engineering proceedings

MDPI

Proceeding Paper

A Smart Glove-Based System for Dynamic Sign Language Translation Using LSTM Networks †

Tabassum Kanwal [1,*], Saud Altaf [2], Rehan Mehmood Yousaf [1] and Kashif Sattar [1]

[1] University Institute of Information Technology, PMAS-University of Arid Agriculture, Rawalpindi 46000, Pakistan; rehan.yousaf@uaar.edu.pk (R.M.Y.); kashif@uaar.edu.pk (K.S.)
[2] Department of Information Engineering Technology, National Skills University, Islamabad 44000, Pakistan; saud@nsu.edu.pk
* Correspondence: tabassum.kanwal@f.rwu.edu.pk
† Presented at the 12th International Electronic Conference on Sensors and Applications (ECSA-12), 12–14 November 2025; Available online: https://sciforum.net/event/ECSA-12.

Abstract

This research presents a novel, real-time Pakistani Sign Language (PSL) recognition system utilizing a custom-designed sensory glove integrated with advanced machine learning techniques. The system aims to bridge communication gaps for individuals with hearing and speech impairments by translating hand gestures into readable text. At the core of this work is a smart glove engineered with five resistive flex sensors for precise finger flexion detection and a 9-DOF Inertial Measurement Unit (IMU) for capturing hand orientation and movement. The glove is powered by a compact microcontroller, which processes the analog and digital sensor inputs and transmits the data wirelessly to a host computer. A rechargeable 3.7 V Li-Po battery ensures portability, while a dynamic dataset comprising both static alphabet gestures and dynamic PSL phrases was recorded using this setup. The collected data was used to train two models: a Support Vector Machine with feature extraction (SVM-FE) and a Long Short-Term Memory (LSTM) deep learning network. The LSTM model outperformed traditional methods, achieving an accuracy of 98.6% in real-time gesture recognition. The proposed system demonstrates robust performance and offers practical applications in smart home interfaces, virtual and augmented reality, gaming, and assistive technologies. By combining ergonomic hardware with intelligent algorithms, this research takes a significant step toward inclusive communication and more natural human-machine interaction.

Keywords: sensory glove; hand gesture recognition; machine learning; deep learning; long short-term memory (LSTM)

Academic Editor(s): Name

Published: date

Citation: Kanwal, T.; Altaf, S.; Yousaf, R.M.; Sattar, K. A Smart Glove-Based System for Dynamic Sign Language Translation Using LSTM Networks. *Eng. Proc.* **2025**, *x*, x. https://doi.org/10.3390/xxxxx

1. Introduction

Many deaf and speech-impaired Pakistanis struggle to communicate and socialize. Technologies that make communication easier and more inclusive are needed to reduce social isolation and access to services and opportunities [1].

More than 120 sign languages exist, including American, Indian, Italian, and Pakistan Sign Language (PSL) [2]. Most Sign Language Recognition (SLR) systems interpret hand motions based on their attributes. In PSL, the SLR system may produce text on a screen or speech on a speaker device, improving communication for Pakistan's deaf and hard-of-hearing [3–6].

Recently, deep learning architectures have achieved state-of-the-art performance on several computer vision problems [7,8]. SLR system design using a Convolutional Neural Network (CNN) may be adjusted for Pakistan Sign Language to increase identification accuracy and usability [3].

Most Sign Language Recognition (SLR) systems use sensor-based, image-based, or hybrid techniques [2–6]. The first method fails to capture the hand's location relative to the other hand and body, which is crucial to comprehending PSL movements. The second option requires the camera to be in front of the user, which might hinder mobility. This limitation might make it difficult for users to manage the camera while doing signs in real life. Due to illumination and field of vision issues that might impact accuracy and consistency, this technique is unsuitable. Users may find the hybrid technique impractical for real-life talks since it requires a glove and camera. This configuration is both expensive and computationally intensive, limiting its acceptance and development.

Our unique Pakistan Sign Language (PSL) identification system uses a sensory glove and machine learning. This strategy makes communication more natural and effective than previous methods. The proposed SLR system recognizes static and dynamic PSL gestures, incorporating a wider range of signs and supporting the development of a real-time application that predicts signs, improves sentence grammar, and converts them into spoken Urdu sentences.

The device uses a sensory glove with an Inertial Measurement Unit (IMU) sensor (MPU9250) to track hand motions and orientation and flex sensors to quantify finger bending. The wireless glove transmits data to a processing unit without physical connections to allow user movement. PSL gestures are predicted using SVM-FE and LSTM machine learning models on the obtained data. Gemini is used to fix grammatical problems and mispredictions in the produced phrases, and Google Text-to-Speech (gTTS) in Urdu reads the result.

The paper's structure follows. SLR systems are reviewed in Section 2. Section 3 describes the Pakistan Sign Language motions used to build the dataset. This study's hardware and machine learning architectures are described in Section 4. Section 5 analyzes the dataset and reviews the machine learning models. Section 6 explains the real-time translation of PSL motions into spoken words. In Section 7, we evaluate our architecture, models, and real-time implementation and suggest improvements. Section 8 ends with observations.

2. Related Work

As illustrated in Figure 1, a Sign Language Recognition (SLR) system may be implemented utilizing a sensor-based, image-based, or hybrid technique [2–6].

