Imaginex International Journal of Interdisciplinary Research and Reviews (IIJIRR)
Volume 1, Issue 3, November – December 2026
ISSN: 3140-0094 (Online)
www.iijirr.com

Gesture-Controlled Robotic Arm for Assistive Human–Machine Interaction

Sai Muthu Lakshmi L M[1], Rishanya Shamganth[1], Senthil Kumaran R[1,*]

[1]Department of Electronics and Communication Engineering, SRM Institute of Science and Technology, Chennai, Tamil Nadu, India
E-mail addresses (in author order): sl1812@srmist.edu.in, rs9135@srmist.edu.in, senthilr7@srmist.edu.in
[*]Corresponding author: senthilr7@srmist.edu.in

Abstract

This paper presents a gesture-controlled robotic arm that lets a user move a five-degree-of-freedom arm with hand gestures instead of a joystick or buttons. An MPU6050 sensor worn on the hand records six-axis motion data, and a CNN-LSTM model classifies it into six predefined gestures: left, right, forward, backward, grip open, and grip close. An STM32F103C8T6 microcontroller reads the sensor and drives the servo motors according to the predicted gesture. Because the system relies only on an inertial sensor, it needs no camera, and its parts keep the cost low, which may suit assistive robotic manipulation. Experimental evaluation of the CNN-LSTM classifier achieved a window-level accuracy of 90.00% (216/240 windows) and a recording-level accuracy of 93.75% (45/48 recordings). The recording-level results demonstrate strong classification performance across the six predefined gesture classes, although further testing is required across multiple users and operating conditions. The average response time of about 150 ms indicates that near real-time control is feasible.

Keywords: Assistive robotics, CNN-LSTM, Deep learning, Embedded systems, Gesture recognition, Human-robot interaction, Inertial measurement unit (IMU), MPU6050, Real-time control, Robotic arm, Servo control, STM32F103C8T6, Time-series classification, Wearable sensors

1. Introduction

Robotic arms have been controlled through gesture-based human–machine interfaces using inertial, electromyographic, and vision-based approaches [1], [2]. Gesture-based control uses hand movement as the input. Motions are captured by inertial measurement units (IMUs) without relying on visual markers or camera-based tracking. An IMU combines an accelerometer and a gyroscope, whose measurements can be used to distinguish gestures [3], [4].

This work applies these approaches to robotic-arm control. An MPU6050 sensor captures the user's hand motion [5], and a CNN-LSTM model classifies gestures from a labeled dataset rather than relying on fixed thresholds for individual sensor axes. During prototyping, the model runs on a host computer; an STM32F103C8T6 microcontroller receives the recognized commands and controls the arm's servos [6].

The objective is to develop a low-cost gesture-controlled robotic arm that accepts movement commands through predefined hand gestures. The system uses a five-degree-of-freedom arm and addresses the complete process from motion sensing and gesture classification to embedded motor control.

2. Related work

Gesture recognition has been studied using vision-based systems and wearable or inertial sensors. Vision-based robotic-arm control has been demonstrated using OpenCV [2], while wearable-sensor research includes IMU-based activity recognition and gesture sensing [3], [7], [8]. Such sensors do not rely on a camera view of the hand, because IMUs measure motion directly. Sequential hand-gesture recognition using cooperative sensing and IMU-based interfaces has also been investigated [9], [10].

An IMU-based gesture-recognition system typically processes accelerometer and gyroscope measurements as time-series data. The MPU6050 integrates a three-axis accelerometer and a three-axis gyroscope, providing six-axis motion measurements [5].

Deep-learning methods have been applied to sensor-based activity and gesture recognition [11]–[13]. Lauss et al. investigated LSTM-based hand-gesture recognition using IMU sensors on a low-power microcontroller [14]. More recently, LSTM networks for accelerometer-based dynamic hand-gesture recognition have been run directly on an STM32 microcontroller [15]. CNN-LSTM architectures have also been studied for activity recognition with wearable sensors [16], [17]. In these models, convolutional layers are used to extract patterns from sensor signals,

while LSTM layers model temporal relationships. This combination is relevant to gesture classification because gestures are represented by sequences of measurements rather than isolated sensor values.

