The Use of Hand Gestures as a Tool for Presentation

Hope Orovwode[1], John Amanesi Abubakar[2], Onuora Chidera Gaius[3], Ademola Abdullkareem[4]
Department of Electrical and Information Engineering, Covenant University, Ota, Ogun State, Nigeria[1, 3, 4]
Department of Computer Science and Engineering, University of Bologna, Bologna, BO, Italy[2]

***Abstract*—Our hands play a crucial role in daily activities, serving as a primary tool for interacting with technology. This paper explores using hand gestures to control presentations, offering a dynamic alternative to traditional devices like mice or keyboards. These conventional methods often limit presenters to a fixed position and depend on the device's proximity. In contrast, hand gesture controls promise a more fluid and engaging presentation style. This study utilizes the HaGRID dataset, supplemented by custom-recorded data, divided into 80% for training, and 10% each for validation and testing. The data undergoes preprocessing and a linear classifier with four dense layers and a SoftMax activation layer is employed. The model, optimized with the Adam optimizer and a learning rate of 1e-1, incorporates a motion classifier (LSTM) with two dense layers and an LSTM layer, tailored for long-distance body pose estimation. The resulting application, a local desktop tool independent of internet connectivity, uses tkinter for its user interface. It demonstrates high accuracy in classifying gestures, achieving 90.1%, 89%, and 90% in training, validation, and testing, respectively, for the linear classifier. The motion classifier records 79.8%, 72%, and 70.1%. The model effectively recognizes and categorizes dataset gestures, capturing live camera feeds to manage presentations. Users benefit from various features, including PowerPoint selection, distance mode, gesture toggling and assignment, and appearance mode. This study illustrates how hand gesture control can enhance presentation experiences, merging technology with natural human movement for a more seamless interaction.**

***Keywords*—Hand gesture; linear classifier; motion classifier; LSTM; interface**

I. Introduction

Our hands are more flexible when it comes to the usage of our body and when it comes to movement, so they are involved greatly in our day-to-day activities. It is a natural occurrence that someone uses their hands immediately when they wake up or want to get something. Researchers have been going over ways of interacting with computers and other devices and the aim is trying to increase human-computer interaction (HCI) through a communication channel from the user to the device. One of the ways we can communicate with a computer is with our "hands" making movements which are said to be hand gestures. The use of hand gestures as a means of interacting with technology has become increasingly popular and allows users to interact more naturally and intuitively and it has started entering the field of presentation [1]. Hand gestures have been used as a form of major communication in presentations for ages. Hand gestures have been effectively used in presentations by renowned presenters and public figures across past times, including Martin Luther King Jr., Winston Churchill, and Steve Jobs, to captivate audiences and convey compelling messages [2]. This historical setting highlights the importance of hand gestures as an important part of effective communication. Hand gestures in the field of presentation enhance communication performance, especially for presenters who may experience stage fright. While slides are often seen as the focal point, effective hand gestures can captivate the audience and draw their attention into the presentation. It then aids in emphasizing essential points, drawing focus on significant details, and illustrating ideas or operations. Presenters can improve listener understanding and recollection of information by adopting gestures that correspond to the topic [3]. Non-verbal indicators, such as hand gestures, are an important part of communicating. These aid in the general comprehension and understanding of the information. Presentations get better when presenters enhance their verbal communication, express emotions, and include richness in their presentations by integrating intentional and well-timed gestures [4].

In recent years, many eyes have been on various technologies to offer improved user interfaces that enable computer interactions as intuitive as human interactions. Furthermore, standard gadgets such as a keyboard and mouse cannot entirely fulfil human interaction needs in presentations [5]. The study on controlling presentations using hand gestures is significant as it aims to enhance user experience, increase interactivity, improve accessibility, enable seamless integration, and drive innovation in presentation delivery. By developing a presentation package that allows presenters to control their presentations using hand gestures, the study seeks to provide a more intuitive and natural interaction method. This approach eliminates the need for physical devices or complex keyboard shortcuts, making the presentation process seamless and enjoyable. Incorporating hand gestures in presentations enhances interactivity, captures the audience's attention, and promotes engagement. The study also emphasizes the importance of seamless integration with popular presentation software and contributes to the advancement of interaction techniques and innovative presentation delivery methods [6].

In summary, this paper investigates the gap focusing on the practical implementation of hand gesture recognition. Section II delves into the existing literature on the subject, establishing the foundation for our study. Section III outlines the research methodology employed, followed by Section IV which presents the results and discussion. Section V presents the user implementation. Finally, Section VI concludes the paper.

II. Related Work

When a 15th-century author describes an individual as 'cute of gesture', they are not referring to mere clapping or shaking performed gracefully. Instead, the author is highlighting the

person's graceful movements and posture [7]. This broader bodily movement was referred to as 'gesture'. The study of gestures is not a recent endeavor. Various physiognomists, such as G. B. Della Porta, Charles Le Brun, and J. C. Lavater, have explored the representation of gestures since the Medieval period. In the 17th Century, Francis Bacon emphasized the importance of gestures as a primary form of symbolism. Giovanni Bonifacio and John Bulwer discussed a universal language of gestures that could facilitate international trade and interactions [8] [9] [10]. Charles Darwin's work on the expression of emotions in humans and animals in the 18th century provided further evidence for the biological inheritance of expressions [11]. Present-day ethologists emphasize the similarities between human and animal bodily movements used to convey emotions such as resentment, superiority, or possessiveness. Certain emotional facial expressions like laughter, tears, yawning, and giggling are nearly universal, transcending linguistic and geographical boundaries [12]. However, contemporary views suggest that a global language of gestures is not feasible, as cultural, and social differences play a significant role. There are also various types of gestures and languages [12].

In the early 19th century, Andrea de Jorio attempted to recreate the mimicry of classical antiquity using Neapolitan gestures. Anthropologist Marcel Mauss, in his work 'The Techniques of the Body' (1935), highlighted the vast cultural variations in bodily activities. These differences became apparent when people from different cultures watched foreign films or encountered unfamiliar gestures. Marcel Mauss' insights led to cross-cultural studies of body language and facial expressions. David Efron's research explained how Italians and Eastern Europeans adapted to the gesture culture of the United States [13] [14]. After World War II, interest in communication surged once again. New development theories introduced powerful models for enhancing communication through analogue and digital codes. The language was seen as an example of digital code, while elements of changing natural behaviour were also considered as forms of language. Gestures, although frequently mentioned, received limited attention in this context [15]. This distinction gave rise to the concept of "nonverbal communication," distinct from oral language, focusing on interpersonal relationship building [16] [17]. Ray L. Birdwhistell introduced the concept of kinesics, aiming to study body movement communication systematically, although this didn't lead to extensive studies of gesture [18] [19]. The coalescence of nonverbal communication studies was dedicated to aspects of behaviour not directly linked to spoken language. Ray L. Birdwhistell's contribution was notable, but his definition of kinesics overlooked the analysis of gestures. Some authors, like Morton Wiener and Paul Ekman, attempted to incorporate gestures into nonverbal communication theory, but their efforts remained somewhat isolated [20] [21].