A glove with many sensors and connections is worn in the sensor-based system approach. The system tracks and records hand and finger motions with these sensors. The computer receives finger bending, motions, orientation, rotation, and hand position data for interpretation. This method is mobile and accurate in gesture recognition. It lacks facial expressions and emotional clues, which Pakistan Sign Language (PSL) uses to convey complex meanings and improve communication.

A glove full of sensors and cables is not needed in the image-based method. This technology captures photos with a camera and recognizes PSL motions using computer vision and image processing. This method, which uses only a camera and CPU to construct a recognition system, is popular because to its simplicity and implementation. But continual camera alignment can disrupt the natural flow of discussion and hurt the user experience, especially in dynamic, real-world environments.

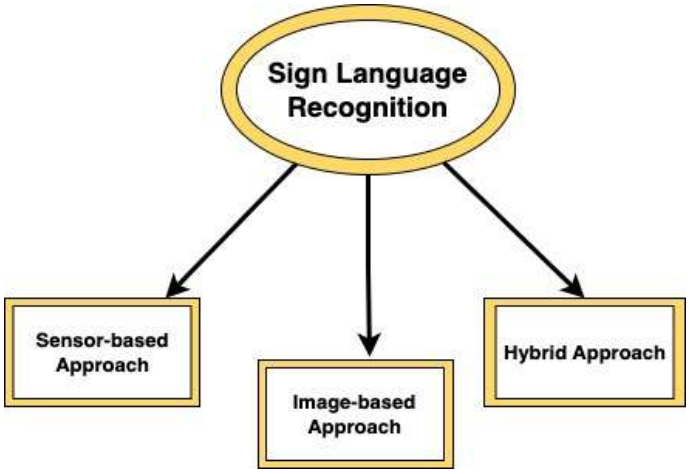

Figure 1. Main approaches for SLR system.

The third way, the hybrid strategy, combines glove- and camera-based benefits. Mutual error correction improves recognition accuracy and system dependability. However, like the image-based system, this solution has mobility issues and may not be suitable for Pakistani daily conversation.

A new Pakistan Sign Language (PSL) recognition method is presented here. This suggested system addresses the gestures, structure, and communication demands of Pakistan's deaf and hard-of-hearing people, making it more practical, real-time, and efficient than current methods).

3. Urdu Signs Overview

This section introduces the Pakistan Sign Language (PSL) movements used to build the dataset, which comprise static gestures for letters and dynamic gestures for words. For real-time recognition and phrase creation, many new control gestures were included. These are 'n' to signal a resting hand, 'space' (' ') to combine letters into words, and 'p' to concatenate words into a phrase and speak it. The package comprises the PSL alphabet, three control signs ('space', 'n', and 'p'), and 16 dynamic signs for typical PSL words. This yields 47 distinct indications sample shown in Figure 2.

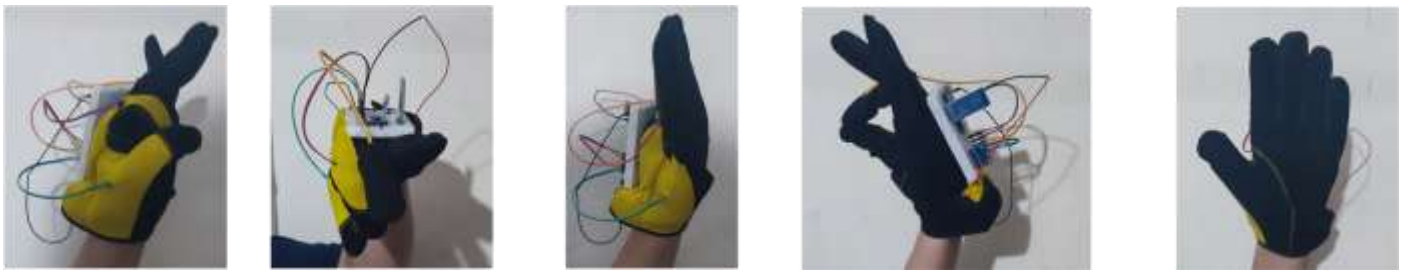

Figure 2. Sample Urdu signs used in the dynamic dataset.

4. Methodology

4.1. Block Diagram

The sensory glove system efficiently translates Pakistan Sign Language (PSL) into spoken Urdu using sensors and microcontrollers. Pakistani deaf and hard-of-hearing people can better communicate by interpreting PSL signals, creating grammatically accurate words, and speaking them.

Figure 3 shows the transmitter and receiver, which make up this work's system. The transmitter system uses a lightweight, wearing glove with sensors to record PSL hand

gestures and motions. The receiver receives this data wirelessly and utilizes machine learning techniques to properly forecast the user's sign. The algorithm then generates logical phrases from these signals and speaks them in real time in Urdu.