Related robotic-control studies use a range of sensing and learning approaches. Shin et al. combined EMG and IMU signals for dynamic-gesture control of a multi-degree-of-freedom robotic arm [1], while Kurpath et al. used IMUs and a potentiometer for robotic arm-hand teleoperation [18]. Cruz et al. investigated DQN-based hand-gesture recognition for controlling robotic platforms [19]. A recent glove that combines five flex sensors with an MPU6050 used a recurrent neural network to classify seven gestures for real-time control of a robotic arm [20]. These approaches provide context for the proposed IMU-based system and its CNN-LSTM classifier.

The proposed prototype runs gesture classification on a host computer and uses an STM32F103 microcontroller for embedded control [6]. Gesture classification and motor control are handled by separate devices: the trained model runs on the host computer, and the STM32 receives a gesture label and converts it to servo commands. Running the classifier on the microcontroller itself, as in [14] and [15], is left for future work (Section 7).

3. System model and architecture

The system has four parts: a sensor that picks up hand motion, a classifier, a microcontroller that turns commands into servo signals, and the arm itself. Figure 1 shows how data moves between them.

The MPU6050 is mounted on the user's hand or wrist and measures three-axis acceleration and three-axis angular velocity, producing six motion signals [5]. While the user makes a gesture, the readings are logged as a short sequence and sent to the host computer. The host computer pre-processes the sequence and passes it to the CNN-LSTM, which assigns it to one of the six gesture classes.

The host computer converts the predicted class into a command and sends it to the STM32F103C8T6 over a serial connection [6]. The STM32 reads the command and sets the PWM output for the matching servo. The arm has five servos, one each for the base, shoulder, elbow, wrist and gripper; which servo a gesture moves depends on how the arm is built.

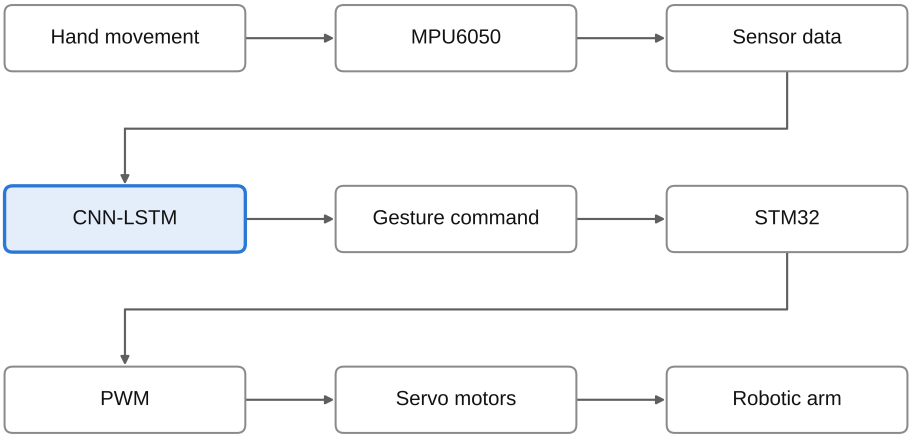

Figure 1. Proposed IMU-based gesture-controlled robotic-arm system architecture.

3.1. Motion sensing and data acquisition

The MPU6050 is used to measure hand motion. It integrates a 3-axis accelerometer and a 3-axis gyroscope, providing six motion channels [5]. At time t, the acceleration and angular-velocity vectors are:

$$a_t = [a_{x,t},\ a_{y,t},\ a_{z,t}] \tag{1}$$

$$\omega_t = [\omega_{x,t},\ \omega_{y,t},\ \omega_{z,t}] \tag{2}$$

The six-channel sensor sample is

$$x_t = [a_{x,t},\ a_{y,t},\ a_{z,t},\ \omega_{x,t},\ \omega_{y,t},\ \omega_{z,t}] \tag{3}$$

The MPU6050 is connected to the STM32 over I²C. The STM32 reads each sample and forwards it to the host computer, which collects the dataset and later runs the classification.

3.2. Gesture dataset

The dataset contains six gestures: BACKWARD, FORWARD, GRIP_CLOSE, GRIP_OPEN, LEFT, and RIGHT. Data were collected using an MPU6050 sensor, which provides three-axis accelerometer and gyroscope readings. Each recording contains 300 samples across six channels: Ax, Ay, Az, Gx, Gy, and Gz. The dataset consists of 300

recordings, with 50 recordings per gesture. The recordings were divided into training, validation, and test sets containing 210, 42, and 48 recordings, respectively.

