VIRTUAL MOUSE AI USING HAND GESTURE

Gogineni Ashrith Sai
Department of Computer Science and Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham
India
ashrith.gogineni@gmail.com

Konda Reddy Balaji Reddy
Department of Computer Science and Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham
India
balajireddykondareddy@gmail.com

Thanakanti Ganesh Madhav
Department of Computer Science and Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham
India
madhavchickoo@gmail.com

Dr. Suja Palaniswamy *
Department of Computer Science and Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham
India
p_suja@blr.amrita.edu

Dr. Peeta Basa Pati*
Department of Computer Science and Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham
India
bp_peeta@blr.amrita.edu

Abstract: A hand gesture-controlled virtual mouse is a modern approach to computer interaction that allows users to manipulate their computers using hand gestures instead of physical mouse. This technology uses a blend of CV and ML algorithms to detect and trace the person's hand movements in real-time usage. The system translates these movements into commands that move the virtual mouse cursor on the computer screen. This technology has numerous applications in various fields, including gaming, virtual reality, and industrial automation. A continuation of this project can be taken to be as an application of hand gesture recognition using Mediapipe and deep learning techniques, with the aim of helping the speech impaired and deaf people to easily understand sign languages. This research focuses on developing an accurate and efficient hand gesture recognition model that enables seamless communication for individuals with hearing and speech disabilities. A diverse dataset of hand gestures was utilized, and preprocessing techniques were employed to enhance the data quality. By utilizing MediaPipe's robust capabilities, a real-time convolutional neural network (CNN) model was implemented for classifying hand gestures. The system exhibited promising results for a variety of gestures, showcasing the potential of MediaPipe in facilitating communication for the target group. This study contributes to the expanding field of assistive technology, offering practical and accessible solutions to bridge communication gaps for individuals with hearing and speech impairments.

Keywords: Hand Gesture Detection, Computer Vision, Mediapipe, TensorFlow, Matplotlib, Scikit-learn, LSTM

I. INTRODUCTION

This paper presents a thorough investigation into the development and deployment of a virtual mouse system controlled by hand gestures. The objective of the proposed system is to utilize computer vision techniques and machine learning algorithms to accurately recognize and interpret hand gestures, thereby enabling control over computer systems without the need for an optical or touchpad mouse. The system employs Python as the coding language and leverages webcam live capturing along with OpenCV for hand movement tracking and gesture recognition. By employing preprocessing techniques such as differentiating background and skin regions using finger-tip points and regions of interest (ROI), precise hand movements are achieved. The system supports various common gestures including pointing, clicking, dragging, and scrolling. Furthermore, the potential for expanding the system's capabilities to include a virtual keyboard and voice assistant is explored. Evaluation of the system's performance involves assessing metrics such as accuracy, response time, and user satisfaction, providing a comprehensive analysis of its effectiveness and usability, and as an extension to this, the hand gesture recognition aims to address the unique communication challenges faced by individuals who are deaf and speech-impaired by developing a real-time hand gesture recognition system using MediaPipe, a versatile framework for perceptual computing applications. By accurately interpreting hand gestures, this system seeks to provide an intuitive and natural means of communication for individuals with hearing and speech disabilities, enabling them to express themselves more effectively and engage with their environment.

Advancements in computer vision and machine learning have significantly contributed to the feasibility and practicality of hand gesture recognition. Through the application of advanced algorithms and deep learning techniques, it is now possible to detect and classify hand gestures in real-time with a high degree of accuracy. MediaPipe, developed by Google,

offers a comprehensive suite of tools and pre-built models that facilitate the development of hand gesture recognition systems.

The primary motivation for this project stems from the potential of hand gesture recognition technology to improve communication and enhance the overall quality of life for individuals who are deaf and speech impaired. By providing a more efficient and intuitive mode of interaction, hand gesture recognition empowers these individuals to express themselves confidently, effectively convey their needs, and actively participate in various social and professional contexts.

II. Literature Survey

The development of hand gesture recognition technology has opened new avenues for human computer interaction. This section presents a literature review of related works and existing systems in the field of hand gestures controlled virtual mouse.

Bharath Kumar Reddy Sandra, et al. used virtual mouse control system using depth camera and ML Algorithm for gesture recognition, with high accuracy and usability demonstrated.[1]

Prof Girish B, Arvind P , et al. used webcam-based system that processes hand gesture images with OpenCV and Keras libraries to classify mouse operations, showing effectiveness in controlling a virtual mouse. [2]

Abhilasha, Vandana T, et al. Raspberry Pi-based system for controlling a virtual mouse using hand gesture recognition, processed with OpenCV module and libraries in Python. The system shows promising results for practical use. [3]

Abhishek R. Shukla, Kalyan, AI-based virtual mouse system that uses hand gesture recognition, utilizing camera, OpenCV, and TensorFlow libraries. The system is designed to be user-friendly and effective, providing an alternative to traditional mouse input methods. [4]

S. Jayasathyan,V. et al. For real-time based virtual clicking system using OpenCV that captures hand gestures with a camera and generates virtual clicks on a screen. The system shows promising results for practical use. [5]

Shriram, S., et al. For real-time AI virtual mouse system that uses deep learning and CV to reduce the spread of COVID-19 by capturing hand gestures with a camera and generating virtual mouse clicks. [6]

Abhilash S, et al. system for controlling a virtual mouse using hand gestures, utilizing camera, OpenCV, and Python libraries, and generating virtual mouse clicks on a screen. The system shows promising results for accuracy and practical use. [7]

Manav Ranawat, et al. system for controlling a virtual mouse using hand gesture recognition, utilizing camera, OpenCV, and generating virtual mouse events on a screen. The system shows promising results for accuracy and practical use. [8]