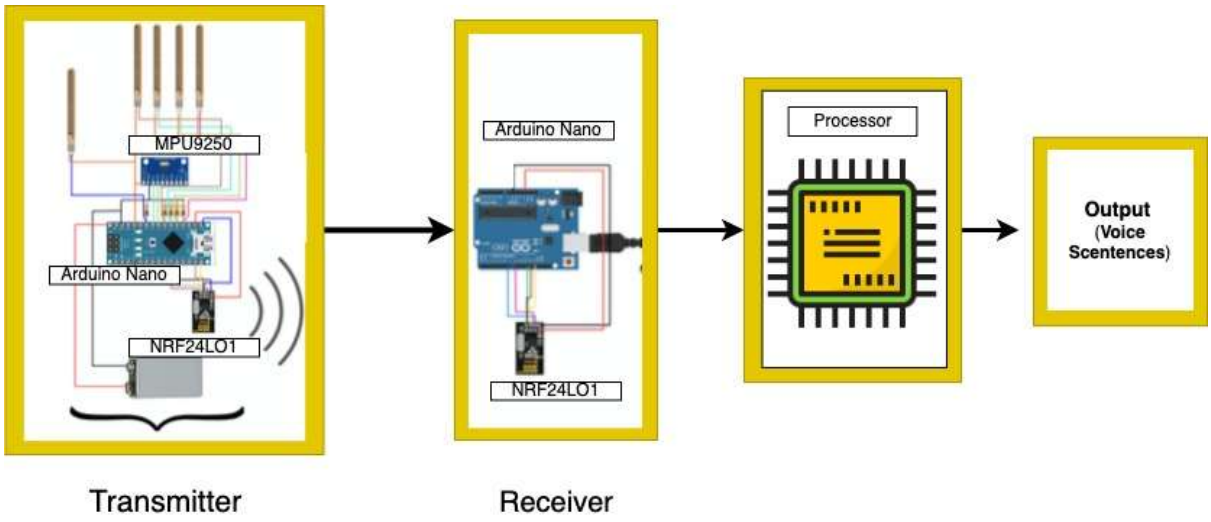

Figure 3. Transmitter–receiver block diagram.

4.2. Sensory Glove or Transmitter System

Figure 4 shows the sensory glove system's main components for Pakistan Sign Language (PSL) identification. An Arduino microcontroller [9] interacts with five flex sensors to detect finger flexing and deliver finger locations, which is needed for recognizing static PSL movements.

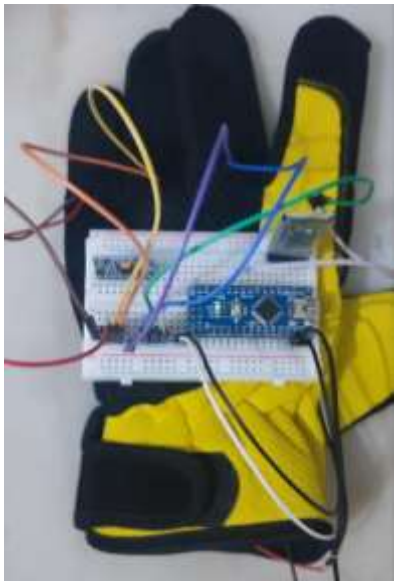

Figure 4. Picture of the actual sensory glove.

The glove uses an MPU9250 IMU to track hand orientation and movement. This sensor's accelerometer, gyroscope, and magnetometer data are essential for dynamic PSL sign recognition. This study tracks hand mobility with accelerometer data. Euler angles can affect hand orientation, but axis alignment can cause gimbal lock. Instead of gimbal lock, quaternions express 3D rotations with four components (one scalar and three imaginary), ensuring stability and accuracy.

Startup calibration ensures the IMU sensor works correctly and provides accurate results. After turning on the glove, the user must calibrate the magnetometer with a figure-8 motion. User lays hand flat and motionless on surface to calibrate accelerometer and gyroscope. This method keeps PSL gesture recognition orientation and motion data constant. The Arduino IDE's FastIMU package automates this calibration, making it simple and dependable.

Manufacturing and finger features can affect flex sensors, which detect finger bending. Flex sensor calibration standardises readings. First, the user must completely flex their fingers for 5 s. Arduino takes many measurements and calculates the median to determine flexed. Next, the user completely expands their fingers and holds for 5 s while taking further readings. The minimum and maximum reference points for each finger are then determined using the median values for flexed and stretched states.

Scalibrated sensor value is calculated using the following formula after recording these reference values:

$$\text{Calibrated Svalue} = \left(\frac{\text{Raw Value} - \text{Raw}_{\text{min}}}{\text{Raw}_{\text{max}} - \text{Raw}_{\text{min}}}\right) \quad (1)$$

Raw Value is the raw sensor measurement, Raw_{max} and Raw_{min} are the median values for fully stretched and flexed positions, respectively. This algorithm normalizes flex sensor data from 0 to 1023 to a defined range of −1 to 1, guaranteeing uniform readings across all sensors, independent of finger or sensor differences.

The five calibrated flex sensors now give precise and consistent finger position data, which is important for Pakistan Sign Language gesture detection. The whole data format contains these parameters:

The data includes quaternion components (q0, q1, q2, q3) for 3D hand orientation, accelerometer readings (ax, ay, az) for hand movement along X, Y, and Z axes, and standardized measurements (s1, s2, s3, s4, s5) for each finger's flex sensors.

The receiver module receives these sensor data wirelessly from an NRF24L01 transceiver (Figure 5). Due to the NRF24L01's 32-byte payload constraint, the data is formatted into a string and delivered in three packets: quaternion, accelerometer, and flex sensor. The transmitter delivers this structured data string at set intervals via the RF connection to the receiver, providing real-time hand gesture data.

