

Dynamic Gesture Recognition Using LSTM for Real-time Indian Sign Language Prediction

Anirudh Singh Rautela
Symbiosis Institute of Technology
Computer Science Department
Pune, India
Anirudh.Rautela.Btech2021

Barish Priyam Chetia
Symbiosis Institute of Technology
Computer Science Department
Pune, India
Barish.Chetia.Btech2021

Esam Ashfaq
Symbiosis Institute of Technology
Computer Science Department
Pune, India
Esam.Ashfaq.Btech2021

Ishaan Bhadrike
Symbiosis Institute of Technology
Computer Science Department
Pune, India
Ishaan.Bhadrike.Btech2021

Pranay Chauham
Symbiosis Institute of Technology
Computer Science Department
Pune, India
Pranay.Chauhan.Btech2021



Abstract— This paper presents an approach towards real-time dynamic gesture recognition both using the combination of LSTM networks and the data collection framework based on cameras. It captures gesture via camera, extracts keypoints of the hand using MediaPipe, and trains an LSTM model to classify gestures according to predefined actions. Therefore, it increases accuracy as well as allows for interaction to be applied in the communication through gesture.

Keywords—Real-time gesture recognition, CNN-LSTM hybrid model, dynamic hand gestures, keypoint extraction, MediaPipe framework, human-computer interaction (HCI), sign language recognition, temporal feature extraction, accessibility

INTRODUCTION

This includes hand movements in device gestures being used to interpret them either as a command or just some kind of non-verbal communication. Though rapidly increasing within its areas through advances of machine learning and computer vision, there is yet significant challenges in the direction of real-time dynamic gesture recognition. Most of the currently available approaches would depend on either CNN-based image-based gesture detection or RNNs to be able to account for time information. Applying these models to real-time recognition tasks would result in several trade-offs between latency and accuracy.

More recent studies have demonstrated that CNNs have been very effective for fixed images in classification tasks. However, they don't learn the movement flow with time in dynamic sequences. Hence, this work combines an LSTM network with feature extraction via MediaPipe to capture hand detailed landmarks and body posture; we attempt to improve upon the model's capacity in distinguishing many gestures through an improved temporal accuracy in the work.

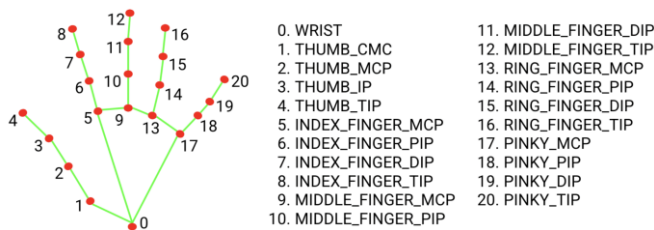


Figure (1) Mediapipe hand detection keypoints

The application of our model includes real-time support for sign language recognition, which can be revolutionary in accessibility in different contexts. The study focuses on a

dataset of dynamic gestures and explores how sequential learning models can capture the subtlety of hand movements more effectively than traditional models. The hand movements are tracked with the media-pipe hand detection module for which key relative co-ordinates are tracked as shown in [Figure (1)][26]. We aim to deliver a fast and accurate gesture recognition solution suitable for real-time deployment through a combination of data preprocessing, efficient feature extraction, and sequence learning.

LITERATURE REVIEW

The application of Convolutional Neural Networks (CNNs) has significantly enhanced the recognition of static gestures. Methods like color segmentation and data augmentation have demonstrated high accuracy, even under challenging background conditions, making them suitable for tasks requiring stationary gesture recognition [1][2]. However, CNNs struggle with dynamic gestures as they fail to capture temporal patterns, which are crucial for recognizing movements across time [3][4]. To address this, Long Short-Term Memory (LSTM) networks are integrated due to their ability to capture sequential dependencies, especially in dynamic gestures, such as those used in sign language [4][5].

Hybrid models that combine CNN and LSTM layers have been developed to address the need for both spatial and temporal feature extraction. These models employ CNNs for spatial data and LSTMs for sequence processing, proving effective in recognizing complex gestures. However, they present increased computational demands, posing challenges for real-time applications [5][6]. Further advancements have been made with 3D-CNNs and LSTM networks, reducing computational loads while effectively handling spatial and temporal features, enabling real-time functionality on intricate gesture datasets [6][7].

MediaPipe's holistic tracking system is widely used in gesture recognition as it offers accurate keypoint extraction for hand and body landmarks. By delivering precise, low-latency tracking, MediaPipe enhances machine learning models, making it ideal for real-time gesture recognition tasks [1][7]. When combined with LSTM networks, MediaPipe further strengthens model robustness, particularly in applications that require accurate hand movement recognition [7][8].