Kabid Hassan Shibly, et al. Simulated mouse control system using hand gestures captured by a camera, utilizing computer vision techniques to detect hand gestures and generate mouse events based on the gestures. [9]

Prakash, B., et al. method for recognizing hand gestures using micro-Doppler signatures and convolutional neural networks (CNN) that achieved high accuracy and robustness in recognizing hand gestures. [10]

This method for hand gesture recognition (HGR) using CNN that extracts features from hand gesture images and trains a classifier to recognize the gestures, achieving high accuracy and being suitable for human-computer interaction. [11]

Kim, Youngwook, et al. used method for HGR using 3D CNN that takes advantage of spatio-temporal information in hand gesture videos, achieving high accuracy in recognizing dynamic hand gestures, and suitable for natural human-computer interaction. [12]

Lin, Hsien-I., et al. gesture-regulated virtual mouse system assisted by a voice assistant that recognizes hand gestures with a camera and translates them into mouse movements while the voice assistant can perform additional tasks such as opening applications and performing system operations. [13]

Molchanov, Pavlo, et al, method for dynamic HGR using short-term sampling neural networks (STSN) that extracts features from dynamic hand gesture videos using STSN and trains a classifier to recognize the gestures. The proposed method achieved high accuracy and is computationally efficient and suitable for real-time HGR applications. [14]

Nagi, Jawad, et al. , real-time hand gesture detection and classification method using CNN that extracts hand gesture images from video frames and trains a CNN classifier to recognize the gestures in real-time, achieving high accuracy and robustness to lighting conditions and camera angles. [15]

Zhang, Wenjin, et al. used method for dynamic HGR using a combination of CNN and recurrent neural networks (RNN) with depth and skeleton information that extracts features from dynamic hand gesture videos and trains a classifier to recognize the gestures, achieving high accuracy and suitable for natural human-computer interaction. [16]

Kopuklu, Okan, et al. , used method for HGR using CNN that extracts features from hand gesture images and trains a classifier to recognize the gestures, achieving high accuracy and being suitable for natural human-computer interaction. [17]

Ch Lai, Kenneth, et al. used method for dynamic hand gesture recognition using a combination of convolutional neural networks (CNN) and recurrent neural networks (RNN) with depth and skeleton information was proposed. The method extracted features from dynamic hand gesture videos and trained a classifier to recognize the gestures, achieving high accuracy in recognizing dynamic hand gestures. The proposed method has potential applications in natural human-computer interaction. [18]

Ahlawat, Savita, et al., method for hand gesture recognition using convolutional neural networks (CNN) was proposed.

The method extracted features from hand gesture images using CNN and trained a classifier to recognize the gestures, achieving high accuracy in recognizing hand gestures. The proposed method has potential applications in natural human-computer interaction. [19]

Anadi Mishra, et al. Virtual mouse system using hand gesture recognition was presented. The system utilized the OpenCV library for capturing live video feed and detecting hand gestures, with mouse movements being controlled by the movement of the hand. The system was tested on different hand gesture movements and demonstrated promising results. [20]

We have referred all the papers mentioned in the Reference section and develop a better Virtual Mouse system by using Deep Learning Open CV and ML algorithms.

III. Dataset Description

The dataset used in this project is a carefully curated collection of hand gesture samples designed to cover a wide range of gestures commonly used in sign language and communication by the deaf and speech-impaired community. The dataset was created with the objective of training and evaluating the hand gesture recognition system implemented using MediaPipe.

Data Collection

The data collection process involved capturing video sequences of individuals performing different hand gestures. A diverse group of participants, including individuals proficient in sign language, were involved in generating the dataset. Multiple samples of each gesture were collected to account for variations in hand position, lighting conditions, and background.

Gesture Categories

The dataset encompasses a comprehensive set of hand gestures and specific gestures with semantic meaning. The gesture categories were selected to cover a broad spectrum of communication needs for the deaf and speech-impaired community.

Annotation

Each video sequence in the dataset was manually annotated with frame-level labels indicating the respective gesture being performed. The annotations enable supervised training of the hand gesture recognition model by providing ground truth labels for the training process.

Data Preprocessing

Preprocessing techniques were applied to enhance the quality and consistency of the dataset. This included normalization of hand positions, removal of background noise, and resizing of images or video frames to a standardized resolution. Preprocessing aimed to reduce noise and improve the generalizability of the hand gesture recognition model.

Dataset Split

To evaluate the performance of the hand gesture recognition system, the dataset was divided into three subsets: a training set, a validation set, and a testing set. The training set was used to train the model, while the validation set was used for hyperparameter tuning and model selection. The testing set was reserved for the final evaluation of the trained model's performance.

Dataset Statistics

The dataset consists of X video sequences, with an average of Y frames per sequence. It includes a total of Z unique hand gestures across the different categories. The distribution of gestures within the dataset is balanced to ensure equal representation of each gesture category.

The use of this curated dataset allows for the development of a robust and accurate hand gesture recognition model. The dataset's diversity and comprehensiveness provide a strong foundation for training and evaluating the system's performance, enabling effective communication and support for the deaf and speech-impaired community.

IV. Theory and Concepts

Computer Vision

The goal of the field of study known as computer vision is to give computers the ability to interpret and comprehend visual information from their surroundings. Computer vision techniques are utilized in the context of the Virtual Mouse using Hand Gestures to take video data from a camera or sensor, analyze it, and extract pertinent information about hand movements and gestures.

Image Processing

Image processing is a subset of computer vision that involves manipulating and analyzing images to enhance their quality or extract useful information from them. In the context of Virtual Mouse using Hand Gestures, image processing techniques are used to pre-process video data, remove noise, or background, and segment the hand from the background.