Figure 5. NRF24L01 transceiver.

A tiny 3.7 V battery powers the glove-based device, making it portable and easy to use for real-world PSL applications including daily communication, teaching, and public engagement.

4.3. Receiver System

The receiver mechanism in Figure 3 continually listens for transmitter data packets. Initialized to listen for signals, the NRF24L01 wireless module stays active until it gets all

data. The receiver algorithm waits until all three data batches—quaternion (orientation), accelerometer (movement), and flex sensor (finger position)—are received before merging them into a data packet.

Pakistan Sign Language (PSL) gesture recognition requires synchronized and reliable incoming data, which this structured reception provides. The processing unit analyzes the complete data set to forecast PSL indications using machine learning methods. After recognizing signs, a text-to-speech technology converts a statement into spoken Urdu, allowing deaf and hearing people to communicate.

4.4. Proposed SVM with Feature Extraction (SVM-FE) Model

SVM [10] and FE are successful classification methods for Pakistan Sign Language (PSL) gesture recognition. This combination streamlines complicated sensor data into important characteristics, increasing the SVM's PSL sign recognition. Features extraction decreases input data dimensionality and highlights the most essential gesture properties, helping the SVM create clearer decision boundaries and improve classification accuracy.

For sequential PSL data, summarizing features helps the model grasp motion patterns and temporal trends in static and dynamic signals.

Table 1. presents the architecture of the SVM-FE model.

Component	Description
Input Features	Raw sensor data from 12 channels: q0–q3, ax–az, s1–s5
Feature Extraction	For each channel: Mean, Std, Min, Max, FFT Magnitude, FFT Std → 72 features total
Normalization	StandardScaler applied to all features (zero mean, unit variance)
Classifier	Support Vector Machine (SVM) with linear kernel
Output	Gesture class probabilities (47 PSL classes)

To categorize PSL signals, the SVM-FE model uses a linear kernel to produce a straight-line (or hyperplane) decision boundary in feature space. The model outputs probability estimates for each class instead of only the most likely label. This probabilistic output aids ambiguous gesture decision-making.

StandardScaler scales input characteristics to 0 and 1 for mean and standard deviation. Normalization balances features (e.g., accelerometer vs. flex sensor readings) and speeds model convergence during training, boosting identification performance.

4.5. Proposed LSTM Model

LSTM networks [11,12] handle sequential data, which is essential for identifying Pakistan Sign Language (PSL), especially dynamic motions. They excel in time-series applications like gesture sequences, speech recognition, and natural language processing because their design learns and retains long-term relationships.

The LSTM model architecture for PSL gesture prediction in our study is shown in Figure 6.

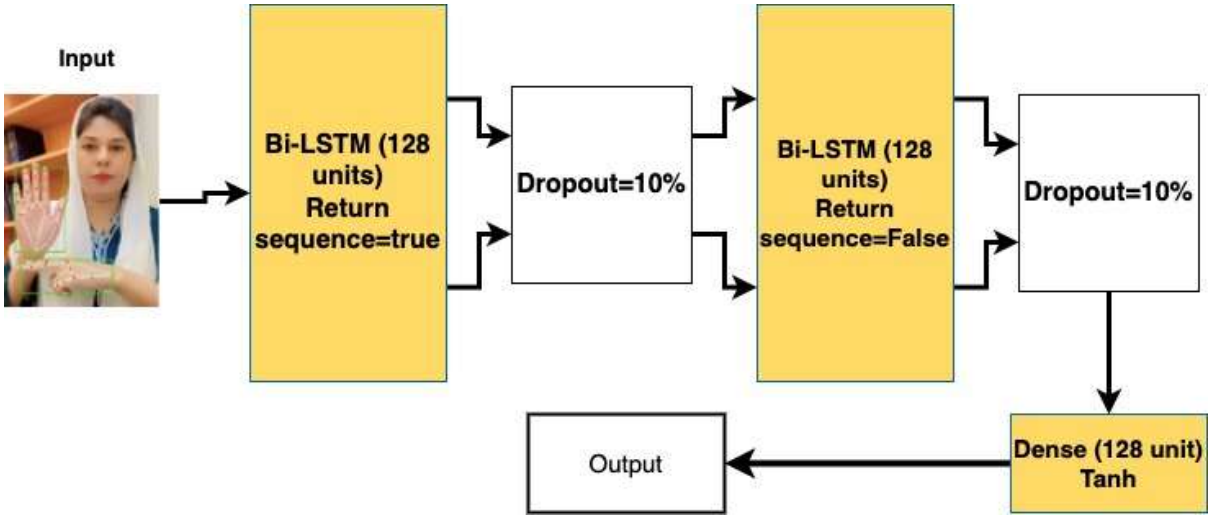

Figure 6. LSTM model architecture.

The sequential neural network model uses bidirectional LSTMs. Initial input layer has 75-time steps and 12 characteristics per time step. Quaternion components (q0, q1, q2, q3), accelerometer measurements (ax, ay, az), and flex sensor data (s1, s2, s3, s4, s5) capture PSL hand gesture motion and orientation.