3.3. Data pre-processing

Each recording was checked for missing or invalid values and the expected data dimensions. The signals were divided into windows of 100 samples with a stride of 50 samples, producing five windows per recording. Each window was represented by a 100 × 6 matrix. Feature standardisation was performed using a StandardScaler fitted on the training data and then applied to the validation and test sets.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_T \end{bmatrix} \in \mathbb{R}^{T \times 6} \tag{4}$$

where $T = 100$ is the number of samples in a window, and each row contains the six IMU channels.

3.4. CNN-LSTM gesture classification

The CNN-LSTM model combines convolutional layers for feature extraction with an LSTM layer for temporal analysis (Figure 2). The network consists of two Conv1D layers with 64 and 96 filters, followed by batch normalisation, max pooling, and an LSTM layer with 64 units. Dropout and L2 regularisation are used to reduce overfitting. A fully connected layer with 64 units and a six-unit softmax output layer produce the gesture predictions. The model was trained using the Adam optimiser with a learning rate of 0.001, sparse categorical cross-entropy loss, a batch size of 32, and a maximum of 80 epochs. Early stopping and model checkpointing were used to retain the model with the lowest validation loss.

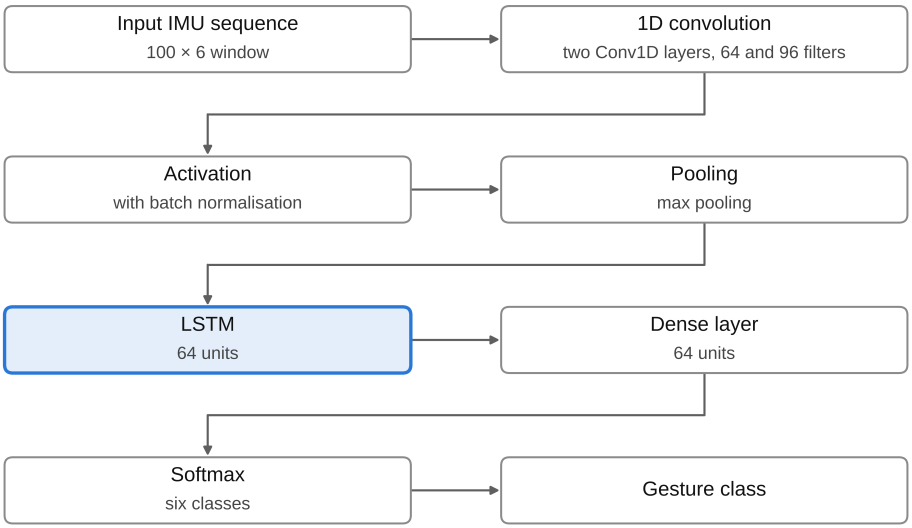

Figure 2. CNN-LSTM architecture for IMU-based gesture classification. Layer sizes are those given in Section 3.4.

3.5. Command generation

The classifier outputs a gesture label, which a command-mapping stage converts into a control command for the STM32. The mapping is as follows: LEFT and RIGHT command base movement in the corresponding directions; FORWARD and BACKWARD command movement toward their predefined positions; and GRIP_OPEN and GRIP_CLOSE command the gripper to open and close, respectively.

3.6. STM32 control

The STM32F103C8T6 Blue Pill serves as the microcontroller. The STM32F103 device family uses an Arm Cortex-M3 core and provides timers with PWM capability, as well as I²C and USART peripherals [6]. In this system, the STM32 interfaces with the MPU6050, receives the classified gesture commands from the host computer and generates PWM signals for the robotic-arm servos.

The CNN-LSTM model runs on the host computer rather than on the STM32 in the proposed prototype. This division assigns classification to the host computer and actuator control to the microcontroller.

3.7. Servo control

The robotic arm has five servo motors, corresponding to its five controlled degrees of freedom. For each received command, the STM32 maps the command to a predefined servo position or movement. Firmware joint limits constrain the commanded positions to the intended mechanical ranges.

The servos are powered from a separate 5 V, 5 A supply whose ground is connected to the STM32 ground, giving a common reference for the PWM signals.

4. Hardware and software implementation and integration

The main hardware components are listed in Table 1. The STM32F103C8T6 Blue Pill serves as the microcontroller, while the host computer processes the dataset and runs the CNN-LSTM model. The MPU6050 connects with the STM32 through the I²C interface [5]. The host computer sends the classified gesture commands to the STM32 over a serial connection. The STM32 generates the PWM signals used to control the robotic-arm servos [6].