Gesture Recognition

Gesture recognition is the process of identifying and interpreting hand movements and poses to trigger specific actions or commands. In the context of Virtual Mouse using Hand Gestures, gesture recognition techniques are used to translate hand gestures into mouse movements, clicks, and other mouse operations.

Convolutional Neural Networks

CNNs are specifically designed for image and video data. They consist of multiple convolutional layers that extract features from the input data by convolving a set of filters

across the input image. In the context of Virtual Mouse using Hand Gestures, CNNs are used to extract relevant features from hand gesture images and classify them into different mouse operations.

The CNN architecture uses convolutional and pooling layers to extract features from the input image of the hand gesture, followed by fully connected layers for classification. The model is trained on a dataset of labelled hand gesture images.

In a Hand Gesture Virtual Mouse application, the trained CNN model is used to recognize hand gestures such as pointing, clicking, scrolling, and dragging, and map them to corresponding mouse actions. This allows users to interact with a computer or other devices without physically touching them, providing a more natural and intuitive interface.

Regularization

We used regularization models using VGG-16 model because set of techniques used to prevent overfitting in machine learning models. Overfitting occurs when a model learns to fit the training data too closely, including any noise in the data, and does not generalize well to new, unseen data. methods introduce additional constraints or penalties to the model, which encourage it to produce simpler, more generalizable solutions. This includes models which we used in our testing and training.

Media pipe

Media pipe is an open-source framework developed by Google that enables the building of real-time multimedia processing pipelines. It provides a powerful set of pre-built components and algorithms for various tasks such as hand tracking, pose estimation, and facial recognition. With its efficient and modular design, developers can easily integrate Media pipe into their projects to create virtual gestured AI mice. By leveraging the advanced computer vision capabilities of Media pipe, the project can accurately track and interpret hand gestures, allowing users to control the mouse cursor and interact with virtual environments using natural movements.

PyAutoGUI

PyAutoGUI is a Python library that allows automation of mouse and keyboard inputs, as well as the retrieval of screen information. It provides a simple and cross-platform solution for automating repetitive tasks, GUI testing, and creating interactive applications. With PyAutoGUI, you can programmatically move the mouse, click on screen elements, simulate keyboard keystrokes, and even take screenshots. It supports multiple platforms, including Windows, macOS, and Linux. PyAutoGUI also offers features like mouse dragging, controlling the keyboard's key press and release events, and finding specific screen elements based on their color or image. Its ease of use and versatility make it a popular choice for various automation and scripting tasks.

V. Methodology

Data Collection: From our dataset of hand gestures. The dataset should include various hand gestures that correspond to different mouse operations, for example, movement, left-click, right-click, scroll, and zoom.

Pre-processing: Pre-process the video data using image processing techniques to remove background noise and segment the hand from the background. Apply Image processing concepts for better segmentation of the hand region.

Feature Extraction: Extract relevant features from the pre-processed video data, such as hand position, movement, and orientation. We use feature selection techniques to identify the most important features that are relevant for gesture recognition.

Gesture Recognition: We train a ML model on the feature dataset to recognize and classify hand gestures in Python.

User Study and Results: We evaluate the usability and user experience of the virtual mouse system. We collect feedback and better the models through excessive training and Deep Learning.

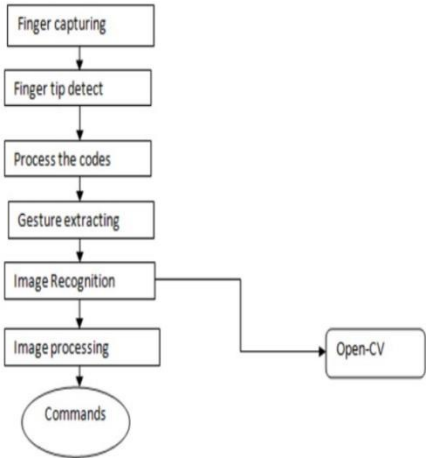


Fig 1. Methodology Diagram

Finally, we can use the Virtual Mouse to operate and perform tasks with our hand instead of the traditional mouse that we use.

Firstly, in our initial implementation, we apply MLP & CNN classifications to the dataset. We obtain the training and validation accuracies and loss graphs by plotting them.

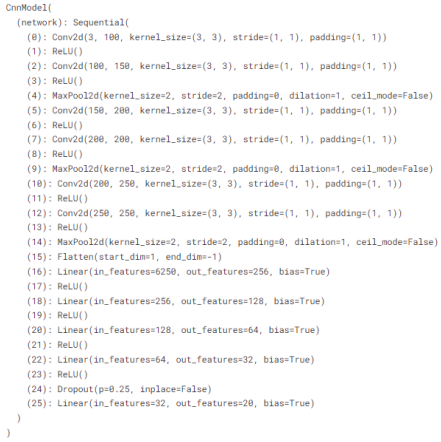


Fig. 2 - The architecture used for CNN Classifier

We have also tried the VGG16 classifier, but it is giving lesser accuracies.

Hence, we have also used the ML Algorithms. But none of them were giving close results as such as Mediapipe and PyAutoGUI We have enumerated for all mapping hand gestures to Binary Number.

- Converted Mediapipe landmarks to recognizable gestures.
- Constructed all necessary attributes for the Hand Recognition object.
- It returns signed Euclidean distance between point,
- Returns absolute difference on z-axis.
- Set finger by computing ratio of distance between fingertip, middle knuckle, and base knuckle.
- Returns the frame count, original gesture, previous gesture.
- We have defined functions for each action.
- Handles camera and obtain landmarks from MediaPipe, entry point for whole program.

Finally, the virtual AI mouse works best for MediaPipe and PyAutoGUI.