A 128-unit bidirectional LSTM layer returns sequences after the input. This design lets the model learn gesture patterns in both forward and backward temporal directions, improving context awareness. After that, a 10% dropout layer randomly disables a percentage of neurons during training to minimize overfitting and promote generalization.

Next, a second 128-unit bidirectional LSTM layer is added, designed to not return sequences and output a single final state summing the input sequence's learnt information. Another 10% dropout layer reduces overfitting risk.

Non-linearity from a dense layer with 128 units and a tanh activation function lets the model record complicated gesture data correlations. The output layer is a thick softmax layer with the same number of units as PSL gesture classes. For multiclass classification and interpretable PSL sign predictions, the softmax function converts the final output into a probability distribution.

Training was stopped early to prevent overfitting. This method stops training if the model's validation loss doesn't improve after a certain number of epochs. This method stops the model from remembering noise or highly particular patterns in the training data and enhances its generalization to fresh PSL inputs.

Layer sizes and dropout rates were chosen to balance complexity and generalization in the model architecture to achieve consistent performance on unseen data during validation.

5. Experiments and Results

5.1. Dataset

Data from one participant was used to record dynamic movements for 47 Pakistan Sign Language (PSL) letters and words. The participant purposefully altered hand speed and location while completing gestures to simulate real-world variability in PSL usage by distinct users under varying ambient and physical situations to guarantee the system can generalize across users. This technique plus a comprehensive calibration method reduce individual-specific variability and improve recognition performance.

The IMU sensor data provides normalized quaternion measurements (−1 to 1), indicating 3D hand orientation. Accelerometer data, often ranging from −2 to +2, was kept

unnormalized to retain important information like abrupt hand movements or impacts during complicated PSL motions. These inherent changes in accelerometer data help the algorithm learn and distinguish swift or powerful motions that normalization would reduce.

Flex sensor data is calibrated and standardized within a −1 to 1 range to account for finger variations and sensor discrepancies. This standardization allows the algorithm to focus on gesture patterns rather than signal fluctuations, enhancing gesture recognition across users.

No data augmentation was done on this dataset. This was done to retain PSL gestures' naturalness. Artificial data may provide inconsistencies or abnormalities that hinder model learning. The dataset provides a clean, realistic basis for PSL gesture recognition model training.

For each PSL sign, 75 time steps were captured at 50 Hz, recording a 1.5-s gesture frame (75 time steps ÷ 50 Hz = 1.5 s). Each time step saves 12 features quaternion orientation (q0, q1, q2, q3), accelerometer measurements (ax, ay, az), and calibrated flex sensor values (s1–s5)—as a single row in a CSV file labeled with the PSL sign.

There are 130 readings per sign, 100 for training, 15 for testing, and 15 for validation. The dataset is balanced over all 47 PSL classes, preventing gesture bias. This structure yields 6110 gesture examples (130 readings × 47 signs), providing a broad dataset that accurately captures the dynamic and expressive nature of PSL motions. Table 2 shows a dataset sample with data format and labels.

Table 2. Sample of the dynamic dataset.

q0	q1	q2	q3	ax	ay	az	s1	s2	s3	s4	s5
0.98	0.05	0.02	0.18	0.10	−0.02	0.98	−0.4	−0.2	0.1	0.3	0.0
0.97	0.06	0.01	0.20	0.12	−0.01	0.95	−0.4	−0.2	0.1	0.3	0.0
0.98	0.04	0.01	0.19	0.11	−0.03	0.97	−0.4	−0.2	0.1	0.3	0.0
0.97	0.05	0.02	0.21	0.13	−0.02	0.96	−0.4	−0.2	0.1	0.3	0.0
0.98	0.06	0.01	0.18	0.10	−0.02	0.99	−0.4	−0.2	0.1	0.3	0.0

The box plot in Figure 7 shows accelerometer and flex sensor data-based PSL gesture class distribution. A box plot (sometimes called a box-and-whisker plot) uses five summary statistics minimum, first quartile (Q1), median, third quartile (Q3), and maximum to show data distribution and symmetry. The central box contains the interquartile range (IQR), the middle 50% of the data, from Q1 to Q3. The smallest and greatest values within 1.5 times the IQR are "whiskers"; data points beyond this range are outliers.

The box plot shows each gesture's sensor reading distribution and variability in this PSL dataset. It shows how motions create discrete sensor patterns, helping the model categorize them.

