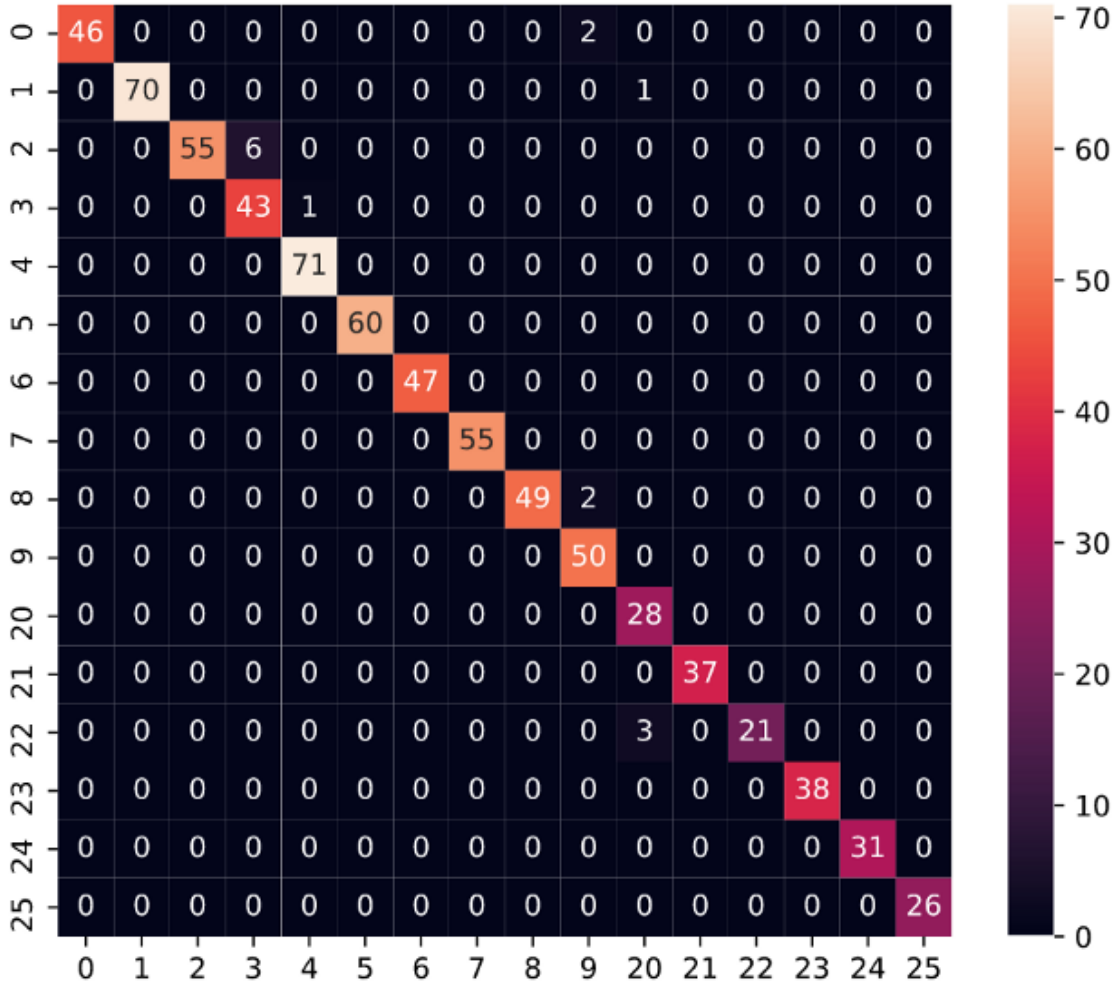
The Tensorflow model trained using the following architecture (above fig. Model Architecture) is saved in the HDF5 file, converted to the TensorFlow Lite model. This Tensorflow Lite model that stores the model architecture and weights is used to classify hand gestures when the keypoint classifier function is called from app.py.

After running the model, we achieved an accuracy of 80.33% and loss of 0.5323 for training. For Validation, accuracy of 97.9% and loss of 0.1646.