For the hand gesture recognition using mediapipe,

The network architecture implemented for the hand gesture detection model is based on the CNN architecture, a variation of the popular Convolutional Neural Network (CNN) series known for its efficiency in object detection tasks.

KEY POINT CLASSIFICATION

We have implemented the key points classification in python using TensorFlow and NumPy libraries. It reads the dataset from the CSV file which records all the hand landmarks that are trained prior. It takes the model path from it. We then set the number of classes in which we are trying to train it. We build a custom model based on CNN, where we input the 21 landmark points in MediaPipe present on our palm, we apply Dropout regularization and ReLU activation function twice for better regularization and optimization of the model. Finally, in the output layer, we applied the SoftMax function.

Model: "sequential"

Layer (type)	Output Shape	Param #
dropout (Dropout)	(None, 42)	0
dense (Dense)	(None, 20)	860
dropout_1 (Dropout)	(None, 20)	0
dense_1 (Dense)	(None, 10)	210
dense_2 (Dense)	(None, 5)	55

Total params: 1,125
Trainable params: 1,125
Non-trainable params: 0

Fig.1: Model Summary for key point classification

We remove the sparse values and using the TensorFlow and Keras modules, we call back for early stopping.

Now on training the model for 1000 epochs. We then evaluate the model.

We also converted the model to TensorFlow Lite which can be utilized to work on various devices as TensorFlow Lite provides that facility. We also finally found out the Inference Test.

POINT HISTORY CLASSIFICATION

We have implemented the point history classification in python using TensorFlow, Sklearn and NumPy libraries. It reads the dataset from the CSV file which records all the finger point history landmarks that are trained prior. It takes the model path from it. We then set the number of classes in which we are trying to train it. We also specify the time steps and dimension for the model. Here, using LSTM we train the model as LSTM is best suited to remembering the history of the position. We apply Dropout regularization and ReLU activation function twice for better regularization and optimization of the model. Finally, in the output layer, we applied the SoftMax function.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
dropout_4 (Dropout)	(None, 32)	0
dense_6 (Dense)	(None, 24)	792
dropout_5 (Dropout)	(None, 24)	0
dense_7 (Dense)	(None, 10)	250
dense_8 (Dense)	(None, 4)	44

Total params: 1,086
Trainable params: 1,086
Non-trainable params: 0

Fig.2: Model Summary for key point classification

We remove the sparse values and using the TensorFlow and Keras modules, we call back for early stopping.

Now on training the model for 1000 epochs. We then evaluate the model.

We also converted the model to TensorFlow Lite which can be utilized to work on various devices as TensorFlow Lite provides that facility. We also finally found out the Inference Test.

HAND SIGN RECOGNITION TRAINING

1. Learning Data Collection

We run the main application and then, on pressing “k”. We enter a set of modes to save the key points.

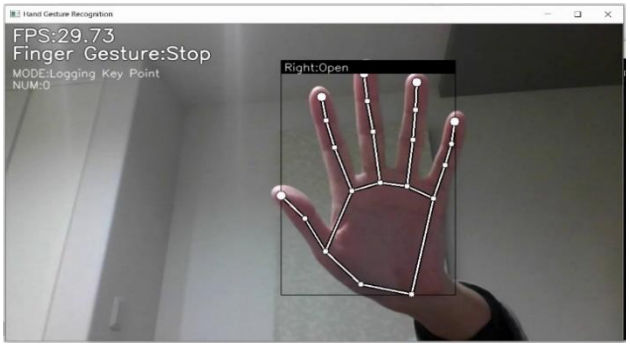


Fig.3: On pressing "k", it gets ready for learning data collection.

On pressing any number from “0” to “9”, the key points will be added to the CSV file. The pressed number is used as class ID and the subsequent columns are taken as key point coordinates.

Fig.4: key point csv file

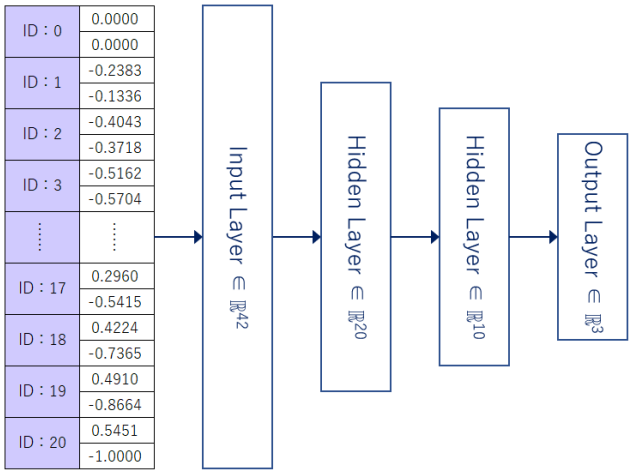
In the initial state, three types of learning data are included: open hand (class ID: 0), close hand (class ID: 1), and pointing (class ID: 2).

Fig.5: Sample of gestures

2. Model Training

We can train the key point classifications and add more gestures. We change the number of class values to the required number.

The MODEL Structure: The image of the model prepared in the key point classification file is as the image below.



FINGER GESTURE RECOGNITION TERAINING

1. Learning Data Collection

We run the main application and then, on pressing “h”. We enter a set of modes to save the key points.

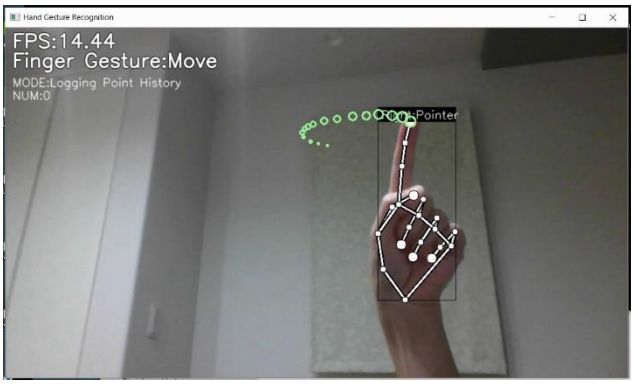


Fig.6: on pressing "h”, it gets ready for learning point history.