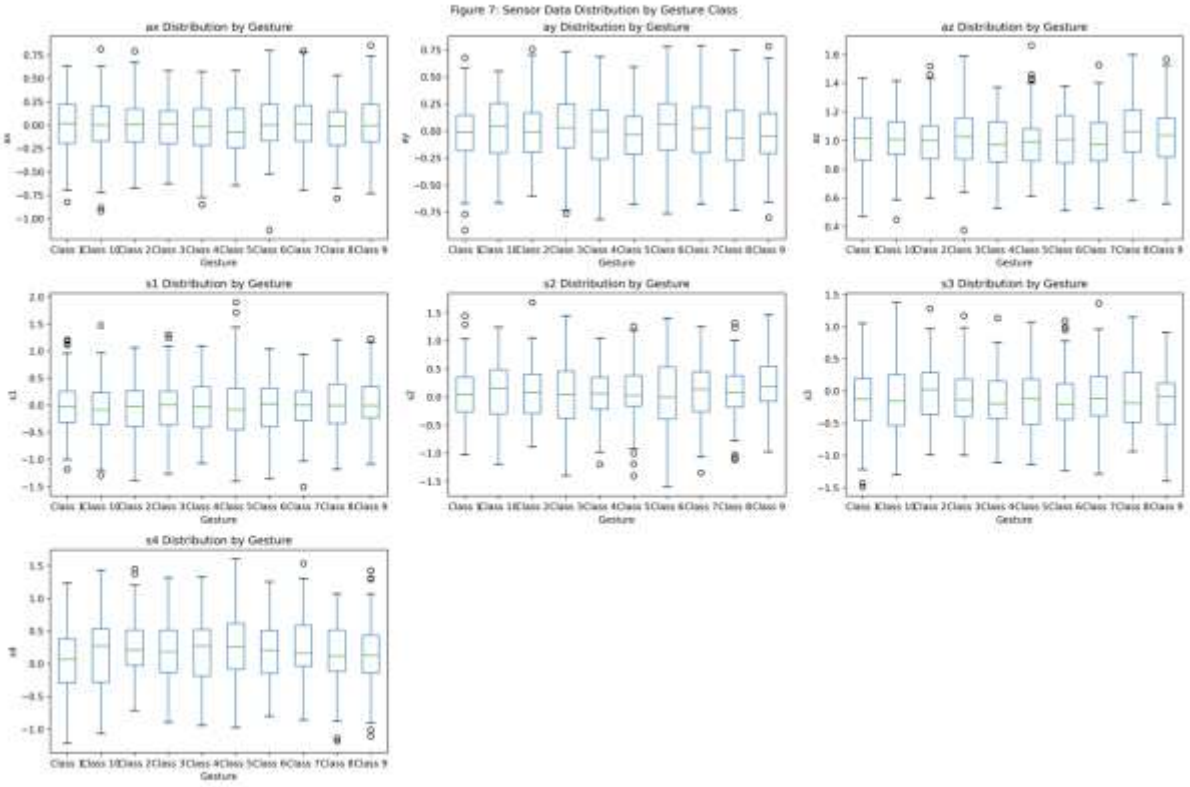

Figure 7. Box plot of multiple sensor values across different gestures.

A crucial discovery from the plots is that broader or taller boxes in either the accelerometer or flex sensor data imply a higher standard deviation, frequently linked with greater movement or unpredictability in the gesture. The accelerometer plots of static PSL movements like letter representations reveal no or very narrow boxes, indicating negligible hand movement. Dynamic gestures, like as the PSL sign for "السلام عليكم" (Peace be upon you), have bigger boxes in accelerometer data because to their strong motion.

Commonly, motions with little sensor reading fluctuation (like static signals) have more outliers. The very sensitive sensors catch slight involuntary hand movements or user changes that cause these outliers.

Note that all gesture class box plots reveal no abnormal data patterns, indicating constant and reliable sensor performance throughout data collection. No unexpected results or dramatic deviations demonstrate that sensor calibration and system integrity were maintained throughout the recording procedure, boosting dataset trustworthiness.

5.2. SVM-FE Model Results

5.2.1. Data Preprocessing for SVM-FE Model

To summarise time-series data from the dynamic gesture dataset and provide sequential PSL gesture data for SVM training, feature extraction is essential. Statistical and frequency-based characteristics are derived from raw sensor sequences to create a compact, learnable format.

Feature extraction for the PSL dataset involves determining the mean, standard deviation, minimum, and maximum for each sensor signal over 75 time steps. Each signal is Fast Fourier Transformed (FFT) to extract the amplitude and standard deviation of frequency components to reflect hand movement pattern variations over time. These features show PSL gesture temporal and spectral properties, which are essential for SVM classification.

After collecting 75 gesture data points, the system computes median, standard deviation, minimum, maximum, FFT magnitude, and FFT standard deviation for each of the 12 sensor channels (q0–q3, ax–az, s1–s5). The gesture sequence is summarized into a compact, useful 72-dimensional feature vector (6 features × 12 sensors) for SVM input.

This feature engineering method helps the SVM model discover unique PSL gesture patterns and changes, enhancing classification performance and resilience across static and dynamic sign inputs.

5.2.2. Results

Table 3 shows SVM-FE model accuracy on training, test, and validation datasets. The confusion matrices for this model's training and validation datasets are shown in Figure 8.

Table 3. SVM-FE model Performance.

Metric/Observation	Result/Description
Training Accuracy	97.8%
Validation Accuracy	94.2%
Test Accuracy	93.6%
Misclassified Static Gestures	"ج" → "ح" ,"ے" → "ر"
Cause of Confusion	Similar hand angles, overlapping quaternion and accelerometer values
Sensor Overlap Example	Flex 2–3 overlap for "ے"/"ر"; Flex 1 overlap for "ج"/"ح"

The SVM-FE model has high accuracy and a balanced confusion matrix on the validation dataset. The model's ability to mistake static PSL motions was a major drawback. Often, "ر" is misinterpreted as "ے" in the test dataset and "ح" as "ج" in the validation dataset.