Gordon Hewes' influential study reignited interest in gestures as a key topic of discussion. Hewes argued that original language might have been gestural and cited Gardner's discovery of sign language in juvenile monkeys as a significant foundation. The study of sign languages progressed rapidly after William Stokoe's work on American Sign Language. Although sign language differs from spoken language, it poses a challenge to linguistic models and necessitates its incorporation into linguistic perspectives [22][23]. Gesture and sign language are intertwined, and the resurgence of interest in sign language has contributed to the recognition of gesture as a significant research area once again. A lot of techniques have been used in advancing hand gestures. The research in [24] used CNN and LSTM for gesture recognition. The authors used the CNN approach followed by the CNN+LSTM approach with a skeleton model (hand points), and they empirically showcased the capability of extracting pertinent attributes from 3D skeleton data. This extraction serves as a precursor to effectively address activity recognition through the utilization of LSTM. The result shows a high performance. Gesture identification, tracking and classification were carried out. K. V. Eshitha et al., [25] firstly, segment the hand gestures by using the skin colour model and AdaBoost classifier based on the type of skin colour, which also segments the hand from the background. It is monitored by the Cam Shift algorithm and classified by CNN to recognize 10 common digits. The result shows a 98.3% accuracy. A. Ikram et al., [26] carried out gesture acquisition, segmentation, feature extraction and classification by compiling different gesture datasets, and these gestures are distinguished from an input image using colour-based segmentation then the Histogram of Gradient technique is used to extract features in this case. After segmentation, the gestures are classified into left-hand and right-hand gestures, together with labels, and then sent to an artificial neural network for training. The result shows a high accuracy. The authors combined CNN and RNN to increase the accuracy of gesture classification using dynamic hand gestures. The result obtained is 85.46% H. Y. Chung et al., [27] I. Dhall et al.,[28] carried out hand gesture classification of people with stroke. The data set contains 140 gestures and is trained using CNN. Accuracy gotten is 99%. P. Parvathy et al., [29] used CNN to improve the accuracy of gesture recognition using the OpenCV library and the result shows a 99.13% accuracy. A. Mujahid et al. [30] proposed a lightweight model based on YOLO and DArkNet-53. The gestures are obtained from the Cambridge hand gesture dataset and are pro-processed and segmented using various techniques and then classified using deep CNN. The result shows a 96.66% accuracy. However, despite extensive research in gesture recognition and classification, deployment has been limited.

III. System Analysis and Design

This section outlines a comprehensive methodology employed in this research. It covers the step-by-step procedure and methods utilized for using hand gestures as a tool for presentation. Additionally, it discusses the process of data acquisition, feature extraction and deployment of the model.

Fig. 1 illustrates a comprehensive block diagram of the system starting from data gathering, followed by Mediapipe hands for preprocessing and extraction then a hybrid model comprising of linear classifier and LSTM will be used to develop the model. This hybrid architecture has been thoughtfully designed to combine the strengths of both components, thereby enhancing the model's overall performance and predictive capabilities. The utilization of the LSTM component holds significant promise in capturing

temporal dependencies within the data, further elevating the model's predictive prowess. The training environment is the Jupital Notebook and finally, the model will be deployed in a model as a package.

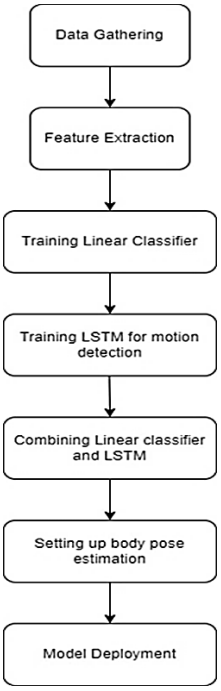


Fig. 1. Block diagram of the system.

A. Data Gathering

This is the first stage in the development of the model. There are two datasets used in the model. The first dataset is called HaGRID (Hand Gesture Recognition Image Dataset) for hand gesture recognition (HGR) systems. HaGRID size is 716GB and the dataset contains 552,992 (1920p × 1080p) RGB images divided into 15 classes of gestures as seen in Fig. 2. The data were split into training 80% for the training set 10% for the validation set and 10% for the test set.

The set contains at least 34,730 distinct individuals and scenes. The subjects range in age from 18 to 65. The dataset was primarily gathered at home, with significant variance in lighting, encompassing both artificial and natural light. Furthermore, the collection contains photographs captured under severe situations, such as facing and backing away from a window. In addition, the subjects were instructed to perform motions between 0.5 and four meters away from the camera.

The second dataset is a custom-made dataset for finger swipe motion which contains five seconds for each video and four classes of gestures.

Fig. 2. Sample of HaGRID dataset.

B. Data Preprocessing

This process utilizes functions provided by the OpenCV library:

1) Color channel conversion: The initial step involves converting the image captured by the webcam from the BGR (blue, green, red) colour channel mode to the RGB mode. This transformation is vital because the models we are using expect images in the RGB format for accurate processing and analysis.

2) Horizontal flipping: To ensure that the image is properly oriented for analysis, a horizontal flip is applied. Since webcams typically capture images facing the subject, this step prevents the image from being displayed in an inverted manner. It ensures that the hand gestures appear as intended when processed by the subsequent stages.

3) NumPy array conversion: Once the colour and orientation adjustments are made, the image is converted into a NumPy array. A NumPy array is a fundamental data structure used in Python for efficient numerical computations. In this case, the image is transformed into a format that can be readily understood and manipulated by deep learning models.

4) Adjusting array shape: The shape of the NumPy array is then modified to adhere to TensorFlow's specific requirements for input data. TensorFlow, a popular deep learning framework, expects the arrangement of data in a specific order. Therefore, the shape of the array is adjusted to have the colour channels come first, followed by the width and height dimensions. This ensures that the input data conforms to TensorFlow's expectations, enabling smooth processing and analysis by the deep learning models.

C. Feature Extraction

This is done using the media pipe hands model and is divided into two phases:

1) Hand detection model: In this phase, a Convolutional Neural Network (CNN) is employed to analyze images or video frames. The primary goal is to detect the presence and location of one or more hands within the visual data. This involves generating a bounding box or a region of interest (ROI) that encapsulates the hand(s). Before entering the CNN, the input image or frame undergoes preprocessing to ensure it is suitable for analysis. This might involve actions such as scaling, normalization, and data transformation into a format compatible with the neural network's requirements.

The architecture of the CNN is meticulously designed to learn patterns and features associated with hands. Comprising multiple layers, including convolutional, pooling, and activation functions, CNN focuses on extracting spatial features from the input data. Throughout this feature extraction process, the CNN discerns important elements like edges, textures, shapes, and spatial relationships that distinguish hands from the background or other objects.

The acquired features are then utilized for classification. Each portion of the input data is assigned a probability or

confidence score, indicating the likelihood of containing a hand.

2) Landmark detection model: After successfully identifying the hand's location using the bounding box, the focus shifts to the more precise estimation of hand landmarks. To achieve this, a specialized CNN is employed, fine-tuned to predict the positions of landmarks on the hand. This CNN operates on the hand ROI extracted from the previous phase.

Similar to the hand detection model, the landmark estimation network comprises layers dedicated to feature extraction. These layers meticulously analyze the hand ROI to capture vital visual patterns, encompassing edges, textures, and other defining characteristics that aid in pinpointing landmarks.

Drawing from these extracted features, the landmark estimation network generates predictions for the coordinates of each hand landmark. These landmarks, totalling 21 key points as seen in Fig. 3, represent precise positions within the image or video coordinate system. These points correspond to crucial locations on the hand, aiding in accurate identification and analysis.

These two phases work collaboratively to provide an integrated solution for hand gesture analysis. The hand detection phase establishes the presence and location of hands, while the landmark detection phase refines this information, pinpointing specific landmarks on the hand. This dynamic process enables precise and nuanced hand gesture recognition, forming the foundation for effective communication and interaction.

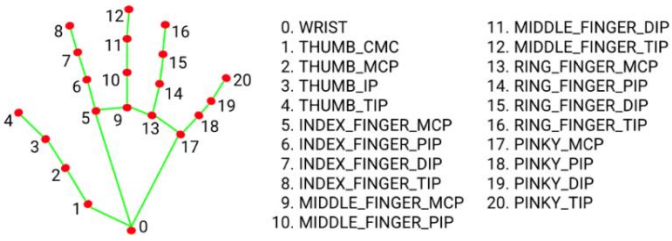


Fig. 3. The 21 key landmarks.