Gesture recognition holds significant potential for accessibility, such as real-time gesture interpretation for

hearing-impaired individuals in both physical and virtual spaces, requiring high accuracy across diverse backgrounds and lighting conditions. This highlights the importance of creating user-independent systems that perform consistently across various scenarios [1][8].

Recent studies support the use of LSTM networks for sequential data in gesture recognition. For example, Kim et al. [9] proposed a dynamic hand gesture recognition model using MediaPipe, Inception-v3, and LSTM, showing improvements in gesture recognition accuracy. Similarly, Kumar et al. [10] explored the combination of MediaPipe and CNNs for real-time American Sign Language (ASL) gesture recognition. Lee and Kim [11] presented LM-Net, a dynamic gesture recognition network that captures long-term temporal context, enhancing performance in complex gestures. Studies by Chen et al. [12] and Kumar et al. [13] further illustrate the benefits of CNN-LSTM integration, particularly for tasks like drone control and dynamic hand gestures, where both spatial and temporal patterns are essential.

Other advancements include hybrid models like CNN-BiLSTM, which have been used for sign language recognition with higher accuracy [15][21]. Further, researchers like Singh and Yadav [24] and Ahn and Han [25] investigated LSTM-based models for real-time applications, such as augmented reality (AR) and multimodal gesture recognition, demonstrating LSTM's efficacy in dynamic gesture prediction and multimodal environments.

In conclusion, literature shows that combining MediaPipe for landmark extraction with LSTM networks is promising for real-time gesture recognition. Although CNN and hybrid models provide benefits, they face limitations with sequential data, making LSTM an essential component for achieving high accuracy in dynamic gesture recognition systems.

THEORETICAL FRAMEWORK

Dynamic gesture recognition involves a more complex setup such as real-time video capture, high-precision keypoint extraction, and sequential data processing. This is aimed at handling complex gestures through the inference of slight movements and positions of body parts such as hands and upper bodies. The core backbone of this approach leans on MediaPipe's holistic tracking model and Long Short-Term Memory (LSTM) neural networks. Feature is extracted from the dataset and passed into the trained model which on receiving processed real time input from webcam generates a gesture output based on probability of prediction [Figure (2)]. Each of these modules works distinctly in developing a system that correctly perceives gestures in realistic environments with any background or illumination and noise.

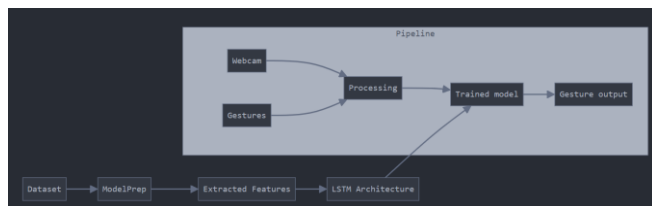


Figure (2) Flow of data

A. MediaPipe Holistic Model

First and foremost, the holistic tracking system that MediaPipe gives is a very important factor in this framework. It is an advanced computer vision model that can map landmarks and detect them across several body parts from hands to face and pose.

The hands are the most fundamental gestures in gesture recognition because the communication weight is carried heavily through them. MediaPipe's holistic model is well able to identify hand and pose landmarks with a much better accuracy and also provides the benefits of real-time tracking that has minimal latency, but in gesture dynamics, changing speed of hand position orientation comes into play.

B. Keypoint Extraction

The holistic model will be adapted such that the hand and pose landmarks mainly form the basis on which extraction of critical points might be focused. Consistent keypoints are produced by the holistic model for both left hand and right hand and thus give the system a capability to handle symmetrical or mirrored gestures, characteristic of gestures in sign languages and communication. The extraction of keypoints is the core of the gesture recognition system, translating visual data into a numerical format for machine learning. Keypoints are specific positions on the hands and pose, tracked over a sequence of frames that captures the spatial configuration of the body as it moves.

C. LSTM Neural Network

For every hand and body movement, gestures have a definite number of frames through which these actions are coordinated. Frames assembled together are further used in the process for recovery with 3D coordinates, including x, y, and z positions of landmarks found. These are taken and filled into arrays that are readily fed into the model during gesture recognition. After keypoints are extracted, the system then processes the sequence of landmarks over time using an LSTM neural network. LSTMs are unique from other networks because they can remember previous inputs; this makes them ideal for data that is temporal in nature, such as gesture sequences. The LSTM can remember previous frames. Therefore, the model will be able to recognize flow and rhythm in gestures like a slow lift of the hand or a slight bend of fingers. This allows the system to distinguish between gestures that look similar but have different motions over time. With LSTMs, the model can reach a very good accuracy in gesture classification, even in complex scenarios.