Model Testing

We use inference_only.py for testing the model. From the classification report, it is observed that 98% accuracy is achieved for the test dataset.

i. Confusion Matrix



ii. Classification Report

Classification Report

	precision	recall	f1-score	support
0	1.00	0.96	0.98	48
1	1.00	0.99	0.99	71
2	1.00	0.90	0.95	61
3	0.88	0.98	0.92	44
4	0.99	1.00	0.99	71
5	1.00	1.00	1.00	60
6	1.00	1.00	1.00	47
7	1.00	1.00	1.00	55
8	1.00	0.96	0.98	51
9	0.93	1.00	0.96	50
20	0.88	1.00	0.93	28
21	1.00	1.00	1.00	37
22	1.00	0.88	0.93	24
23	1.00	1.00	1.00	38
24	1.00	1.00	1.00	31
25	1.00	1.00	1.00	26
accuracy			0.98	742
macro avg	0.98	0.98	0.98	742
weighted avg	0.98	0.98	0.98	742

The following steps are included to develop the algorithm:-

1. The first step is to capture the image using the camera.
2. The camera then extracts and recognizes the human hand from the input image.
3. Find the landmarks
4. Get the tip of the index and middle finger
5. Check which fingers are up
6. Only Index finder : it means it in in moving mode
7. Convert Coordinates:

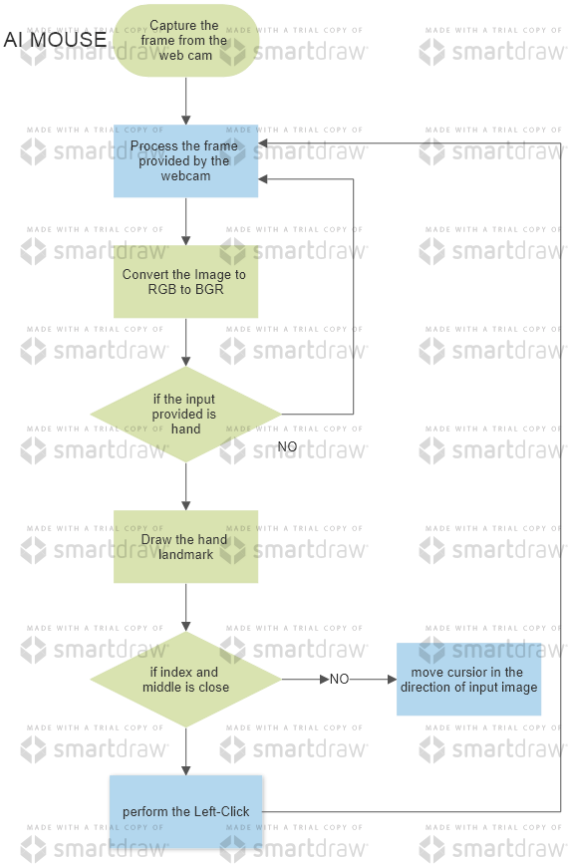
Because resolution of each computer machine is different
We need to convert those coordinates to computer resolution

8. Smoothen Values
9. Move Mouse
10. Check Whether we are in clicking mode

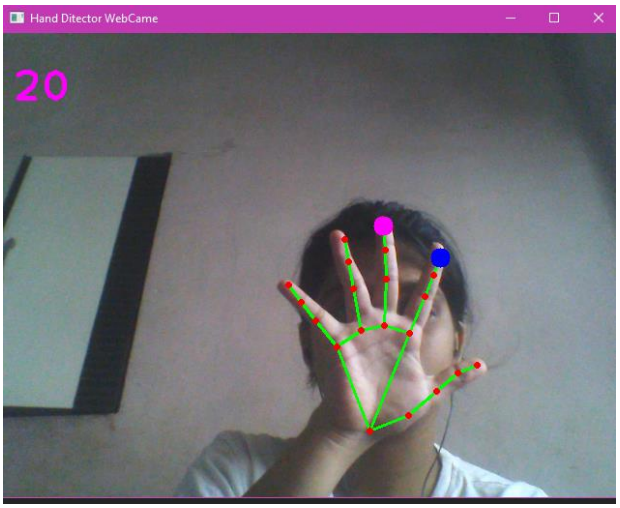
if index finger and middle finger are up