Table 1. Main hardware components.

Component	Purpose
STM32F103C8T6 Blue Pill	Microcontroller and PWM generation
MPU6050	Six-axis hand-motion sensing
Robo Wings 5-DOF robotic arm	Mechanical manipulation
Servo motors	Joint and gripper actuation
5 V, 5 A power supply	External servo power
ST-Link V2	STM32 programming and debugging
Laptop	Dataset processing and CNN-LSTM inference
Jumper wires and connectors	Electrical connections

The software has two components: a host-computer application for data processing and gesture classification, and firmware on the STM32 for sensor acquisition, command handling and servo control.

4.1. Software

Python is used for IMU data collection, preprocessing, model training and evaluation, and gesture prediction. The processing pipeline has the following steps:

(1) Acquire and store IMU samples.

(2) Assign gesture labels to recorded sequences.

(3) Preprocess the labelled data and divide it into training, validation and test subsets.

(4) Train and evaluate the CNN-LSTM model.

(5) Load the trained model for predicting the gestures.

(6) Map each predicted gesture to a control command and transmit it to the STM32.

The host-computer software uses Python libraries for sensor-data processing, feature normalization, CNN-LSTM model training, evaluation, visualization, and real-time gesture prediction. The implemented pipeline includes data acquisition, preprocessing, model inference, gesture-to-command mapping, and serial communication with the STM32.

4.2. STM32 firmware

The STM32F103C8T6 reads accelerometer and gyroscope data from the MPU6050 through I²C and sends the readings to the host computer through a USB-to-TTL serial connection. The host computer classifies the gesture using the trained CNN-LSTM model and returns the predicted command to the STM32, which controls the robotic-arm servos. The servos are powered by an external 5 V, 5 A supply, with a common ground shared with the microcontroller. Figure 3 shows the firmware flowchart, and Figure 4 shows the circuit connections and the communication flow.

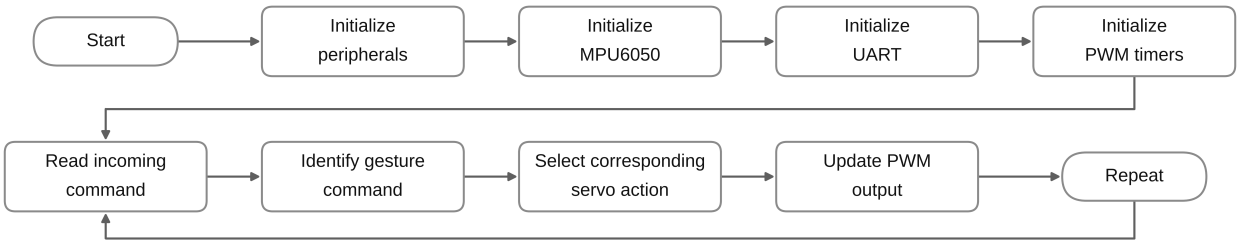

Figure 3. STM32 firmware flowchart.