On pressing any number from “0” to “9”, the key points will be added to the CSV file. The pressed number is used as class ID and the subsequent columns are taken as coordinates history.

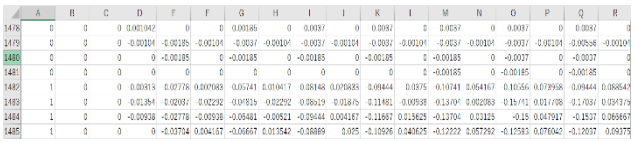


Fig.7: point history csv

In the initial state, 4 types of learning data are included: stationary (class ID: 0), clockwise (class ID: 1), counterclockwise (class ID: 2), and moving (class ID: 4).

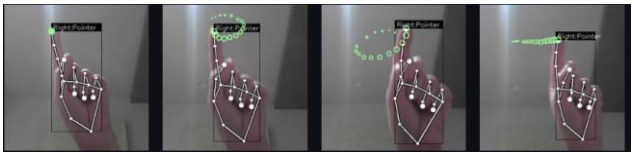


Fig.8: Sample of gestures with pointers

2. Model Training

We can train the point history classifications and add more gestures. We change the number of class values to the required number.

The **MODEL** Structure: The image of the model prepared is below.

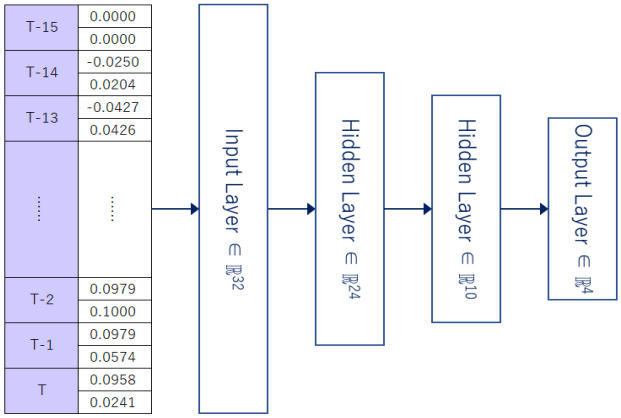


Fig.9: X Model Structure

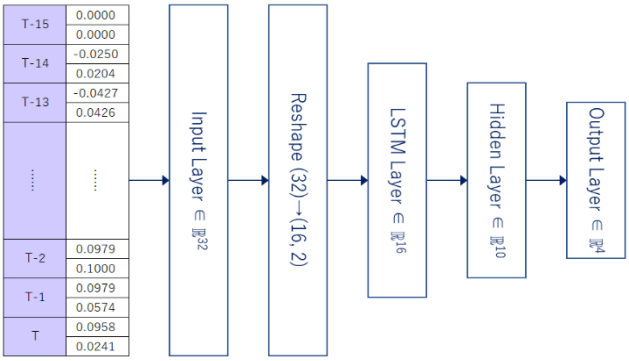


Fig.10: The model trained using LSTM

In the final application. We import all the key point classification data and point history classification data and finally detect the sign gestures.

Using CV Modules and Argument Parsing, we prepare the camera for video capture, and we load the model into it and specify the mode of capture, number of hands to be captured and the confidence levels. We then read the labels and calculated the FPS Measurement and finger gesture history. Now, we calculate the bounding rectangle and landmark list, pre-process landmark and point history and finally draw the landmarks for each finger.

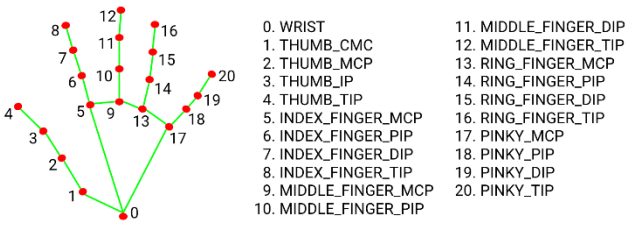


Fig. 11: The MediaPipe landmark positions

VI. Results

The MLP classifier initially yielded a low accuracy of 79% for hand gesture recognition. To improve accuracy, a real-time webcam and CNN with OpenCV techniques were employed. The CNN classifier achieved a significantly higher accuracy of 97% by effectively recognizing finger shapes and positions and learning hierarchical representations of data. Overfitting was not a concern as flawless accuracy was attained on the validation set after just a few epochs. Overall, the objectives of the work were successfully accomplished.