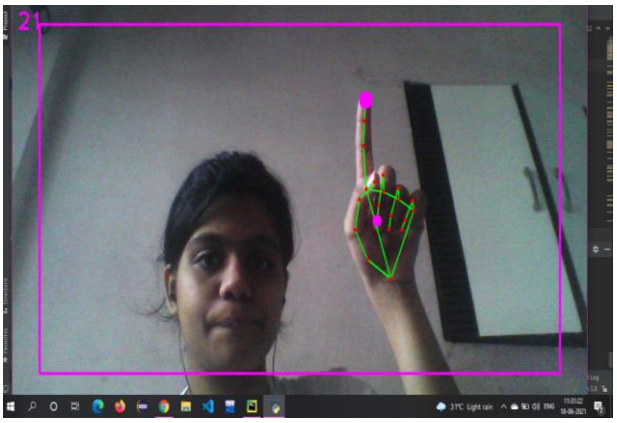
11. Find distance between index finger and middle.
12. if distance is less than a limited distance, Then perform click function

Flow chart


AI MOUSE
Capture the frame from the web cam
Process the frame provided by the webcam
Convert the Image to RGB to BGR
if the input provided is hand
NO
Draw the hand landmark
if index and middle is close
NO
move cursior in the direction of input image
perform the Left-Click

Result analysis and Discussion





During this project we have successfully completed the model training tasting training our model to perform the certain mouse task so we are getting up around almost of 92% of accuracy .

Feature	**Existing System**	**Proposed System**
Stability	The Existing System is poor on stability front	The proposed system is very stable as compared to its predecessor
environment	The existing system is highly dependent on the environment in which the system is being used	The proposed system has a very les dependency on the environmental factors.
Complexity	The existing system are very complex and require powerful processors	The proposed system is very simple and requires very basic processors
Practicality	The existing system in not a practically viable option in the real world	The proposed system is a practically viable option in the real world.
Future Perspective	The existing system does not integrate well with the existing technology	The existing system can integrate well with the other technologies

Conclusion and Future Scope

The motive of this project was to make the machine more interactive and responsive towards human behavior. The sole aim of this project was to make a technology that is affordable and portable with any standard operating system.

The proposed system is used to control the pointer of the mouse by detecting the human hand and moving the pointer in the direction in the human hand respectively. the system Control simple function of the mouse such as left-clicking, dragging and cursor movement.

The method detects the finger and palm at 21 point and tracks it continuously for the movement of the cursor when the distance between the index fingers and middle finger of the human hand is less closer to each other the process performs the task of the left-click.

References

[1] A. Manjula1, A. Rama Mohan Reddy, “Sentiment Analysis on Social media”, International Journal of Computer Engineering In Research Trends, 6(11):pp1-6, November 2019.
https://github.com/kartikrawool/sentiment_analysis/blob/master/Final_Notebook.ipynb

1. https://google.github.io/mediapipe/solutions/hands.html
2. https://paperswithcode.com/paper/first-person-hand-action-benchmark-with-rgb-d
3.] Facebook. Oculus Quest Hand Tracking. https://www.oculus.com/blog/oculusconnect-6-introducing-hand-tracking-onoculus-quest-facebook-horizon-and-more/. 1
4. Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. Robust 3d hand pose estimation from single depth images using multi-view cnns. IEEE Transactions on Image Processing, 27(9):4422–4436, 2018. 1 [5] Liuhao Ge, Zhou Ren, Yunchen
5. Google. Tensorflow.js Handpose. https://blog.tensorflow.org/2020/03/ face-and-hand-tracking-in-browser-withmediapipe-and-tensorflowjs.html. 1
6. Google. Tensorflow lite on GPU. https://www.tensorflow.org/lite/ performance/gpu advanced