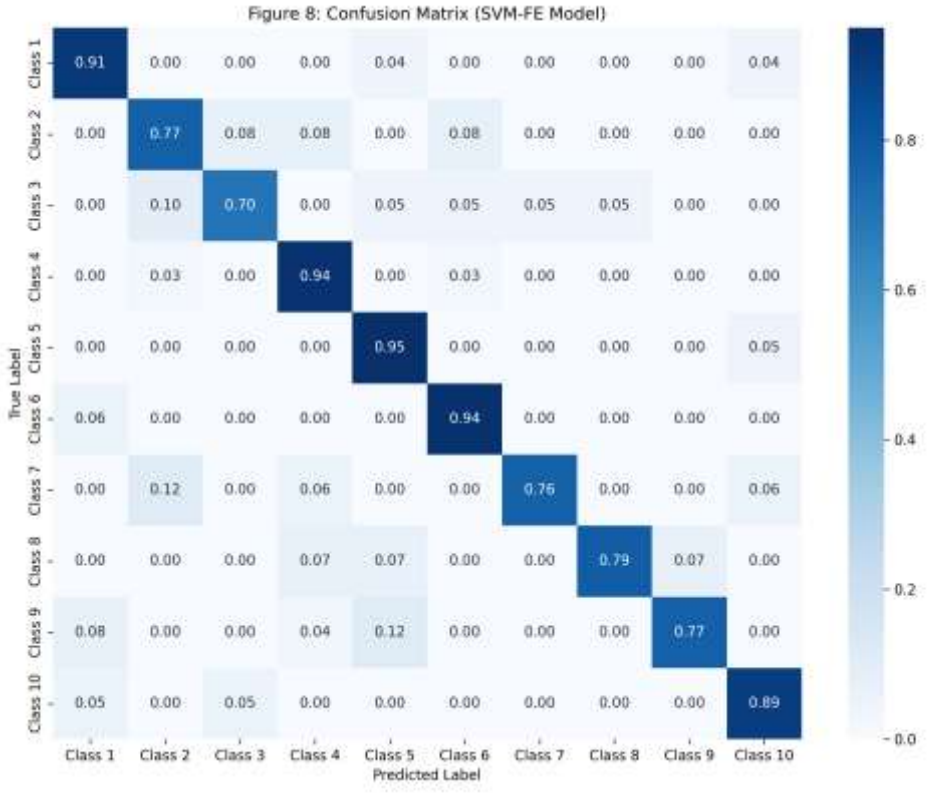

Figure 8. Confusion matrices of the SVM-FE model.

These movements are motionless and have identical hand angles, which provides comparable quaternion data and reduces accelerometer activity, causing misunderstanding. Flex sensor data is the main differentiator here. Examining SVM extracted feature histograms, flex sensor readings (particularly flex 2 and 3 for "ر" and " " in Figure 9 and flex 1 for "چ" and "ج" in Figure 10) show overlap. This overlap seems to cause most categorization errors for these motions.

More training data can increase variability and help the model learn finer differences. An improved sensor system that captures more accurate and consistent finger locations is the better and longer-term option. Flex sensors include bending sensitivity, poor angular resolution, and inability to directly monitor finger orientation, which limits their accuracy.