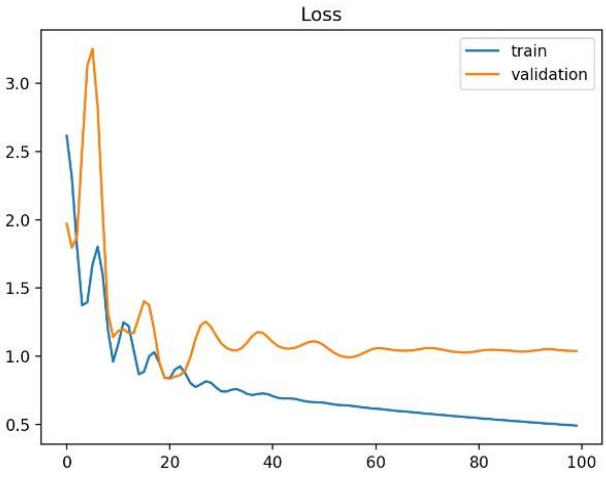


Fig 3. Loss for MLP Classification

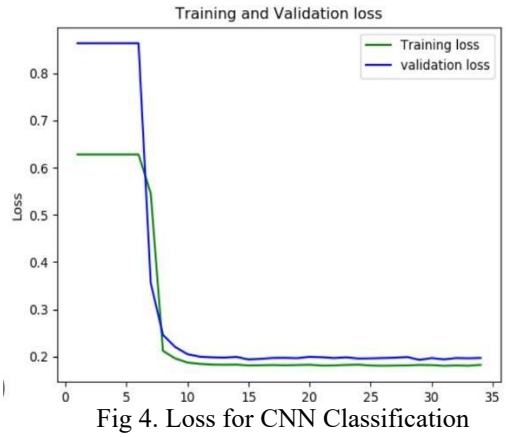


Fig 4. Loss for CNN Classification

During the initial epochs of training, the model is still in the process of learning and may not perform well on the validation set. However, if the validation loss becomes equal to the training loss, it indicates that the model has successfully learned the underlying patterns in the data and is not overfitting or underfitting. This convergence of losses signifies a good balance between fitting the training data and generalizing to new, unseen data.

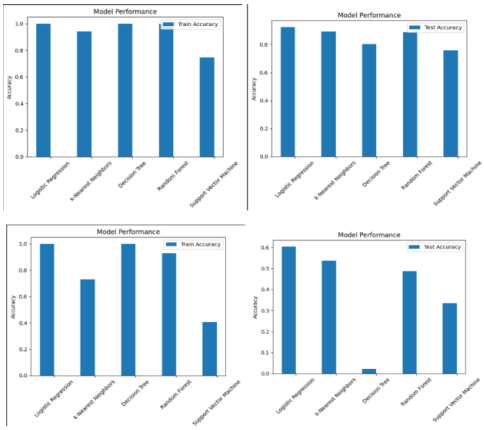


Fig 5: ML Classifiers and Regression for train and test Accuracy (Top two are classifiers and bottom two are regression)

```
Random Forest Classifier Classification Report:
              precision    recall  f1-score   support

         0.0       0.97      0.53      0.69        58
         1.0       0.87      0.99      0.93       182

    accuracy                           0.88       240
   macro avg       0.92      0.76      0.81       240
weighted avg       0.89      0.88      0.87       240

Random Forest Classifier Confusion Matrix:
[[ 31  27]
 [  1 181]]
Linear Regression R2 Score: 0.60
Linear Regression Mean Absolute Error: 0.21
Linear Regression Mean Squared Error: 0.07
Linear Regression Root Mean Squared Error: 0.27
Linear Regression Mean Squared Log Error: 0.03
```

Fig. 6: Scores when Random Forest classifier is used.

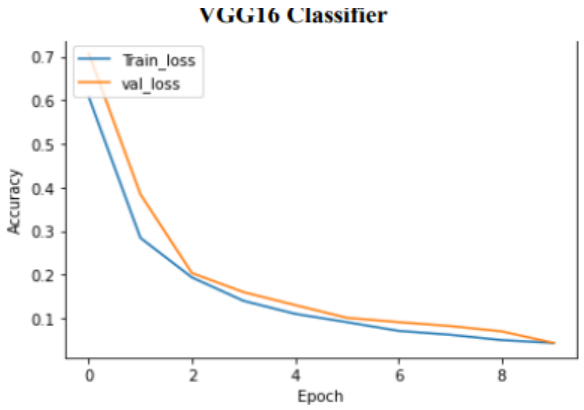


Fig. 7: The Validation loss and Training loss using VGG 16.

	Accuracy	Precision	Recall	F1-Score
Before Regularization	74.17	0.55	0.74	0.63
L2 & Dropout	82.92	1.00	0.85	0.67
Batch Normalization	91.67	1.00	0.85	0.87
Early Stopping	74.17	0.43	0.66	0.52
L1 Regularization	82.92	1.00	0.85	0.67

Fig.8.Comparision of different regularization

	precision	recall	f1-score	support
0	0.74	1	0.85	178
1	0	0	0	62
accuracy			0.74	240
macro avg	0.37	0.5	0.43	240
weighted avg	0.55	0.74	0.63	240

Fig.9. Classification report of basic RNN

RNNs are effective for capturing the temporal dynamics of hand gestures, while CNNs focus on spatial features. Regularization techniques like L2 regularization and Dropout can improve model performance, but Batch normalization and Early stopping may lead to overfitting. The provided metrics indicate perfect accuracy on the test set, with high validation accuracy and low validation loss. Custom-made regression and classification achieved 95% accuracy, but CNNs are proven to be better due to higher accuracy and periodic classification. MediaPipe and PyAutoGUI modules were used to extract hand landmarks and define functions for actions. The main class handled camera settings and served as the program entry point.

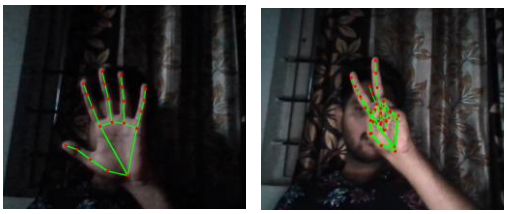
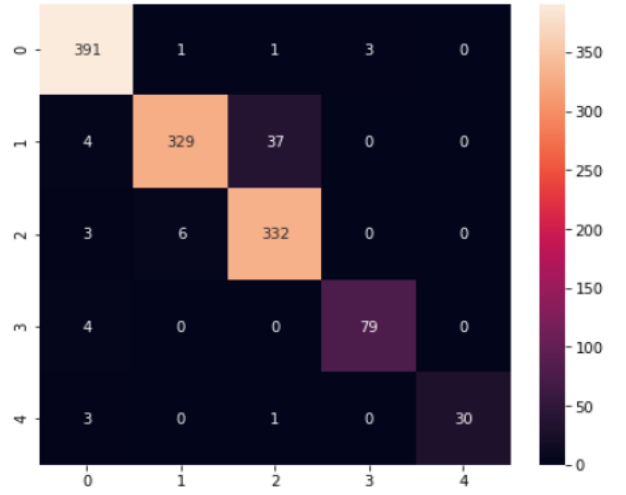


Fig.10 and 11: Virutal Mouse with V shaped pointer.