D. Real-Time Prediction with Thresholding

To have a smoother flow, the system involves a real-time prediction mechanism based on a threshold-based approach. It utilizes the combination of a cooldown period along with skip-frame techniques in stabilizing the predictions where the system will not reproduce similar or wrong results. A cooldown counter for consistency in output will prevent further predictions when a gesture is detected. High confidence, predictions greater than the set probability threshold, increase the robustness of the system while avoiding false positives. In live settings such as a public space or interactive virtual environments, real-time settings can ensure a proper run.

I. Static Signs

A. Keypoint Extraction

The static gesture recognition system uses MediaPipe's keypoint extraction. This is a machine learning library optimized for real-time landmark detection. The core of the extraction lies within the preprocessing.py script, which actually uses the Hands model from MediaPipe to detect hand landmarks or keypoints. Every hand landmark is mapped to the corresponding x, y coordinates that are spatial positions in each image frame.

1) Image and Landmark Processing:

The function calc_landmark_list calculates the positions of landmarks for each hand on an image from raw landmark data from MediaPipe into coordinates in pixels. For each landmark, the relative position is necessary for the interpretation of the hand gesture and capture of hand orientation, finger position, and shape variations.

2) Normalization:

The pre_process_landmark function normalizes all landmark's coordinates to be relative to a base point, commonly the wrist; this eliminates the influence of size and orientation on hands. Another thing, the landmarks were flattened to a one-dimensional array for the efficient input in the neural network to further spatial pattern recognition.

3) Data Augmentation:

This will improve generalization since the function augment_image horizontally flips the image, mirroring gestures to create more samples for training.

B. Data Collection

Similarly, keypoint is extracted while saving to CSV for training in preprocessing.py and each processed image, the keypoints are labeled with their id corresponding to the certain static gesture that is expressed using either letters or numbers.

1) Gesture Labels:

Each move obtains a name-for example, "A" or "1"-designating the class for further classification. Names correspond to class names used during the training period.

2) Data Logging:

Keypoints with corresponding gesture labels are stored in structured form in keypoint.csv [Fig. (3)], resulting in an orderly dataset on which the models will train. This system facilitates ready access and systematic handling of samples for training.

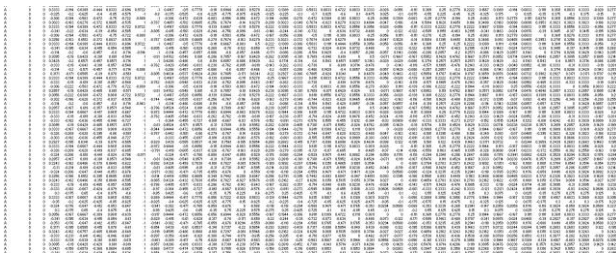


Figure (3) Head csv file with the logs

C. Dataset

To collect images for the static gesture recognition system, the target scope aims to capture all significant points of hand positions depicting any letter from A-Z or numbers from 0 to 9. This will thus ensure that diversity of such gestures ensures generalisability with cross-use, as in different users and possibly different conditions.

1) Data Collection and Structure

There are about 1,000 images per gesture class. Thus, the total is 36,000 images. Each image is a unique gesture corresponding to a specific letter from A to Z or a number from 0 to 9. Thus, this well-balanced dataset ensures having enough samples for each class to train effectively in a multi-class classification model.

2) Image Preparation and Preprocessing

Every picture is capturing hand in one specific static pose. All the images were captured with rather different illumination conditions as well as changes in hand position for adding augmentations. The images were pre-processed as follows:

- Keypoint Extraction: The MediaPipe Hands model extracts the hand landmarks from an image and encodes them as structured arrays of x, y coordinates for each keypoint.
- Normalization Keypoints are normalized relative to a base point (wrist position). Thus, the model is not sensitive to slight hand size and orientation movements.
- Data Augmentation: To increase generalization, all images were flipped horizontally, and the effective size of the dataset was doubled to 2,000 samples per class. This augmentation ensures that the model identifies both left and right hands of gestures.

3) Data Organization

The dataset has directories labeled with class labels for each unique gesture. There would be efficient loading and retrieval of data due to this structure of the directories:

- Root Directory: The main directory (data/) holds all gesture images.
- Subdirectories: Each gesture class (e.g., A, B, 1, 2) has its own subfolder within the main directory. Within each class folder, individual image files are named sequentially (e.g., data/A/0.jpg, data/A/1.jpg, etc.) for streamlined access.
- CSV Logging: Keypoints and labels are stored in keypoint.csv, each row per image's keypoints with a corresponding gesture label. This structure allows easy batching and model input during training.

4) Consistency and Quality Control

- Root Directory: The main directory (data/) holds all gesture images.
- Subdirectories: Each gesture class (e.g., A, B, 1, 2) has its own subfolder within the main directory. Within each class folder, individual image files are named sequentially (e.g.,

data/A/0.jpg, data/A/1.jpg, etc.) for streamlined access.