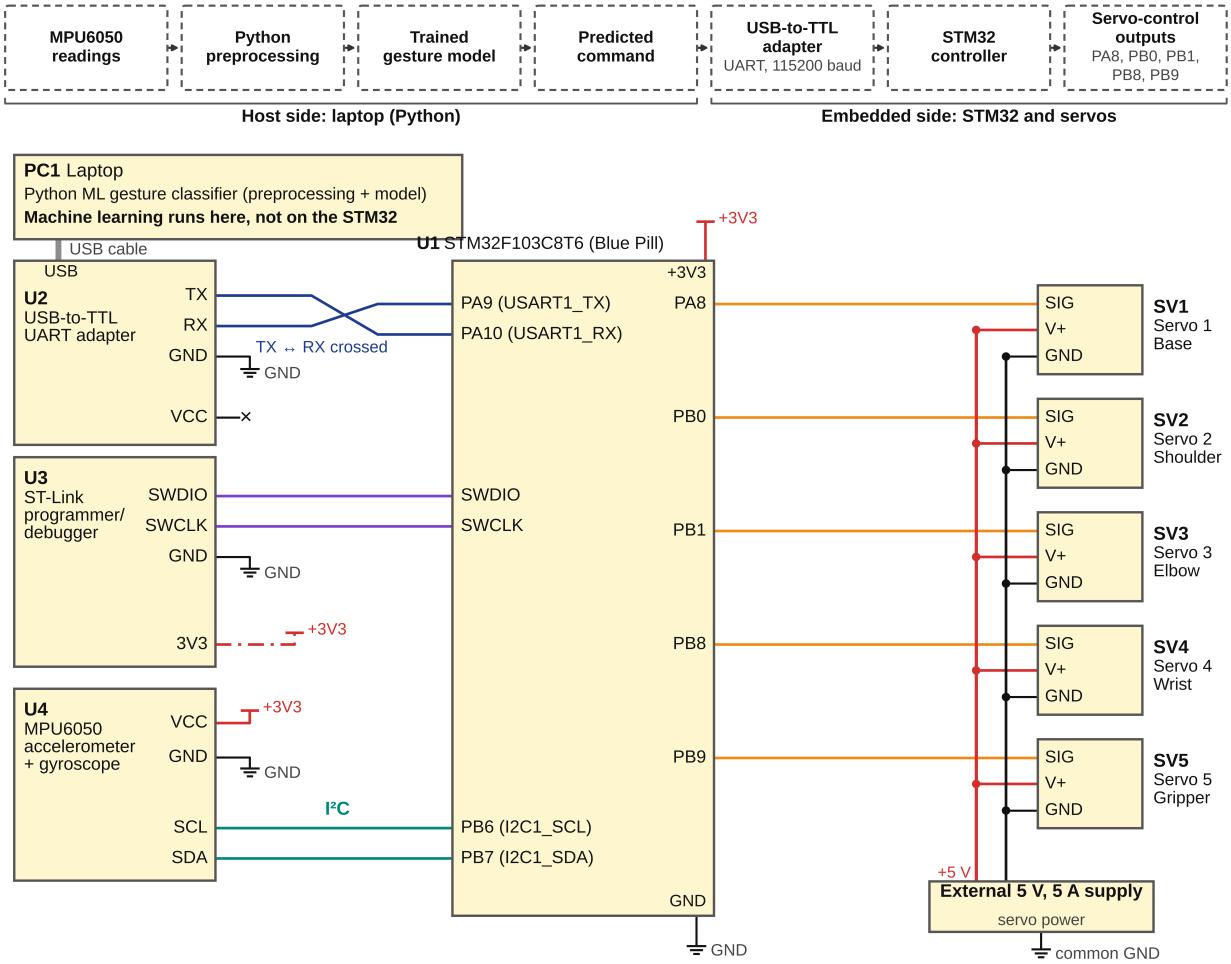

Figure 4. Circuit connections and communication flow between the MPU6050, host computer, STM32 microcontroller and robotic-arm servos. The servo supply and common ground follow Sections 3.7 and 4.2.

5. Experimental methodology

The system was evaluated in three stages: gesture classification, real-time response and robotic-arm operation.

5.1. Gesture classification evaluation

The CNN-LSTM model was evaluated using recordings reserved for testing. These recordings were kept separate from the training data to assess how well the model could recognize gestures that it had not encountered during training. Accuracy, precision, recall, and F1-score were used to evaluate classification performance. The evaluation was carried out at both the window level and the recording level, providing two ways to examine the model's ability to identify the six predefined gestures.

5.2. Real-time response evaluation

The system's response time was measured from the completion of a hand gesture to the corresponding movement of the robotic arm. The measured response time was approximately 150 ms, including gesture processing and command communication. This indicates that the system can respond to hand gestures with low latency, supporting its use in real-time robotic-arm control.

5.3. Robotic-arm evaluation

The robotic arm was evaluated using the six supported gesture commands: BACKWARD, FORWARD, GRIP_CLOSE, GRIP_OPEN, LEFT, and RIGHT. The host computer processed the MPU6050 data using the trained CNN-LSTM model and transmitted the predicted command to the STM32, which controlled the corresponding arm movement. The response time for these commands is given in Section 5.2. The evaluation demonstrates the system's ability to connect gesture classification with robotic arm control.

6. Results and discussion

Table 2 presents the precision, recall, and F1-score for each gesture class on the test set, along with an overall recording-level accuracy of 93.75% (45 out of 48 recordings).

Table 2. Recording-level gesture classification results on the test set.