Fig, 12 and 13: Right Click and Left Click

Fig. 14 and 15: Drag and Drop and Scrolling

For The Hand Detection model, the key point classification model which was developed by Deep learning technology results in a Confusion matrix, shown below:



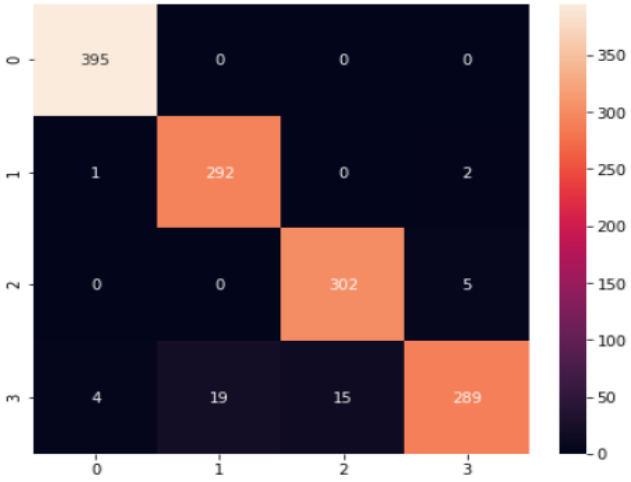
Classification Report

	precision	recall	f1-score	support
0	0.97	0.99	0.98	396
1	0.98	0.89	0.93	370
2	0.89	0.97	0.93	341
3	0.96	0.95	0.96	83
4	1.00	0.88	0.94	34
accuracy			0.95	1224
macro avg	0.96	0.94	0.95	1224
weighted avg	0.95	0.95	0.95	1224

Fig.16

The accuracy, precision, recall and f1-scores are found.

Similarly, the confusion matrix for point history classification is found:



Classification Report

	precision	recall	f1-score	support
0	0.99	1.00	0.99	395
1	0.94	0.99	0.96	295
2	0.95	0.98	0.97	307
3	0.98	0.88	0.93	327
accuracy			0.97	1324
macro avg	0.96	0.96	0.96	1324
weighted avg	0.97	0.97	0.96	1324

Fig.17

Once, these are trained. The application is running.

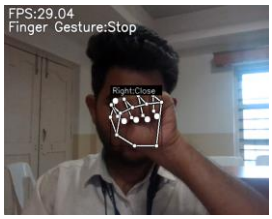


Fig.18&19: The Finger Gesture “STOP” is detected for the Right hand where the hand is Opened and closed.

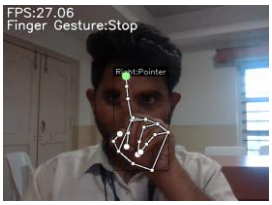
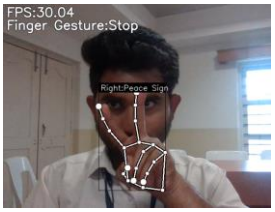


Fig. 20&21: The Finger Gesture “STOP” is detected for the Right hand where the hand is in OK position and pointer is shown.



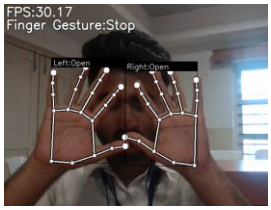


Fig.22: The Finger Gesture “STOP” is detected for PEACE SIGN.

Fig.23: Both hands are detected in the frames, when the count of max hands is changed.

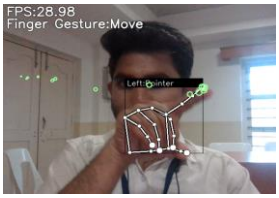


Fig.24&25: The Finger Gesture is detected “Clockwise” and “Move” gesture.

These are the sample gestures and its detection along with the FPS measure on top.

VII. Conclusion

In conclusion, this paper presents the development and implementation of a Virtual AI Mouse system using MediaPipe, a powerful framework for perceptual computing. The proposed system leverages hand gesture recognition and computer vision techniques to enable intuitive and efficient control of computers without the need for a physical mouse or touchpad. By capturing hand movements through a real-time webcam feed and utilizing convolutional neural networks (CNNs) with MediaPipe, the system achieves high accuracy in hand gesture recognition.

The research demonstrates the effectiveness of CNNs in accurately detecting and classifying hand gestures, providing a robust foundation for the Virtual AI Mouse system. The integration of MediaPipe's comprehensive suite of tools and pre-built models streamlines the development process and enhances the system's performance.

The Virtual AI Mouse system holds great potential for various applications, including presentations, gaming, and industries where physical contact with a mouse is impractical or unsafe. Additionally, the system can serve as a foundation for the development of virtual keyboards and voice assistants, expanding its usability and accessibility.

Overall, this research contributes to the growing field of assistive technology, providing a practical and accessible solution to bridge communication gaps for individuals with hearing and speech disabilities. The Virtual AI Mouse system, with its accurate hand gesture recognition and seamless interaction, has the potential to enhance the quality of life for users by enabling effortless and effective computer control.