D. The Linear Classifier Model

To commence, the landmark features obtained from hands within the HaGRID dataset using the Mediapipe hands model serve as the input for training the linear classifier. This classifier is structured with four dense layers, also known as fully connected layers. In this arrangement, each neuron is connected to every neuron in the preceding layer and Rectified Linear Activation Units (ReLU) are placed between these neurons. The final layer employs a SoftMax activation, effectively converting logits into probabilities that facilitate classification. The architecture of the linear classifier model can be visualized in Fig. 4, offering a visual summary of its components and connectivity.

During the optimization process, the model is fine-tuned using the Adam optimizer, which utilizes a learning rate of 1e-1. This optimizer plays a pivotal role by iteratively adjusting the weights and biases of the network throughout the training process. It effectively manipulates these parameters to minimize the loss function, thereby enhancing the model's predictive accuracy.

The Adam optimizer operates with the inclusion of two beta parameters: β1 and β2. These parameters govern the decay rates of moving averages of gradients and squared gradients, respectively. In this context, β1 is set to 0.9, and β2 is set to 0.99. These values essentially dictate the extent to which past gradients and squared gradients influence the current weight adjustments during training.

To improve training accuracy and mitigate overfitting, a dropout mechanism is applied across all layers with a rate of 0.2. Dropout involves randomly deactivating a portion of neurons during each training iteration. This deliberate deactivation reduces interdependencies between neurons, encouraging the network to acquire more generalized and robust representations, ultimately enhancing its ability to perform well on new, unseen data.

E. The Motion Classifier Model

To achieve motion detection, we employed a specialized neural network called LSTM, which stands for Long Short-Term Memory. This LSTM network is designed to analyze sequences of landmarks captured throughout 30 consecutive frames or steps. To facilitate training, we utilized a dataset specifically recorded to capture different types of motion. Using the Mediapipe model, we extracted landmark features from this custom motion dataset. These extracted landmark features were then used as input to train the LSTM network. The motion classifier layer diagram is shown in Fig. 5 and the summary of the motion classifier model is shown in Fig. 6.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                2752

 dense_1 (Dense)             (None, 128)               8320

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 32)                2080

 dense_4 (Dense)             (None, 18)                594

=================================================================
Total params: 22,002
Trainable params: 22,002
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4. Linear classifier model architecture.

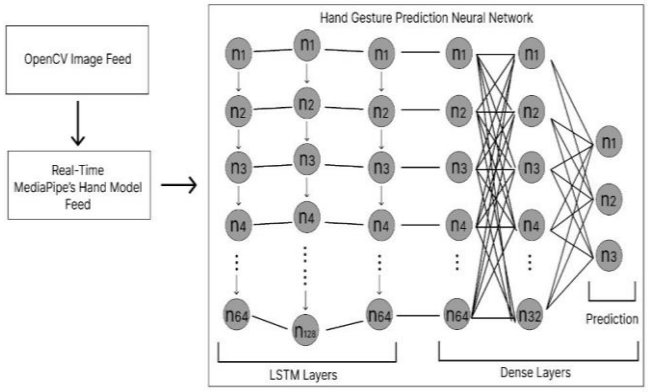
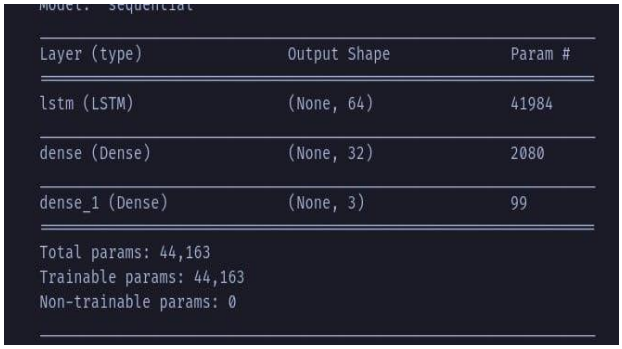


Fig. 5. Model classifier layer diagram.



Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 64)	41984
dense (Dense)	(None, 32)	2080
dense_1 (Dense)	(None, 3)	99

Total params: 44,163
Trainable params: 44,163
Non-trainable params: 0

Fig. 6. Summary of the motion classifier model.

The key strength of the LSTM network lies in its ability to recognize and comprehend patterns within motion data over an extended period. It achieves this by capturing both short-term and long-term dependencies present in the sequential landmark information. This capability enables the LSTM network to understand the intricate nuances of motion sequences, making it adept at detecting and interpreting motion patterns. Throughout the training process, the LSTM network continually adjusts its internal parameters to minimize the disparity between the predicted motion labels and the actual, true motion labels present in the dataset. By iteratively fine-tuning its parameters, the LSTM network effectively learns to classify and differentiate various types of motions. This process involves learning the distinctive characteristics and patterns associated with different motions, enhancing its ability to accurately detect and classify motions within new, unseen data.

F. Combining the Linear Classifier and the LSTM Model

Employing an LSTM (Long Short-Term Memory) model to analyze every individual frame in real-time for prediction is computationally demanding. Due to its high computational needs, we adopt a different strategy by integrating the two previously described models.

First, the linear classifier is applied to each frame captured in real time. The linear classifier, known for its computational efficiency, operates on the landmarks or features extracted from individual frames. It predicts the corresponding gesture or action for each frame efficiently. If the linear classifier confidently detects a meaningful gesture or action in a frame, indicating the potential presence of significant motion, a sequence of frames is then directed to the LSTM model for further motion analysis. The linear classifier's outcome serves as a trigger for the LSTM model's involvement. Specifically, the sequence of frames encompasses the present frame along with a set of previous frames. This sequence is inputted to the LSTM model, which has been designed to excel at capturing temporal patterns and dependencies within motion data.

By analyzing this sequence of frames over time, the LSTM model comprehends the evolving motion pattern. The LSTM's ability to account for sequential information empowers it to provide more precise predictions regarding the detected motion. This innovative approach involves the strategic fusion of the linear classifier and the LSTM model, thereby enhancing computational efficiency. The linear classifier rapidly processes individual frames, identifying potential gestures swiftly. However, the more computationally intensive LSTM model is only engaged when the linear classifier detects a gesture or action, ensuring resource-intensive calculations are focused precisely on moments of interest.

This approach effectively enables real-time motion detection while mitigating the overall computational demands. By leveraging the strengths of both models, we strike a balance between efficiency and accuracy, resulting in a streamlined and effective system for motion analysis.

G. Setting up Body Pose Estimation

To cater to situations in which users are situated at a significant distance from the camera, a specialized feature known as Long-Distance Mode is introduced for Body Pose Estimation. While the standard Mediapipe feature extraction excels when users are in proximity (<2m), it might not accurately capture hand landmarks for individuals located far away.

The concept is illustrated in Fig. 7, which presents a body pose diagram. In this context, we focus on specific points of interest, specifically points 13 to 22 that correspond to the location of the hands. Instead of relying on the precise detection of hand landmarks, the approach involves utilizing an estimated bounding box that encapsulates the hands' spatial extent. This bounding box, serving as input for the classifier layers, becomes the basis for making predictions. In this scenario, the classifier operates on the information derived from the bounding box rather than individual hand landmarks. This adjustment allows the classifier to predict and classify gestures or actions based on the overall position and region of the hands within the bounding box.

By incorporating the Long-Distance Mode and leveraging the bounding box estimation from the Mediapipe pose extraction, the system adeptly handles situations where users are positioned at a considerable distance from the camera. This innovation ensures that the system remains versatile and capable of accurately analyzing hand gestures, even when users are far away.

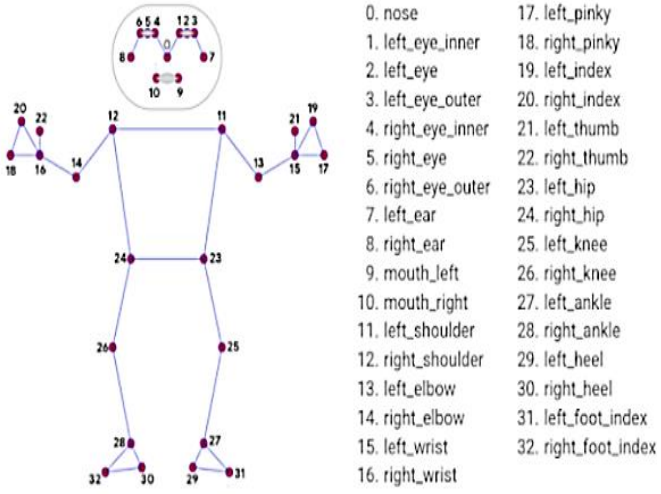


Fig. 7. Body pose diagram.