Gesture	Precision	Recall	F1-score
LEFT	0.86	0.75	0.80
RIGHT	1.00	1.00	1.00
FORWARD	1.00	1.00	1.00
BACKWARD	0.78	0.88	0.82
GRIP_OPEN	1.00	1.00	1.00
GRIP_CLOSE	1.00	1.00	1.00
Macro average	0.94	0.94	0.94

The CNN-LSTM model was evaluated on six gesture classes using recordings reserved for testing. At the window level, the model achieved an accuracy of 90.00%, correctly classifying 216 out of 240 test windows. At the recording level, it achieved an accuracy of 93.75%, correctly classifying 45 out of 48 recordings. The class-wise precision, recall, and F1-scores at the recording level are presented in Table 2. The results show that the model recognized FORWARD, GRIP_CLOSE, GRIP_OPEN and RIGHT correctly across the evaluated recordings.

Figure 5 shows the confusion matrix at the window level (216 of 240 test windows correct). Most window-level errors (21 of 24) involve LEFT or BACKWARD; at the recording level, all errors are between LEFT and BACKWARD. Figures 6 and 7 show training and validation accuracy and loss over 35 epochs. Both curves converge, and validation loss levels off slightly above training loss in later epochs, which suggests mild overfitting given the small dataset. Table 3 summarises the system performance.

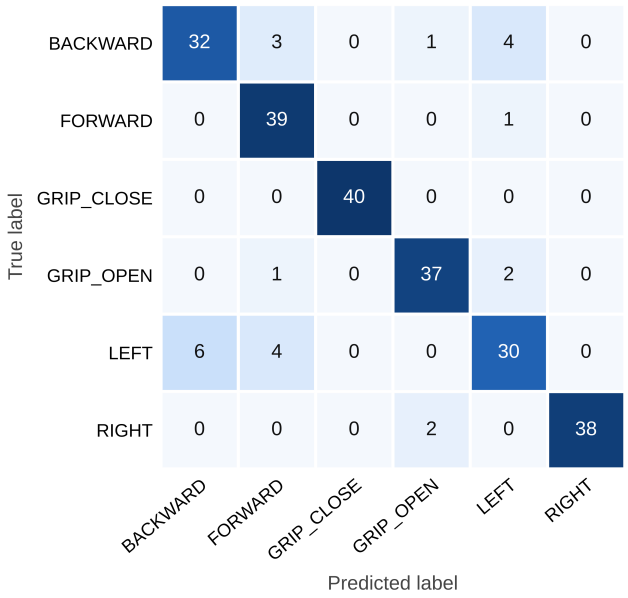

Figure 5. Window-level confusion matrix for the test set (240 windows from 48 recordings); cell values are window counts.

Table 3. System performance.

Parameter	Measured value
Window-level test accuracy	90.00% (216/240 windows)
Recording-level test accuracy	93.75% (45/48 recordings)
Average response time	150 ms
Number of gesture classes	6
Number of IMU channels	6
Number of servo motors	5

The results should be considered alongside the robotic arm's physical response. Classification accuracy alone does not establish reliable operation because an incorrect or delayed command may result in an unintended movement. Accordingly, evaluation should consider both gesture-classification performance and end-to-end system response.

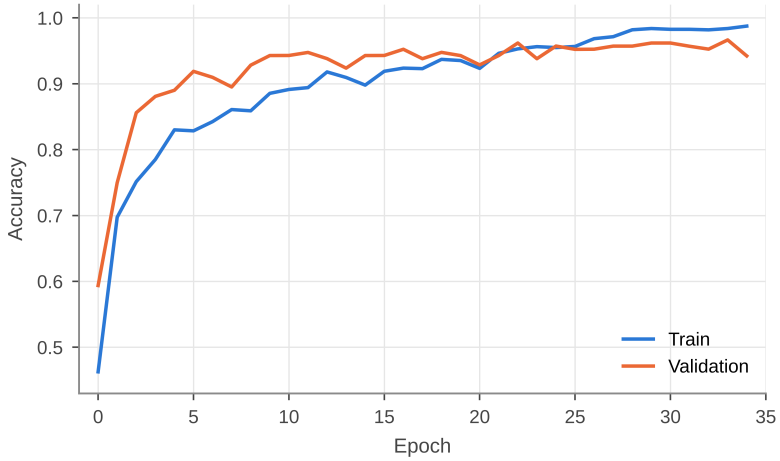

Figure 6. Training and validation accuracy of the CNN-LSTM model over 35 epochs.