- CSV Logging: Keypoints and labels are stored in `keypoint.csv`, each row per image's keypoints with a corresponding gesture label. This structure allows easy batching and model input during training.

D. Model Architecture

A densely connected neural network model is designed to classify static gestures based on hand landmarks. The architecture was implemented using TensorFlow and Keras, structured to recognize spatial patterns within hand landmarks.

1) Model Layers:

- Input Layer: It accepts preprocessed arrays of hand landmarks as input.
- Hidden layers: There are multiple fully connected dense layers that learn the high-level spatial patterns over static hand positions. It has dropout layers to regularize and reduce overfitting.
- Output Layer: Softmax -activated dense layer produces class probabilities for each input corresponding to some gesture.

2) Compilation:

- Input Layer: It accepts preprocessed arrays of hand landmarks as input.
- Hidden layers: There are multiple fully connected dense layers that learn the high-level spatial patterns over static hand positions. It has dropout layers to regularize and reduce overfitting.
- Output Layer: Softmax -activated dense layer produces class probabilities for each input corresponding to some gesture.

E. Training Process

The model is further trained with their extracted hand landmarks and corresponding labels in a supervised manner. Data is then split into training and testing sets to enable generalizable learning.

1) Training Parameters:

Training: It trains on 50 epochs with a batch size of 128, and tracks its performance on an 80-20 validation split.

2) Early Stopping:

Training with validation : Early stopping based on validation loss Prevents overfitting, by halting training when no further improvements have been detected.

F. Model Evaluation

The performance of the model is checked on the test set on how well it identifies the unseen static gestures. Performance metrics include accuracy, precision, recall, and F1-score.

1) Prediction and Classification:

The model evaluates class probabilities for each test sample and predicts classes based on the probability distribution over gesture classes. It then compares classes against the corresponding ground truth labels to compute evaluation metrics.

2) Metric Calculation:

Standard evaluation metrics are accuracy, precision, recall, and F1-score providing a full view of the model capability in classifying stationary gestures.

G. Real-Time Prediction

The script `realtime.py` loads the trained model for real-time gesture recognition through a webcam interface. This model essentially allows people to see the predicted gestures overlaid onto the video feed.

1) Gesture Detection Pipeline:

- Key Point Extraction. During processing of the frame, MediaPipe captures keypoints.
- Normalization: Keypoints are normalized just like in the training setup for fair recognition.

2) Gesture Prediction and Visualization:

During computation, the model infers gesture in real time; with recognized gestures appearing onscreen to visually provide an immediate feed-through loop for live user interactivity.

II. Dynamic Gesture

A. Keypoint Extraction

`KeypointsExtraction.py` The main script involved in transforming the raw video frames into meaningful data is that of MediaPipe's holistic model, which can detect hand and pose landmarks in real time. It maps the detected landmarks onto x, y, and z coordinates within each frame, thereby representing the spatial position of each key point on the hands and upper body.

Keypoint extraction would hold two principal functions which are `image_process` and `keypoint_extraction_with_position`. The first one, the `image_process` would prepare every frame for processing by MediaPipe, while the latter, `keypoint_extraction_with_position` would draw out the exact location of each landmark. Therefore, this would represent the relative position of hand within the frame that would give a spatial context. For example, a raised hand, bent finger, or other type of hand gesture would be possible by the specific patterns in these coordinates. It converts all these coordinates into a single array for effective storage of spatial data about every gesture sequence.

B. Data Collection

`DataCollection.py` will be the most central part of collecting gesture data for model training. Within this, sequences of frame captures through a webcam are being taken. The process is designed to create labeled datasets for each gesture that trains the gesture recognition model. MediaPipe Holistic has been added to the system to capture all major key and significant hand and body landmarks in the process in order to track motion and positioning better for gestures.

1) Setup and Initialization

a) Gesture Labels:

- A list of gesture labels is defined, which are later used to denote class names for model classification.

b) Directory Structure:

- The PATH variable defines the root folder for the storage of data. For every action of gestures, it creates a subfolder that stores all the recorded sequences.
- For example, frames of each sequence are stored in distinguished directories in the folder structure data/Hello/0, data/Hello/1, etc, so that they can be accessed easily at training time during analysis of data.

c) Camera Initialization:

- The code checks if the camera is accessible (cap.isOpened()). If not, an error message appears, terminating the script gracefully.
- It verifies whether harvesting data can be done without any interruption considering camera unavailability.