H. Flow Chart of the Model

The flow chart of the working of the model can be visualized in Fig. 8. The algorithm for the flow chart is as follows.

Step 1: Start

Step 2: Gather the data

Step 3: Carry out Preprocess and Data extraction

Step 4: Select the neural network to use for the model

Step 5: Set up body pose estimation

Step 6: Train, test and validate the model

Step 7: Is Performance Satisfactory?

- If Yes, Go to Step 8
- If No, Go to Step 4

Step 8: Save Model

Step 9: Stop

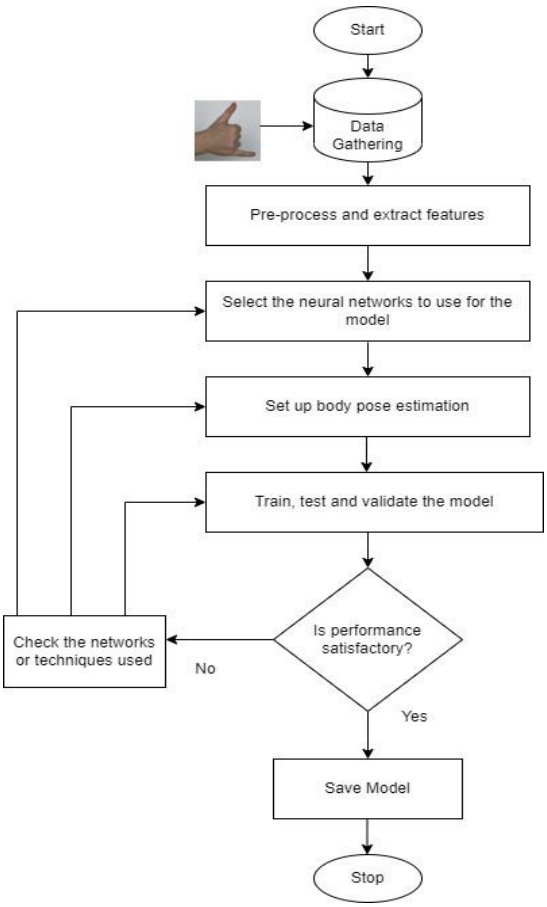


Fig. 8. Flow chart of the system.

I. Model Deployment

The model is built as a local desktop application that does not require any internet connection during usage. The user interface for the application was made using custom tkinter – a python library which uses the tkinter as its baseline but adds customizations to make the interface more user-friendly; also the models are deployed using TensorFlow lite (tflite) models to speed up inference.

Various diagrams as seen in Fig. 9 and Fig. 10 are used to illustrate the working of the deployed model as software. The diagrams are the use case diagram and the flow chart respectively. Fig. 9 shows the Unified Modeling Language (UML) of the system functions and responses, and Fig. 10 shows the flow of the deployed system.

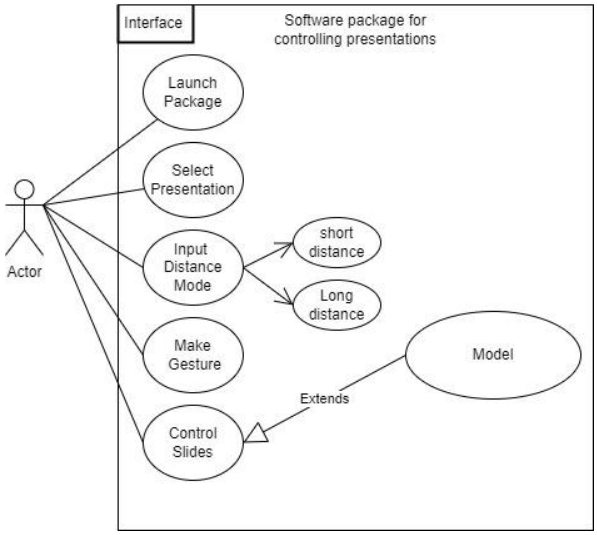


Fig. 9. Use case diagram.

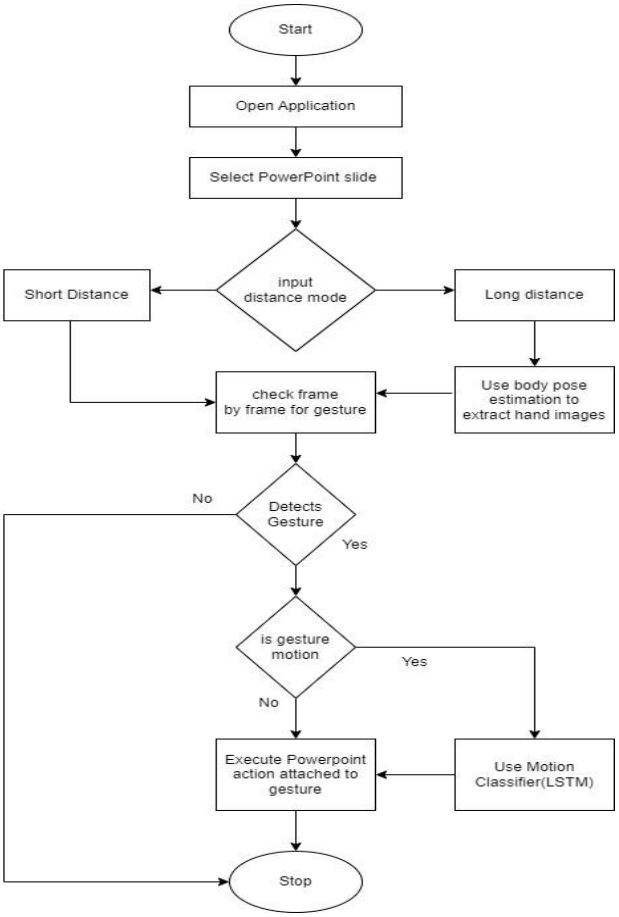


Fig. 10. Model deployed flow chart.

IV. Result and Discussion

This section discusses the results and execution of the model from the previous section and the performance of the system when evaluated. It defines the system constraints and tells us a bit more about the effectiveness of the system.

A. Model Training and Validation

The linear classifier model was trained using a strategy of 10 epochs, which means the entire dataset was processed 10 times. In each epoch, a batch of 128 data points was used for training. The entire training process took approximately three minutes, with each epoch lasting about 16 seconds. Notably, as the epochs progressed, the accuracy of the model consistently improved while the loss decreased. This is a positive sign that the model was learning effectively without becoming overfitted.

Throughout the training phase, the model achieved high levels of accuracy across different datasets: the training dataset, the validation dataset, and the testing dataset. Specifically, after the completion of 10 epochs, the model achieved accuracy rates of 90.1%, 89%, and 90% on the training, validation, and testing datasets, respectively. This indicates that the model had become well-acquainted with the dataset and was capable of accurately classifying the gestures within it.

Fig. 11 illustrates the model's training progress over each epoch. This visualization focuses on key metrics: loss, accuracy, val_loss (loss on the validation dataset), and val_accuracy (accuracy on the validation dataset). These metrics are tracked and displayed after every epoch to provide insight into the model's performance.