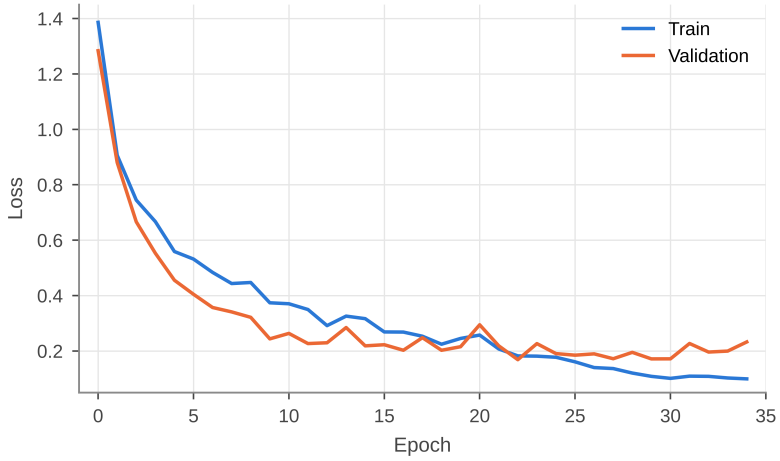

Figure 7. Training and validation loss of the CNN-LSTM model over 35 epochs.

7. Conclusion and future scope

This paper presented a gesture-controlled robotic arm using an MPU6050 inertial sensor, a CNN-LSTM gesture-classification model, an STM32F103C8T6 microcontroller, and a five-degree-of-freedom robotic arm. The system classifies predefined hand gestures from six-axis inertial data and maps the predicted classes to commands for the arm. The CNN extracts local patterns from the sensor recordings, while the LSTM models their temporal characteristics.

Gesture classification runs on the host computer, while the STM32 handles embedded control and generates PWM signals for the servo motors. This division allows the prototype to use a microcontroller for actuator control without requiring it to execute the neural network.

The CNN-LSTM classifier achieved a window-level test accuracy of 90.00% (216/240 windows) and a recording-level test accuracy of 93.75% (45/48 recordings). These results demonstrate the feasibility of the proposed host-computer-based gesture-classification approach for controlling a five-degree-of-freedom robotic arm. However, evaluation across additional users, gesture-execution conditions, and physical arm trials is required to establish the robustness and reliability of the complete system.

Future work will expand the dataset to include multiple users and assess performance under varied gesture-execution conditions. Further work may examine additional pre-processing, sensor fusion, and model optimization for embedded inference. Additional commands and user feedback mechanisms could also be investigated.

Declarations

Conflict of interest. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding. This research received no external funding.

Data availability. The gesture dataset and code are available from the corresponding author on reasonable request.

Use of generative AI. During the preparation of this work the authors used an AI-assisted tool for language editing. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Ethics approval and consent. The gestures were performed by the authors, who consented to the use of their data. Formal ethics approval was not required for this non-invasive study.

Author contributions. Sai Muthu Lakshmi L M, Rishanya Shamganth and Senthil Kumaran R contributed to this work. All authors read and approved the final manuscript.

References

[1] S. Shin, R. Tafreshi, and R. Langari, “EMG and IMU based real-time HCI using dynamic hand gestures for a multiple-DoF robot arm,” *J. Intell. Fuzzy Syst.*, vol. 35, no. 1, pp. 861–876, 2018, doi: 10.3233/JIFS-171562.

[2] J. Paterson and A. Aldabbagh, “Gesture-controlled robotic arm utilizing OpenCV,” in *Proc. 3rd Int. Congr. Human-Comput. Interact., Optim. Robot. Appl. (HORA)*, Ankara, Turkey, 2021, pp. 1–6, doi: 10.1109/HORA52670.2021.9461389.

[3] M. Cornacchia, K. Özcan, Y. Zheng, and S. Velipasalar, “A survey on activity detection and classification using wearable sensors,” *IEEE Sens. J.*, vol. 17, no. 2, pp. 386–403, Jan. 2017, doi: 10.1109/JSEN.2016.2628346.