2) Data Collection Process

Each gesture is captured in a controlled sequence of frames (20 frames per sequence in this configuration, with every frame recorded and labeled as a unique part of the gesture sequence).

a) Recording Control:

- When the script asks the user to begin recording a particular gesture, the space bar is pressed to initiate the recording of the gesture and sequence.
- This setup allows manual control; it gives user time to prepare the gesture and ensures only intended frames are being captured, which reduces noise in datasets.

b) Frame Capture and Processing:

- For every frame, MediaPipe Holistic uses `image_process(image, holistic)` for processing a webcam image so that results with landmarks are given.
- The script overlays landmark points on the camera feed using `draw_landmarks(image, results)`, which gives the user visual feedback to achieve better accuracy.

c) Keypoint Extraction with Position:

- For each frame, `keypoint_extraction_with_position(results, image.shape)` retrieves the landmarks with coordinates x, y, and z based on the size of the image.
- The spatial context in these movements is greatly important to the model for it to get the spatial structure of gestures.

d) Frame Saving:

- Each frame's keypoints are saved as a .npy file, capturing the landmarks for each sequence of the gesture.
- It is nimble and easy to handle since separate frames keep it separated and certain frames can be recovered or left out without altering other data.

e) Error Handling and Exit Options:

- It has 'q' to quit recording to abort the program at any moment from the recording making the data capture incomplete.
- This flexibility allows the user to abort and restart specific gestures if errors occur, promoting a clean dataset.

This just leads to an ordered, homogeneous dataset of gesture sequences of equal frame counts, with the model able to train nicely and get truly effective, well-labeled inputs.

C. The Dataset:

The resultant dataset will be a well-organized collection of labeled gesture sequences, structured to facilitate effective model training. Each gesture is recorded in sequences of frames, with each sequence stored in its respective subdirectory. The dataset is designed with the following attributes:

1) Structure and Storage:

- Root Directory: The main folder (data/) contains all gesture data, with each gesture type organized under subfolders.
- Subdirectories: Each gesture (e.g., "Hello") has its own folder, further divided into sequential subdirectories (data/Hello/0, data/Hello/1, etc.) where each subdirectory represents an individual gesture instance.
- Frame Data: Each sequence is made up of 20 frames, each stored separately to maintain consistency across all gesture sequences.

2) Data Consistency:

- Homogeneity: Each gesture has an equal frame count, creating a uniform dataset that is easier to batch and feed into the model.
- Coordinate Arrays: Each frame's landmarks are stored as .npy files, containing a structured array of x, y, and z coordinates. This format enables quick loading and processing without reformatting.

3) Labeling and Classification:

- Gesture Labels: Every gesture sequence is labeled, providing clear class names for each gesture type, which serves as the target variable for the model.
- Spatial Context: The coordinates in each frame capture the relative positioning of keypoints, offering a spatial context that the model can leverage to distinguish gestures based on body and hand movements.

4) Error Handling and Flexibility:

- Manual Recording Control: Users can initiate and end recordings, reducing unwanted noise in the dataset by ensuring only intentional gestures are recorded.
- Error Tolerance: Any interrupted or erroneous recordings can be restarted, supporting a clean, high-quality dataset.

```
[
  [0.45, 0.77, -0.05], # Right wrist
  [0.48, 0.72, -0.04], # Right thumb CMC
  [0.50, 0.68, -0.03], # Right thumb MCP
  [0.53, 0.65, -0.04], # Right thumb IP
  [0.55, 0.63, -0.03], # Right thumb tip
  ...
  [0.60, 0.85, -0.01], # Left shoulder
  [0.58, 0.90, 0.0], # Left elbow
  ...
]
```

Figure (4) A sample npy file storing relative coordinates wrt the frame

Each array captures the spatial position of landmarks, giving a snapshot of the gesture in that frame [Figure (4)]. When saved as .npy, this array can be easily loaded in Python

This organized and labeled dataset of gesture sequences, with consistent spatial context, will enable the model to learn and distinguish various dynamic gestures effectively. The captured keypoint arrays provide rich data for both training and validation, setting a solid foundation for dynamic gesture recognition.

D. Model Training

The ModelTraining.py script develops a CNN-LSTM hybrid model, which is supposed to extract both spatial features, that is, from hand landmark coordinates, and temporal patterns across frames, which are essential for accurate gesture classification. Such an architecture is beneficially suited for dynamic gestures, as it captures local spatial dependencies while learning sequential patterns. Here's a detailed explanation of each layer and what they do in the model to handle gestures represented by hand landmarks.

1) Model Architecture

The architecture uses a Sequential model from Keras as follows [Figure (5)].

a) Input Layer:

Shape: (frames, 126), representing the number of frames that make up each gesture sequence, where 126 depicts the flattened 21 keypoints for every hand, x, y, and z coordinates, respectively.

b) Conv1D Layer:

The given Conv1D layer of 64 filters with kernel size 3 extracts spatial patterns from each frame of the input hand landmarks.