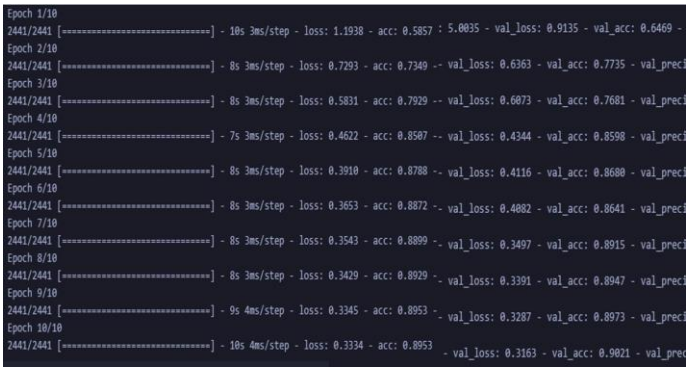


Fig. 11. Output of the linear classifier model's training process.

Here's a breakdown of what these metrics represent:

1) Loss: This metric indicates how much the model's predictions deviate from the actual values (labels) in the training dataset. It quantifies the error between predicted and actual values.

2) Accuracy: This metric represents the proportion of correctly classified data points in the training dataset. It shows how well the model is performing in terms of correctly predicting gestures.

3) Val_loss: Similar to the loss metric, val_loss measures the error between predicted and actual values, but it specifically pertains to the validation dataset.

4) Val_accuracy: Like accuracy, val_accuracy indicates the proportion of correctly classified data points, but on the validation dataset. It provides insight into how well the model generalizes to unseen data.

Fig. 12 and Fig. 13 provide a visual representation of the trends highlighted in Fig. 11, offering a closer look at the training and validation accuracy of the model as well as the corresponding training and validation loss throughout the 10 epochs. The x-axis on these figures denotes the number of epochs, while the y-axis illustrates the model's accuracy and loss.

From Fig. 12, it becomes evident that during the initial stages of training, the model's training accuracy starts at a relatively low point. At this early phase, the model lacks the understanding to accurately identify static hand gestures and is essentially making random guesses. However, as training progresses through each epoch, there is a steady and gradual improvement in accuracy. This trend signifies that the model is learning and becoming more adept at recognizing and distinguishing various hand gestures over time.

Examining Fig. 13, the graph portrays the trajectory of the model's loss on both the training and validation datasets across the epochs. Initially, during the early epochs, the training loss was notably high. This is attributed to the model's nascent stage, where it generates random and unrefined predictions, resulting in higher discrepancies between predicted and actual values. However, as the number of epochs advances, the training loss consistently diminishes. This decline indicates that the model is progressively honing its ability to accurately identify hand gestures within the training dataset.

Furthermore, the graph reveals the validation loss, which pertains to the model's accuracy in recognizing hand gestures on unseen validation data. As the epochs unfold, the validation loss follows a similar downward trajectory, signifying the model's enhanced capacity to generalize its learning from the training dataset to effectively identify hand gestures in new, previously unseen data.

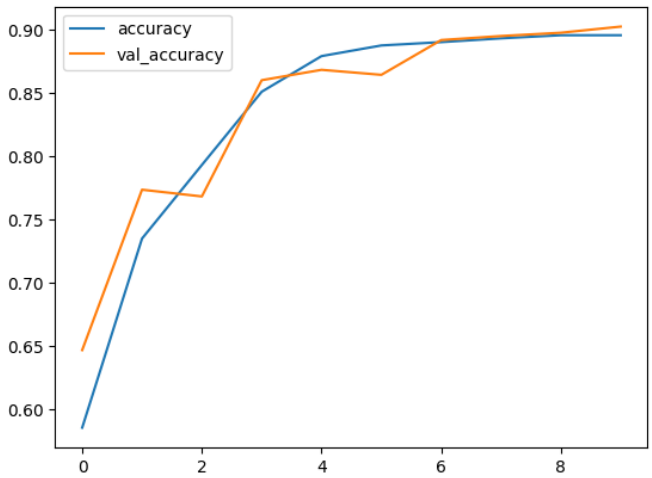


Fig. 12. Training and validation accuracy curves of the linear classifier model.

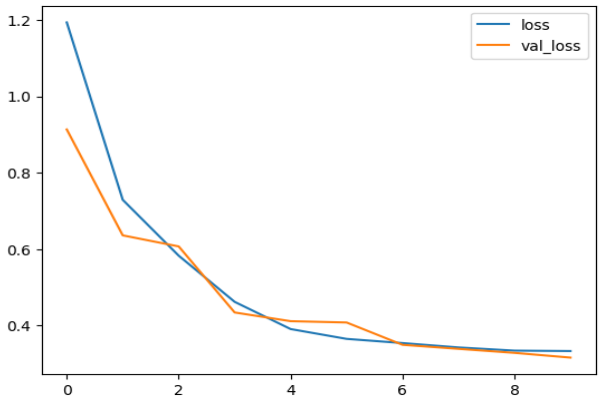


Fig. 13. Training and validation loss curve of the linear classifier model.

In essence, these graphical representations within Fig. 12 and Fig. 13 align with the narrative conveyed in Fig. 11, collectively providing a comprehensive insight into the model's iterative learning process. The figures underscore the model's journey from initial uncertainty and randomness to increasingly accurate and refined hand gesture recognition, ultimately attesting to its growing proficiency in this specific task.

The motion classifier model (LSTM) underwent training with a total of 200 epochs, where each epoch involved processing a batch of 64 data points. The entire training process took approximately 33 minutes, with each epoch lasting about 10 seconds. Similar to the linear classifier, the accuracy of the motion classifier increased while the loss decreased consistently with each successive epoch.

Throughout the training phase, the motion classifier model demonstrated high accuracy levels across different datasets: the training dataset, the validation dataset, and the testing dataset. Specifically, after the completion of the training process, the model achieved accuracy rates of 79.8%, 72%, and 70.1% on the training, validation, and testing datasets, respectively. This performance indicates that the model had become well-acquainted with the dataset and was proficient in correctly classifying different types of motions.

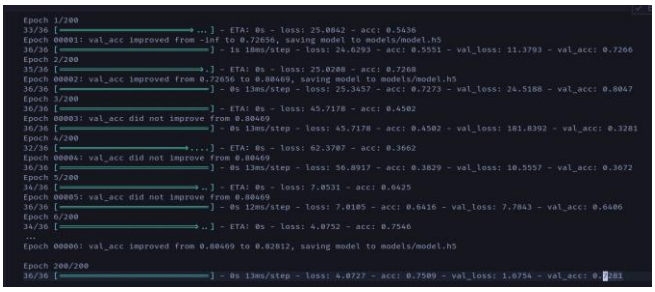


Fig. 14. Output of the motion classifier (LSTM) model's training process.

The training progress and performance of the motion classifier model are visualized in Fig. 14, 15, and 16. Fig. 14 provides an overview of the model's output during the training process, highlighting key metrics such as accuracy and loss across epochs. Fig. 15 and 16 depict the training and validation accuracy curves, as well as training and validation loss curves respectively, providing a graphical representation of the model's learning journey.

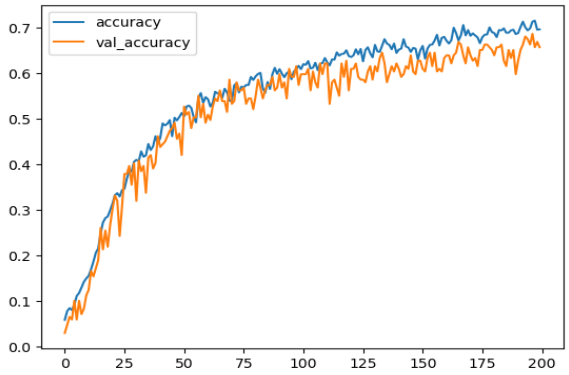


Fig. 15. Training and validation accuracy curve of the LSTM model.

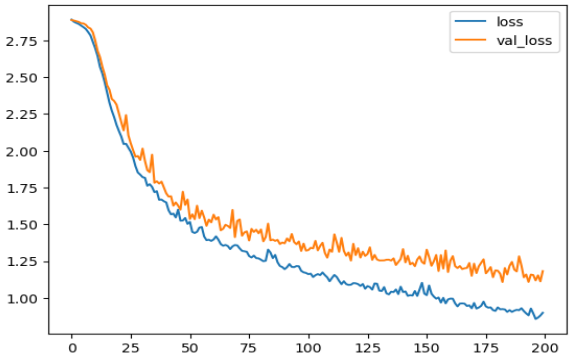


Fig. 16. Training and validation loss curves of the LSTM model.