[4] K. Altun and B. Barshan, “Human activity recognition using inertial/magnetic sensor units,” in *Human Behavior Understanding (HBU 2010)*, Lecture Notes in Computer Science, vol. 6219. Berlin, Germany: Springer, 2010, pp. 38–51, doi: 10.1007/978-3-642-14715-9_5.

[5] InvenSense Inc., “MPU-6000 and MPU-6050 product specification,” rev. 3.4, Doc. PS-MPU-6000A-00, Aug. 2013.

[6] STMicroelectronics, “STM32F103x8, STM32F103xB: Medium-density performance line Arm-based 32-bit MCU with 64 or 128 KB flash,” datasheet DS5319, rev. 20, Jul. 2025.

[7] Ó. D. Lara and M. A. Labrador, “A survey on human activity recognition using wearable sensors,” *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1192–1209, 3rd Quart. 2013, doi: 10.1109/SURV.2012.110112.00192.

[8] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, “Activity recognition using cell phone accelerometers,” *ACM SIGKDD Explor. Newslett.*, vol. 12, no. 2, pp. 74–82, Dec. 2010, doi: 10.1145/1964897.1964918.

[9] X. Zhang, Z. Yang, T. Chen, D. Chen, and M.-C. Huang, “Cooperative sensing and wearable computing for sequential hand gesture recognition,” *IEEE Sens. J.*, vol. 19, no. 14, pp. 5775–5783, Jul. 2019, doi: 10.1109/JSEN.2019.2904595.

[10] M. Kim, J. Cho, S. Lee, and Y. Jung, “IMU sensor-based hand gesture recognition for human-machine interfaces,” *Sensors*, vol. 19, no. 18, Art. no. 3827, Sep. 2019, doi: 10.3390/s19183827.

[11] M. F. Trujillo-Guerrero et al., “Accuracy comparison of CNN, LSTM, and Transformer for activity recognition using IMU and visual markers,” *IEEE Access*, vol. 11, pp. 106650–106669, 2023, doi: 10.1109/ACCESS.2023.3318563.

[12] Y. Chen and Y. Xue, “A deep learning approach to human activity recognition based on single accelerometer,” in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, 2015, pp. 1488–1492, doi: 10.1109/SMC.2015.263.

[13] M. Zeng et al., “Convolutional neural networks for human activity recognition using mobile sensors,” in *Proc. 6th Int. Conf. Mobile Comput., Appl. Serv. (MobiCASE)*, 2014, pp. 197–205, doi: 10.4108/icst.mobicase.2014.257786.

[14] D. Lauss, F. Eibensteiner, and P. Petz, “A deep learning based hand gesture recognition on a low-power microcontroller using IMU sensors,” in *Proc. 21st IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2022, pp. 733–736, doi: 10.1109/ICMLA55696.2022.00122.

[15] K. Di Leo, G. Biagetti, L. Falaschetti, and P. Crippa, "Microcontroller implementation of LSTM neural networks for dynamic hand gesture recognition," *Sensors*, vol. 25, no. 12, Art. no. 3831, 2025, doi: 10.3390/s25123831.

[16] M. A. Khatun et al., "Deep CNN-LSTM with self-attention model for human activity recognition using wearable sensor," *IEEE J. Transl. Eng. Health Med.*, vol. 10, Art. no. 2700316, 2022, doi: 10.1109/JTEHM.2022.3177710.

[17] F. J. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, Jan. 2016, doi: 10.3390/s16010115.

[18] M. I. Kurpath, P. K. Adwai, J. Bodireddy, K. Chandrasekaran, and K. Navin Sam, "An IMUs and potentiometer-based controller for robotic arm-hand teleoperation," *Sens. Actuators A, Phys.*, vol. 367, Art. no. 115019, Mar. 2024, doi: 10.1016/j.sna.2024.115019.

[19] P. J. Cruz, J. P. Vásconez, R. Romero, et al., "A deep Q-network based hand gesture recognition system for control of robotic platforms," *Sci. Rep.*, vol. 13, Art. no. 7956, May 2023, doi: 10.1038/s41598-023-34540-x.

[20] N. Herbaz, H. El Idrissi, H. Sabir, and A. Badri, "Smart wearable glove for enhanced human-robot interaction using multi-sensor fusion and machine learning," *Int. J. Electr. Comput. Eng.*, vol. 15, no. 6, pp. 5162–5172, Dec. 2025, doi: 10.11591/ijece.v15i6.pp5162-5172.