Activation: relu, to introduce non-linearity so the model learns intricate spatial features across all keypoints.

c) MaxPooling1D Layer:

This layer reduces the spatial dimension, thus enabling the model to focus on key spatial features while at the same time minimizing computational load.

Pooling Size: 2, which will reduce the dimensionality and help in generalization.

d) First LSTM Layer:

Units: 32, return_sequences = True so that the next LSTM layer receives the entire sequence of processed features.

Activation: relu, which enables the LSTM to keep information regarding sequential dependencies of the gesture.

e) Second LSTM Layer:

Units: 64. Final Summary of the entire sequence. Outputs a distilled representation that captures the time evolution of the gesture.

f) Dense Output Layer:

Purpose of softmax-activated dense layer is the classification of gesture into one of the pre-defined gesture classes.

Neurons: The number of neurons maps directly to the number of gesture classes and provides a probability distribution for classification.

2) Model Compilation

The model is compiled with:

- Optimizer: Adam, and it efficiently manages a dynamic learning rate for smooth convergence.
- Loss Function: categorical_crossentropy: for multi-class classification.
- Metrics: accuracy, allows the model to keep track of its performance in classifying gestures during training.

3) Training Process

The dataset, containing hand landmark sequences, is now loaded and split into training and testing sets. The model will go through 100 epochs so that it could refine its parameters toward better accuracy in recognizing gesture sequences.

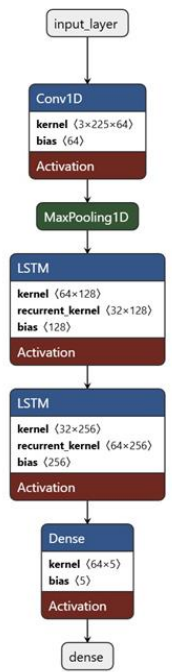


Figure (5) Model Architecture

4) Model Evaluation

Then, it evaluates the trained model on the test set to know about the accuracy of the model in real-world scenarios. Predictions are generated, and accuracy is calculated by comparing the predicted labels with actual test labels.

E. Real Time Prediction

RealTime.py module is dealing with actual prediction of gesture in a real-time way by getting loaded the model that has learned with all kinds of poses along with opening the Webcam in the screen. Minimize the prediction stability also there with cool down mechanisms with a few frames ahead from the frame where keypoints is captured as the whole 20 frames. Output comes only at highly confident keypoints through gestures prediction. These enhancements have significantly improved Realtime accuracy.

RESULTS

The static and dynamic gesture recognition models demonstrated strong performance across various metrics, validating the data processing pipeline, dataset quality, and model architecture.

I. Static Signs

For Static Signs, the alphabet and number gestures trained on the model had high accuracy and performance consistency across the key metrics. It reached an accuracy of 0.984, with a precision of 0.987, recall of 0.986, and an F1-score of 0.986. These metrics prove how the model may classify a wide variety of static gestures in different hand poses and lighting conditions. A higher value in precision would indicate a low rate of false positives wherein the model correctly identifies the gesture but mistakenly classifies other inputs as gestures. The same high recall score validated the model's reliability towards correctly detecting the gestures with fewer instances of missed gestures. The balance in precision and recall as reflected through the F1-score necessarily implies that the model is robust and suitable for a real-world static gesture recognition system.

II. Dynamic Signs

The LSTM model showed perfect accuracy during training, reflected in the classification report as performance over the dataset. This is based on predictions of training data rather than seeing the data, thus reflecting that the model has learned from the data effectively.

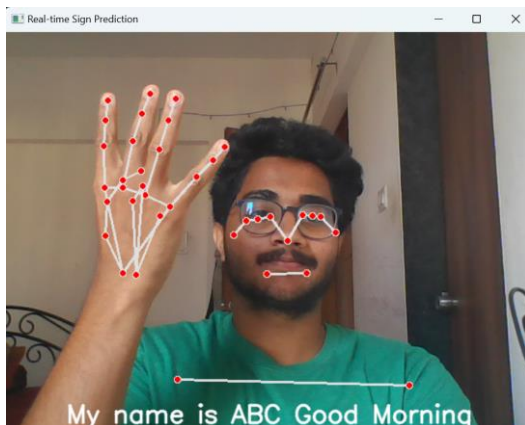


Figure (6) Real-Time sentence formation

During real-time testing, it performed very well with the efficiency of gesture prediction without a delay [Figure (6)]. The cooldown mechanism and hand-detection check were incorporated into the design of the prediction logic such that unwanted or duplicate predictions were avoided. Through this external logic, it offered smooth, accurate outputs and was not interrupted by quick hand movements or changes in positions, thus enhancing user experience.