In Fig. 15, it is observable that the accuracy initially starts at a lower point, similar to the linear classifier model. As training progresses over the epochs, the accuracy gradually improves, reflecting the model's increasing ability to recognize and classify motion patterns.

Fig. 16 displays the training and validation loss curves, illustrating a similar pattern observed in the linear classifier. At the start of training, the training loss is relatively high due to the model's random guesses. However, as the epochs advance, the training loss consistently decreases, indicating improved accuracy in recognizing motion patterns. The validation loss also follows a similar trajectory, indicating the model's enhanced ability to generalize its learning to unseen data.

B. Performance Index

Table I presents a comprehensive summary of how well the linear classifier model performed using various evaluation metrics. These metrics provide insights into the model's ability to accurately classify hand gestures.

- Accuracy: This metric measures the overall correctness of the model's predictions. The reported accuracy of 90% indicates that the model correctly classified 90% of the gestures in the dataset.
- Precision: Precision refers to the ratio of true positive predictions to the total number of positive predictions made by the model. In this case, the precision score of 92% implies that when the model predicts a gesture as positive (i.e., a certain hand gesture), it is correct 92% of the time.

- Recall: Recall, also known as sensitivity or true positive rate, calculates the ratio of true positive predictions to the total number of actual positive instances in the dataset. A recall of 90% indicates that the model correctly identified 90% of the actual hand gestures.
- F1-score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. An F1 score of 91% suggests that the model achieves a balanced trade-off between precision and recall.

TABLE I. PERFORMANCE OF THE LINEAR CLASSIFIER MODEL

Accuracy	*Precision*	*Recall*	*F1-score*
90%	92%	90%	91%

Fig. 17 visually represents the model's performance in terms of accuracy, precision, recall, and F1 score using a bar chart. This chart offers a clear overview of the model's strengths in terms of accuracy and precision, indicating its ability to make accurate predictions, particularly with high precision. However, it may miss some positive cases, as indicated by the relatively lower recall score.

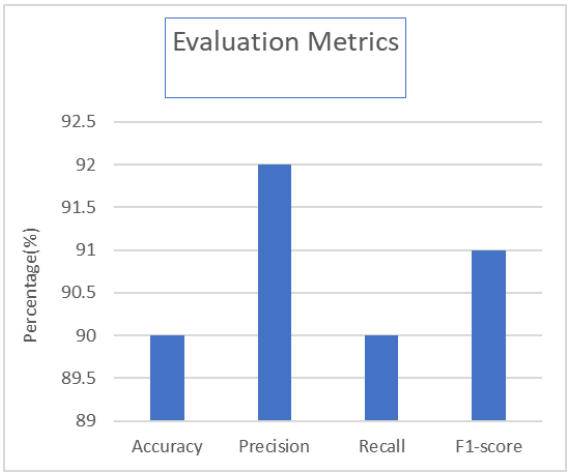


Fig. 17. Bar chart showing the linear classifier's performance.

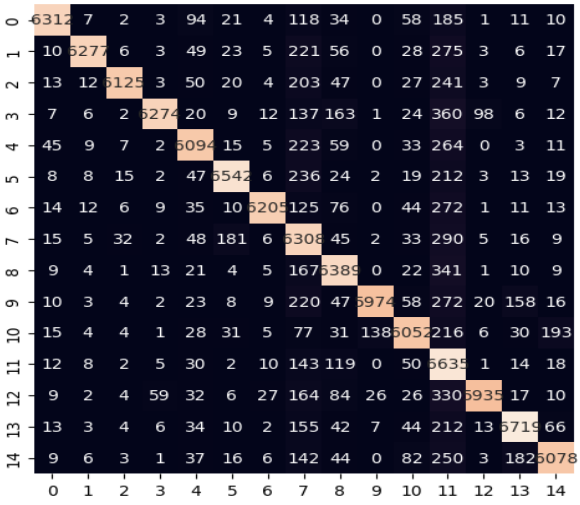


Fig. 18. The confusion matrix of the linear classifier model.

Fig. 18 and Fig. 19 present confusion matrices for the linear classifier model and the LSTM model, respectively. These matrices provide a visual depiction of the model's classification performance across different classes. They allow us to see how well the model correctly predicted each class and where it might have made errors.

Table II delves into the performance of individual classes. It provides a breakdown of precision, recall, and F1-score metrics for each class. For instance, for class 0, the precision score of 0.97 means that when the model predicted class 0, it was correct 97% of the time. The recall score of 0.92 indicates that the model correctly identified 92% of instances belonging to class 0. The F1-score of 0.94 represents a balanced measure of precision and recall for class 0.

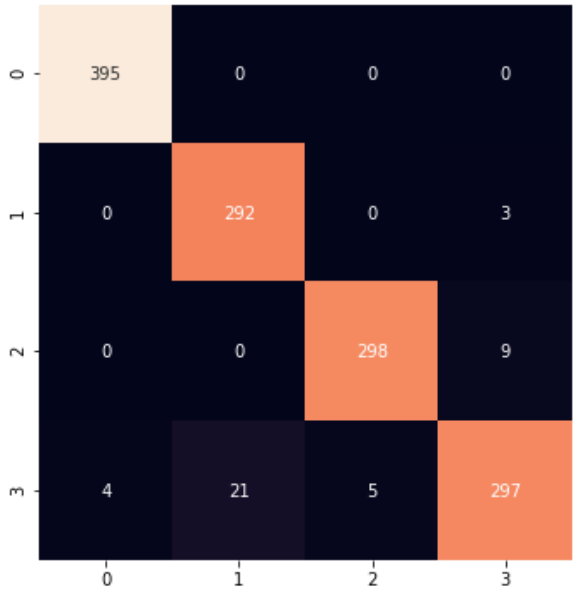


Fig. 19. The confusion matrix of the LSTM model.

TABLE II. PRECISION, RECALL AND F1-SCORE OF THE CLASSES

Class	*Precision*	*Recall*	*F1-Score*
0	0.97	0.92	0.94
1	0.99	0.90	0.94
2	0.99	0.91	0.94
3	0.98	0.88	0.93
4	0.92	0.90	0.91
5	0.95	0.91	0.93
6	0.98	0.91	0.94
7	0.73	0.90	0.81
8	0.88	0.91	0.90
9	0.97	0.88	0.92
10	0.92	0.89	0.90
11	0.64	0.94	0.76
12	0.97	0.88	0.93
13	0.93	0.92	0.92
14	0.94	0.89	0.91

V. User Implementation

A. The PowerPoint Selection

The interface permits users to choose which presentation file to open. The file browser approach is used to do this. The interfaces include a file browser which enables users to search the file system of their device for the desired PowerPoint presentation file. When selected, the slides pop up in the background as shown in Fig. 20.

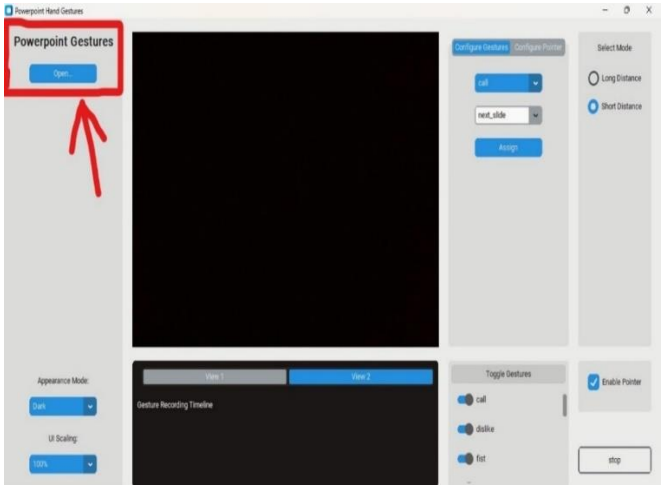


Fig. 20. Power point selection.