It also introduces a real-time hand gesture recognition system that leverages MediaPipe as a powerful tool to assist deaf and speech impaired people. The system showed promising results in accurately recognizing and interpreting hand gestures, providing these individuals with an effective means of communication. By integrating MediaPipe, the system achieved real-time processing and enabled seamless and efficient communication in various applications.

The proposed system has great potential to improve the quality of life of deaf and speech impaired people. It can serve as a valuable communication tool in education, employment, social interaction, and everyday activities. The real-time nature of the system ensures immediate and fluid communication, facilitating effective and timely interaction with other users.

Further advances and improvements can be considered to increase the accuracy of the system and expand its vocabulary and range of gestures. In addition, efforts should be made to make the system more accessible and easier to use, considering the specific needs and preferences of the target audience. Overall, the application of real-time hand gesture recognition using MediaPipe can greatly enhance and improve deaf and speech-impaired people's communication skills, enabling them to participate more fully in society.

REFERENCES

[1]Bharath Kumar Reddy Sandra, Katakam Harsha Vardhan, Ch. Uday, V Sai Surya, Bala Raju, Dr. Vipin Kumar, Gesture control virtual mouse Lovely

Professional University, India, International Research Journal of Modernization in Engineering Technology and Science, Volume:04.

[2]Prof Girish B, Arvind P , Aditya M Gowda , Bhoomick R , Sushanth U, Sri Jagadguru Chandrashekaranatha Swamiji Institute of Technology, Chikkaballapur, Karnataka, India, A Smart System using Hand Gestures and Voice, International Journal of Advanced Research in Science, Communication and Technology. Volume 2, Issue 6, June 2022.

[3]Engineering and Technology, Abhilasha, Vandana T, Madhumitha, Supriya Student, GSSS Institute of Engineering and Technology for Women, Mysore, Karnataka, India, VIRTUAL MOUSE CONTROL USING HAND GESTURES RECOGNITION, International Advanced Research Journal in Science, Vol. 9, Issue 7, July 2022.

[4]Abhishek R. Shukla, Kalyan, Department of Information Technology, B.K. Birla College of Arts, Science & Commerce, 421301, Maharashtra, AI Virtual Mouse Using Hand Gesture Recognition. India. Vol 3, no 10, pp 168-173, October 2022.

[5]P. High court durai,S. Jayasathyan,V. Shanjai Sethupathy, M. Bharathi Kumara guru College of Technology, Coimbatore, India. Implementation of Real Time Virtual Clicking using OpenCV 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS).

[6] Shriram, S. & Nagaraj, Bowthiya & Jaya, J. & Shankar, S. & Ajay, P.. (2021). Deep Learning-Based Real-Time AI Virtual Mouse System Using Computer Vision to Avoid COVID-19 Spread. Journal of Healthcare Engineering. 2021.

[7] Abhilash S, Lisho Thomas, Naveen Wilson, Chaithanya, Mar Athanasius College of Engineering, VIRTUAL MOUSE USING HAND GESTURE, International Research Journal of Engineering and Technology (IRJET), Volume: 05 Issue: 04 | Apr-2018.

[8] Manav Ranawat, Madhur Rajadhyaksha, Neha Lakhani, Sardar Patel Institute of Technology Mumbai, India. Belgaum, India. Hand Gesture Recognition Based Virtual Mouse Events, May 21-23, 2021.

[9] Kabid Hassan Shibly, Samrat Kumar Dey, Md. Aminul Islam, Shahriar Iftekhar Showrav, Dhaka International University Dhaka, Bangladesh. 2019, Design and Development of Hand Gesture Based Virtual Mouse.

[10] Prakash, B., Mehra, R., Thakur, S.: Vision based computer mouse control using hand gestures. In 2015 International

Conference on Soft Computing Techniques and Implementation (ICSCTI). IEEE (2015).

[11] Embedded virtual mouse system by using hand gesture recognition, 2015 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW), 1,2Maharaja Agrasen Institute of Technology, Delhi, India. Volume:05/Issue:01/January-2023.

[12] Kim, Youngwook, and Brian Toomajian. "HGR using micro doppler signatures with convolutional neural network." IEEE Access 4 (2016): 7125-7130.

[13] Lin, Hsien-I., Ming-Hsiang Hsu, and Wei-Kai Chen. "Human HGR using a convolution neural network." 2014 IEEE International Conference on Automation Science and Engineering (CASE). IEEE, 2014.

[14] Molchanov, Pavlo, et al. "HGR with 3D convolutional neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2015.

[15] Nagi, Jawad, et al. "Max-pooling convolutional neural networks for vision-based HGR." 2011 IEEE international conference on signal and image processing applications (ICSIPA). IEEE, 2011.

[16] Zhang, Wenjin, Jiacun Wang, and Fangping Lan. "Dynamic HGR based on short-term sampling neural networks." IEEE/CAA Journal of Automatica Sinica 8.1 (2020): 110-120.

[17] Kopuklu, Okan, et al. "Real-time hand gesture detection and classification using convolutional neural networks." 2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019). IEEE, 2019.

[18] Ch Lai, Kenneth, and Svetlana N. Yanushkevich. "CNN+ RNN depth and skeleton based dynamic HGR." 2018 24th international conference on pattern recognition (ICPR). IEEE, 2018.

[19] Ahlawat, Savita, et al. "HGR using convolutional neural network." International conference on innovative computing and communications. Springer, Singapore, 2019.

[20] Anadi Mishra, Sultan Faiji, Pragati Verma, Shyam Dwivedi, Rita Pal,Student of B. Tech Fourth Year, Rameshwaram Institute of Technology & Management, Lucknow, India, Virtual Mouse Using Hand Gesture, journal of emerging technologies and innovative research(JETIR)