These results were achieved with the help of data collection using MediaPipe [Figure (7)], which was able to pick up precise hand keypoints. This allowed for a comprehensive and reliable dataset, which helped the model generalize well to real-time scenarios and ensure that it could handle dynamic hand gestures with accuracy and speed.

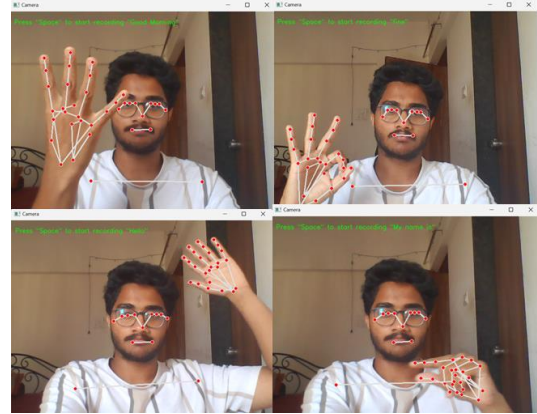


Figure (7) Data-Collection process

Overall, the results validate the system's ability to recognize both static and dynamic gestures effectively. The static gesture model's high performance metrics confirm the reliability of the hand landmark data and neural network architecture, while the dynamic model's real-time adaptability highlights the effectiveness of the LSTM design and prediction logic. Together, these components establish a solid foundation for gesture-based communication, achieving robust and accurate recognition across a wide range of gestures.

FUTURE SCOPE

This project on dynamic gesture recognition holds great promise for applications across domains. One of the promising areas for future work is in improving accessibility technology, especially real-time translation for sign language. Good groundwork has been laid down by the current models; however, there is room to expand capabilities in recognizing more complex gestures and even finger-spelling for full vocabulary coverage in sign languages. Future implementations could be considering the use of this model as a basis for recognizing multilingual sign languages because it could be adapted to apply to the different nuances from one region to another.

More advanced sensor data would be incorporated, like depth information from sensors such as the Microsoft Kinect or Apple's LiDAR, which could offer even richer spatial data than 2D images. This would enhance the accuracy of gestures with complex depth cues and increase robustness against varying lighting and background conditions. Depth-based data can let the model operate with richer three-dimensional gestures, thus becoming more suitable for the

applications of virtual reality and augmented reality, which are often mostly gesture-driven.

Another area of potential future work is the optimization of the architecture of the model for deployment in mobile and embedded systems that have very limited resources of computation. With such a structure, in the form of an efficient lightweight MobileNet combined with LSTM, or further techniques such as model quantization and pruning, one might even deploy this system in mobile phones or wearables. All of these advances would support applications in personal devices, smart home systems, and wearable technology, enabling gesture-based commands in more user-centric environments.

Real-time capabilities can be improved through future iterations with advanced techniques of temporal smoothing in order to stabilize and further decrease noise in predictions. It would be possible by integrating techniques like Kalman filtering and exponential smoothing, which are very commonly used in time series data. Filtering out the noise in terms of unnecessary fluctuations, such methods make predictions stable and accurate. This is necessary for healthcare applications, robotic control, and even some interactive public installations where accuracy and consistency are paramount.

This approach can be improved for adaptability and generalization to new conditions by further datasets covering different hand shapes, orientations, and lighting conditions. Fine-tuning on user-specific data for achieving more personalization or looking into unsupervised learning for adaptation of new types of gestures without manual extensive labeling can be looked at as future studies. Such approaches will benefit not only the accuracy of results but also increase its usability by people with different morphologies of hands and types of gestures.

Lastly, this technology can develop applications on collaborative human-computer interaction, particularly in remote communication and work. This could be made possible by using the integration of gesture recognition into voice commands that create a much more intuitive multimodal interface that integrates verbal and non-verbal communication seamlessly. This kind of multimodal approach can be very invaluable in telemedicine, education, and virtual team collaborations by allowing a more enhanced interactive experience through natural gestures as commands and expressions within digital environments.

The following table [Table (1)] exhibits some interesting applications of LSTM networks across different domains. It can be noticed here that the model generalizes quite well in the complex real-world scenarios shown here. LSTM does consistently well across various domains, from energy forecasting to structural seismic prediction.

References	Application	Journal
Ghimire et al.	Solar radiation forecasting	Applied Energy
Liu	Volatility forecasting	Expert Systems with Applications
Hong et al.	Fault prognosis of battery systems	Applied Energy
Krishan	Air quality prediction	Air Quality and Atmosphere
Zhang et al.	Structural seismic prediction	Computers and Structures
Hua et al.	Time Series Prediction	IEEE Communications
Zhang et al.	Wind turbine power prediction	Applied Energy
Vardan et al.	Earthquake trend prediction	Electrical and Computer Engineering

Table (1) LSTM in Real Life fields [27]

Because of the intricacies involved with gesture recognition, dynamic gesture detection using MediaPipe keypoints would be a wonderful application of LSTM. This makes the model well-suited for further investigation in this domain, since the temporal dependencies it captures are exactly what gesture data is sequential in.