B. Distance Mode Selection

The user can select which mode is needed based on his/her distance from the system. A short distance is used in this operation. This can be visualized in Fig. 21.

C. Gesture Selection and Assignment

The interface allows the user to select gestures, and their functions and then assign them. In Fig. 22, the gesture used in this case is "call" and it is assigned to a function called "next_slide" which moves the slides page by page, the other 17 gestures in this model can also be assigned to other functions.

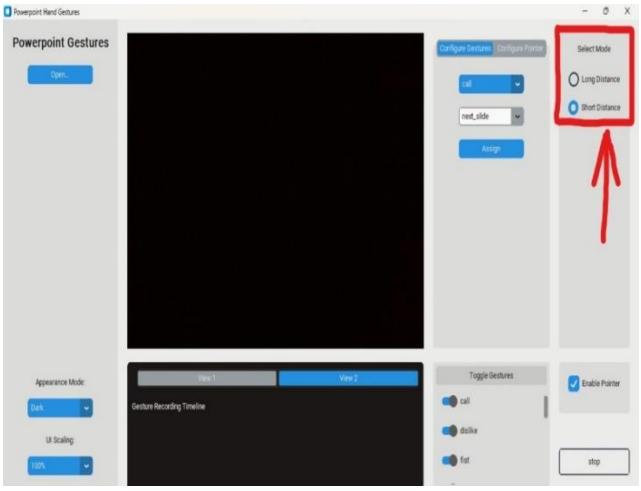


Fig. 21. Input distance mode.

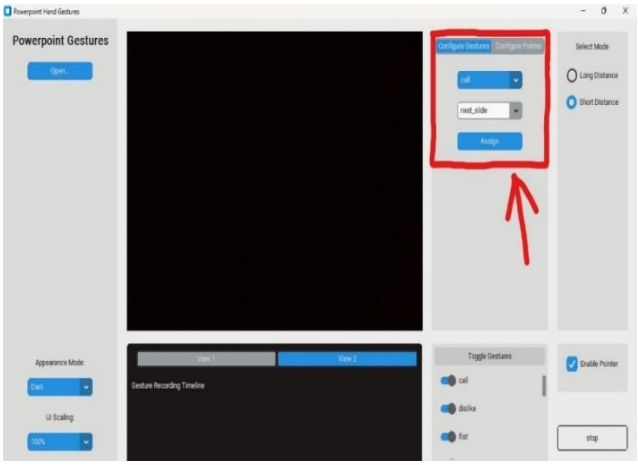


Fig. 22. Gesture selection and assignment.

D. Gesture Toggler

After assigning the gesture needed by the user, the user can turn on or off gestures based on their preferences as shown in Fig. 23, Fig. 24 And Fig. 25 shows that when the user toggles off a gesture, the gesture can no longer be recognized by the system and cannot perform its assigned function on the slides. When the gesture is toggled on, the system recognizes the gesture and the action assigned to it on the slides.

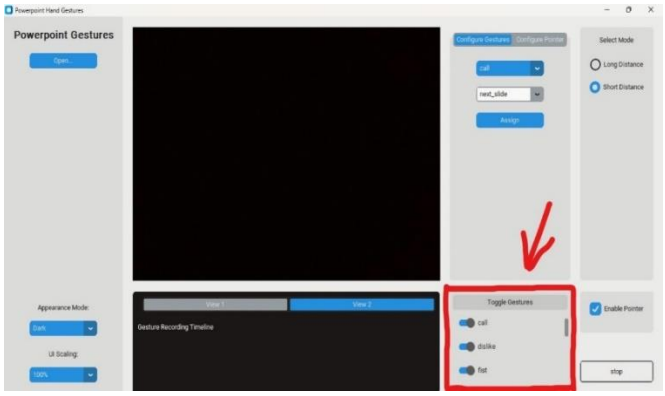


Fig. 23. Gesture toggler.

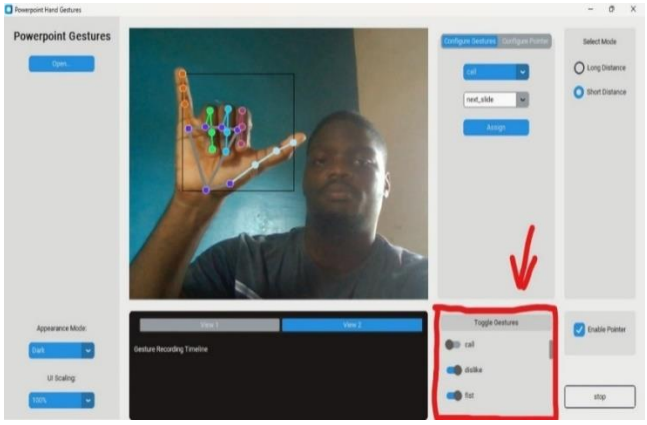


Fig. 24. Toggling OFF a gesture.

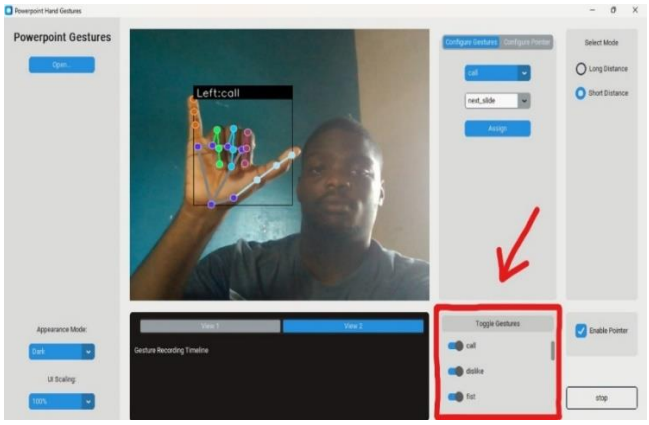


Fig. 25. Toggling ON a gesture.

E. Flow Chart

Fig. 26 shows the flow of the working of the package from the first stage to the end. This shows how the user should operate the software package.

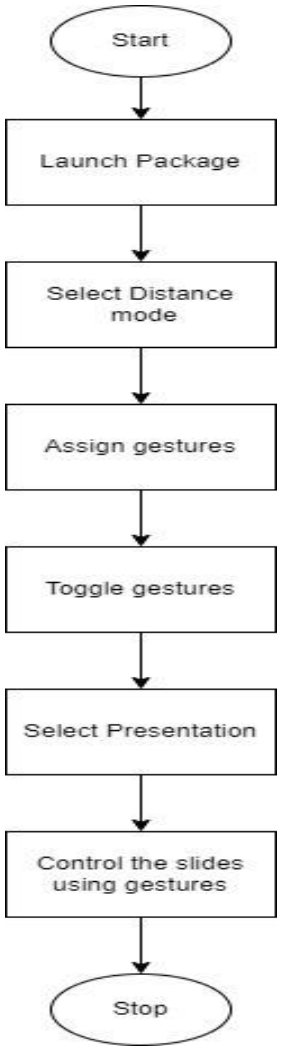


Fig. 26. Interface flowchart.

VI. Conclusion

In this study, we developed a software package that controls presentations using hand gestures. The system's performance was thoroughly examined through various evaluation metrics and visualization tools. Our findings highlight the effectiveness of the developed models in accurately identifying and classifying hand gestures.