ACKNOWLEDGMENT

We thank our guide, Dr. Dipti Theng, for the entire research work; special thanks are also to our co-guide, Prof. Madhuri Hiwale, who is helpful and informative during the work of this project. We thank our computer science department at the Symbiosis Institute of Technology for their continued support.

REFERENCES

- [1] Eid, A., & Schwenker, F. (2023). Visual static hand gesture recognition using convolutional neural network. *Algorithms*, 16(8), 361.
- [2] Prakash, A. J., Plawiak, P., & Samantray, S. (2022). Real-time hand gesture recognition using fine-tuned convolutional neural network. *Sensors*, 22(3), 706.
- [3] Plawiak, P., & Samantray, S. (2021). Real-time dynamic gesture recognition in human-computer interaction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.
- [4] Rehman, M. U., et al. (2022). Dynamic hand gesture recognition using 3D-CNN and LSTM networks. *Computers, Materials & Continua*, 70(3), 4675-4690.
- [5] Zheng, J., et al. (2020). Hybrid CNN-LSTM model for dynamic gesture recognition. *IEEE Transactions on Multimedia*.
- [6] MediaPipe. (2022). MediaPipe holistic tracking system for real-time applications.
- [7] Pathak, R., et al. (2023). Gesture recognition in accessibility: Enhancing communication for the hearing-impaired. *Journal of Human-Computer Interaction*.
- [8] Zhang, Y., & Li, M. (2021). Dynamic gesture recognition using stacked LSTM networks. *IEEE Xplore*.
- [9] Kim, J., Jamil, S., Lee, J., & Ullah, F. (2024). Next-Gen dynamic hand gesture recognition: MediaPipe, Inception-v3, and LSTM-based enhanced deep learning model. *Electronics*, 13(16), 3233.
- [10] Kumar, R., Bajpai, A., & Sinha, A. (2023). Mediapipe and CNNs for real-time ASL gesture recognition. *arXiv preprint*.
- [11] Lee, H., & Kim, J. (2021). LM-Net: A dynamic gesture recognition network with long-term temporal context. *SpringerLink*.
- [12] Chen, L., et al. (2022). Real-time hand gesture recognition using CNN and LSTM for drone control applications. *MDPI Sensors*, 22(9), 4598.
- [13] Kumar, N., Kumar, S., & Kumar, M. (2023). Dynamic hand gesture recognition using CNN. *SSRN Electronic Journal*.
- [14] Jaeho, K., et al. (2022). Facial expression recognition using hybrid CNN and ConvLSTM networks. *SpringerLink*.
- [15] Kumar, A., Bajaj, P., & Verma, R. (2023). Enhancing sign language recognition: A CNN-BiLSTM approach. *IEEE Xplore*.
- [16] Javed, F., et al. (2023). Assessing the influence of LSTM and post-processing in CNN-based gesture recognition. *SSRN Electronic Journal*.
- [17] Rehman, S. (2023). Enhancing the accuracy of gesture recognition using CNN. *IEEE Transactions on Human-Machine Systems*.
- [18] Liu, Z., & Wang, H. (2023). Gesture recognition in human-computer interaction: Challenges and solutions. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.
- [19] Wang, F., et al. (2023). Comparative analysis of CNN and LSTM for continuous hand gesture recognition. *IEEE Transactions on Human-Machine Systems*.
- [20] Park, J., & Kim, Y. (2024). High-efficiency real-time gesture recognition using CNN-LSTM networks. *Springer Nature*.
- [21] Chen, X., & Zhang, S. (2023). A CNN-BiLSTM based multimodal continuous hand gesture recognition system. *PLOS ONE*.

- [22] Ahmad, S., et al. (2022). Data glove-based gesture recognition using CNN-BiLSTM model. PLOS ONE.
- [23] Lin, D., & Chen, Z. (2023). Real-time dynamic gesture recognition: Combining CNN and LSTM for improved accuracy. SpringerLink.
- [24] Singh, P., & Yadav, A. (2023). Using LSTM networks for real-time gesture prediction in AR applications. IEEE Xplore.
- [25] Ahn, Y., & Han, J. (2022). Real-time multimodal gesture recognition using CNN and LSTM integration. MDPI Sensors, 22(6), 1105.
- [26] https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker
- [27] https://www.researchgate.net/figure/The-notable-applications-of-LSTM_tb13_338565420