The linear classifier model demonstrated impressive results, achieving an overall accuracy of 90%. This model's strong precision and recall scores further emphasize its ability to make accurate predictions while effectively identifying positive cases. The bar chart depicting the model's performance underscored its proficiency in precision, shedding light on its potential for minimizing false positive predictions. The confusion matrix provided valuable insights into the model's classification patterns. By analyzing true positives, true negatives, false positives, and false negatives, we gained a clearer understanding of where the model excelled and where improvements could be made. This granular analysis helps guide future refinements and optimizations of the system. Additionally, the LSTM model, designed to capture temporal patterns in motion data, demonstrated its efficacy in motion detection. While achieving lower accuracy compared to the linear classifier model, the LSTM model's performance remained solid, with an accuracy of 70.1% on the testing dataset. This model's potential for capturing long-term dependencies and recognizing complex motion patterns positions it as an asset for more nuanced applications. The study also addressed challenges related to distance by implementing Long-Distance Mode, which enhanced the system's adaptability to users situated far from the camera. This innovation showcases the system's robustness in accommodating varying scenarios and environments.

In conclusion, the developed system, consisting of a linear classifier and LSTM model, exhibits strong potential for real-time and accurate hand gesture identification. The combination of efficient classification and temporal analysis provides a comprehensive approach to gesture recognition. As a result, this system holds promise for a wide range of applications, from interactive presentations to virtual reality interfaces, enhancing user experience and interaction. Further research and optimization can propel this system towards even greater accuracy and utility in real-world settings.

Acknowledgment

The authors would like to acknowledge Covenant University for the funding of this paper.

References

[1] Y. Zhu, Z. Yang, and B. Yuan, “Vision-based hand gesture recognition,” in Proceedings of International Conference on Service Science, ICSS, 2013, pp. 260–265. doi: 10.1109/ICSS.2013.40.

[2] D. Ballotta, “Public speaking and presentations a critical review: The caring speaker,” 2010, [Online]. Available: http://hdl.handle.net/10071/2068.

[3] Alausa, D. W., Adetiba, E., Badejo, J. A., Davidson, I. E., Obiyemi, O., Buraimoh, E., ... & Oshin, O. (2022). Contactless palmprint recognition system: a survey. IEEE Access, 10, 132483-132505.

[4] S. E. Hollingsworth, K. Weinland, S. Hanrahan, M. Walker, E. Elwood, and M. Linsenmeyer, "Introduction to speech communication." Oklahoma State University Libraries, 2021.

[5] C. Roberto, "Development of a Hand-Gesture Recognition System for Human-Computer Interaction," 2014.

[6] Alausa, D. W., Adetiba, E., Badejo, J. A., Davidson, I. E., Akindeji, K. T., Obiyemi, O., & Abayomi, A. (2023, January). PalmMatchDB: An On-Device Contactless Palmprint Recognition Corpus. In 2023 IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA) (pp. 318-325). IEEE.

[7] A. Daly, J. Bremmer, and H. Roodenburg, A Cultural History of Gesture, vol. 39, no. 1. Cornell University Press Ithaca, NY, 1995. doi: 10.2307/1146410.

[8] Okokpujie, K., & Apeh, S. (2020). Predictive modeling of trait-aging invariant face recognition system using machine learning. In Information Science and Applications: ICISA 2019 (pp. 431-440). Springer Singapore.

[9] T. D'Orazio, R. Marani, V. Renò, and G. Cicirelli, "Recent trends in gesture recognition: How depth data has improved classical approaches," Image Vis. Comput., vol. 52, pp. 56–72, 2016, doi: 10.1016/j.imavis.2016.05.007.

[10] L. Benke, "Gesture, intimacy, and violence in contemporary artists' books," Reconstr. Stud. Contemp. Cult., vol. 16, no. 1, 2016.

[11] C. Darwin and F. Darwin, The expression of the emotions in man and animals. Oxford University Press, USA, 2009. doi: 10.1017/CBO9780511694110.

[12] A. Kendon, "Pragmatic functions of gestures some observations on the history of their study and their nature," Gesture, vol. 16, no. 2, pp. 157–175, 2017.

[13] J. Y. Mensah and M. J. Nabie, "The Effect of PowerPoint Instruction on High School Students' Achievement and Motivation to Learn Geometry.," Int. J. Technol. Educ., vol. 4, no. 3, pp. 331–350, 2021.

[14] J. Ruesch and G. Bateson, Communication: The social matrix of psychiatry. Routledge, 2017. doi: 10.4324/9781315080932.

[15] A. Kendon, "Introduction: Current issues in the study of 'nonverbal communication,'" in Nonverbal Communication, Interaction, and Gesture: Selections from SEMIOTICA, De Gruyter Mouton, 2010, pp. 1–53. doi: 10.1515/9783110880021.1.

[16] A. Kendon, "Gesture and anthropology: notes for a historical essay," Gesture, vol. 18, no. 2/3, pp. 142–172, 2019.

[17] A. Mehrabian, "Communication without words," in Communication Theory: Second Edition, Routledge, 2017, pp. 193–200. doi: 10.4324/9781315080918-15.

[18] P. Ekman and W. V. Friesen, "Hand movements," in Communication Theory: Second Edition, Routledge, 2017, pp. 273–292. doi: 10.4324/9781315080918-21.

[19] C. F. Hockett, "Language, mathematics, and linguistics," in Language, mathematics, and linguistics, De Gruyter Mouton, 2019.

[20] M. Weinert, "On the Contemporary Theories of the Development of Human Language," Acad. J. Mod. Philol., no. 12, pp. 229–238, 2021.

[21] J. M. Power, "Historical Linguistics of Sign Languages: Progress and Problems," Front. Psychol., vol. 13, 2022, doi: 10.3389/fpsyg.2022.818753.

[22] A. M. Borghi, F. Binkofski, C. Castelfranchi, F. Cimatti, C. Scorolli, and L. Tummolini, "The challenge of abstract concepts," Psychol. Bull., vol. 143, no. 3, pp. 263–292, 2017, doi: 10.1037/bul0000089.

[23] J. C. Nunez, R. Cabido, J. J. Pantrigo, A. S. Montemayor, and J. F. Velez, "Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition," Pattern Recognit., vol. 76, pp. 80–94, 2018.

[24] J. H. Sun, T. T. Ji, S. Bin Zhang, J. K. Yang, and G. R. Ji, "Research on the Hand Gesture Recognition Based on Deep Learning," in 2018 12th International Symposium on Antennas, Propagation and EM Theory, ISAPE 2018 - Proceedings, 2019, pp. 1–4. doi: 10.1109/ISAPE.2018.8634348.

[25] K. V. Eshitha and S. Jose, "Hand Gesture Recognition Using Artificial Neural Network," in 2018 International Conference on Circuits and Systems in Digital Enterprise Technology, ICCSDET 2018, 2018, pp. 1–5. doi: 10.1109/ICCSDET.2018.8821076.

[26] A. Ikram and Y. Liu, "Skeleton-based dynamic hand gesture recognition using LSTM and CNN," in ACM International Conference Proceeding Series, 2020, pp. 63–68. doi: 10.1145/3421558.3421568.

[27] H. Y. Chung, Y. L. Chung, and W. F. Tsai, "An efficient hand gesture recognition system based on deep CNN," in Proceedings of the IEEE International Conference on Industrial Technology, 2019, vol. 2019-Febru, pp. 853–858. doi: 10.1109/ICIT.2019.8755038.

[28] I. Dhall, S. Vashisth, and G. Aggarwal, "Automated hand gesture recognition using a deep convolutional neural network model," in Proceedings of the Confluence 2020 - 10th International Conference on Cloud Computing, Data Science and Engineering, 2020, pp. 811–816. doi: 10.1109/Confluence47617.2020.9057853.

[29] P. Parvathy, K. Subramaniam, G. K. D. Prasanna Venkatesan, P. Karthikaikumar, J. Varghese, and T. Jayasankar, "Development of hand gesture recognition system using machine learning," J. Ambient Intell. Humans. Comput., vol. 12, no. 6, pp. 6793–6800, 2021.

[30] A. Mujahid et al., "Real-time hand gesture recognition based on deep learning YOLOv3 model," Appl. Sci., vol. 11, no. 9, p. 4164, 2021.