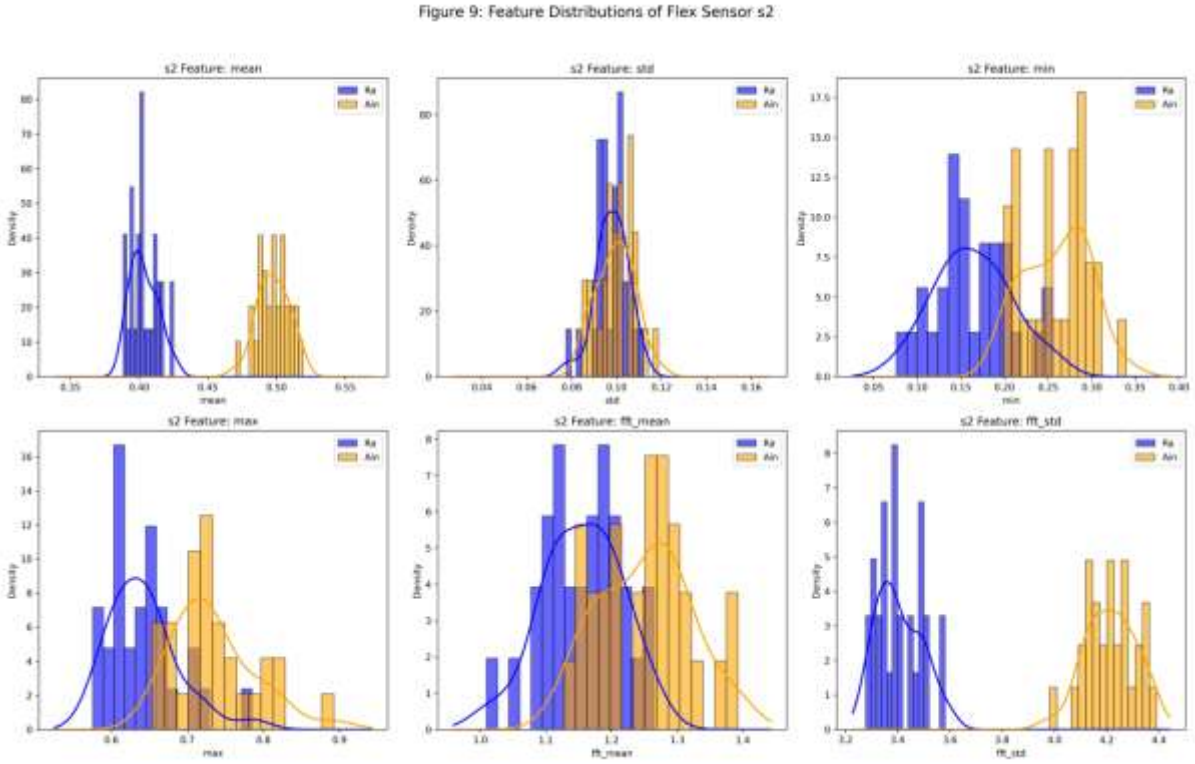

Figure 9. Histogram of feature-extracted data from flex sensors 2 and 3 for gestures.

More sophisticated sensing technologies including capacitive sensors, optical bend sensors, and multi-axis finger joint trackers might record tiny finger motions and gesture information. This will greatly enhance PSL gesture recognition accuracy, especially for static signals distinguishable just by finger location, improving system performance and robustness.

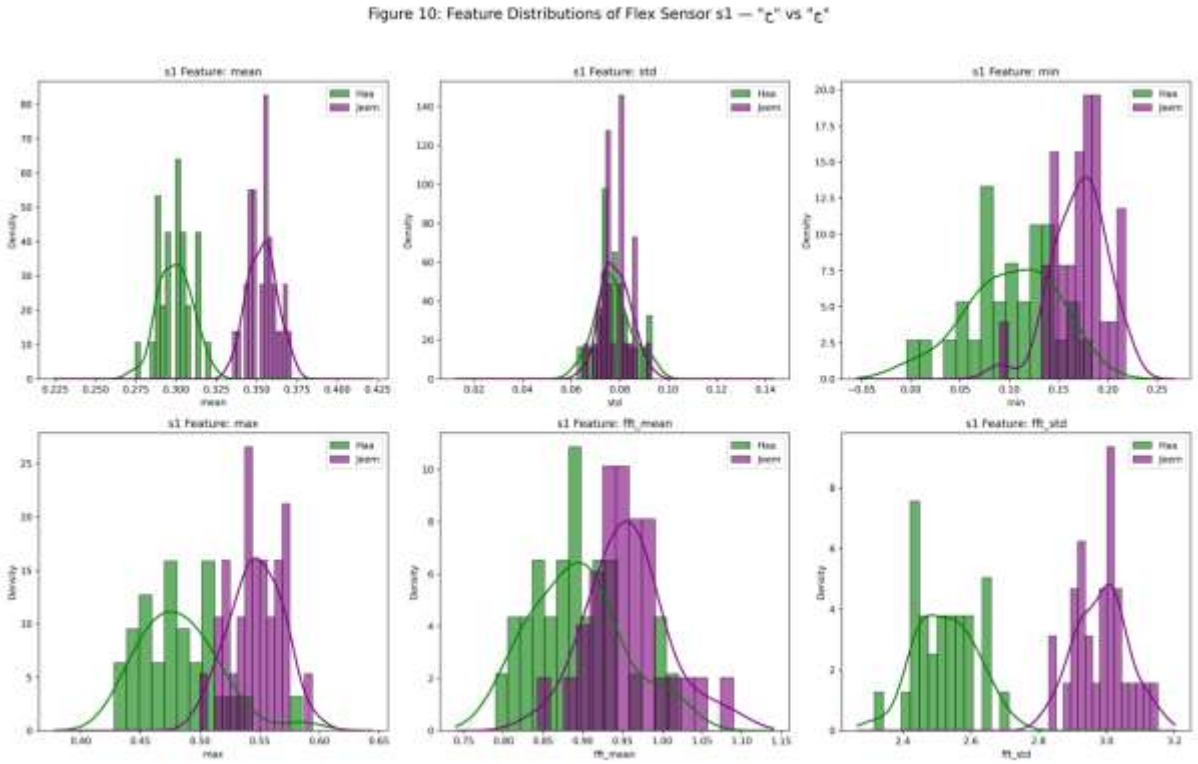

Figure 10. Histogram of feature-extracted data from flex sensor 1 for gestures.

5.3. LSTM Model Results

5.3.1. Data Preprocessing for LSTM Model

Data preparation prepares the dataset for training the LSTM model, which learns temporal relationships from sequential data. Section 5.1 describes each gesture instance in the dataset as 75 time steps with 12 attributes and a single label describing the motion.

To train the LSTM model, raw data is molded into a 75 × 12 matrix for gesture samples. Each row represents a time step, and each column represents one of 12 sensor characteristics (quaternions, accelerometer data, and flex sensor readings). This modification preserves gesture data temporal structure, allowing the model to capture dynamic patterns. Each matrix is coupled with its gesture label to create a dataset for supervised LSTM training.

5.3.2. LSTM Model Training

Figure 11 shows the accuracy and loss variation for the training and validation datasets during LSTM model training. For multi-class classification problems with integer labels, the model was created using the Adam optimizer, which is efficient with sparse gradients and non-stationary goals, and the sparse categorical cross-entropy loss function. Training focused on accuracy, the major performance parameter [10,11].

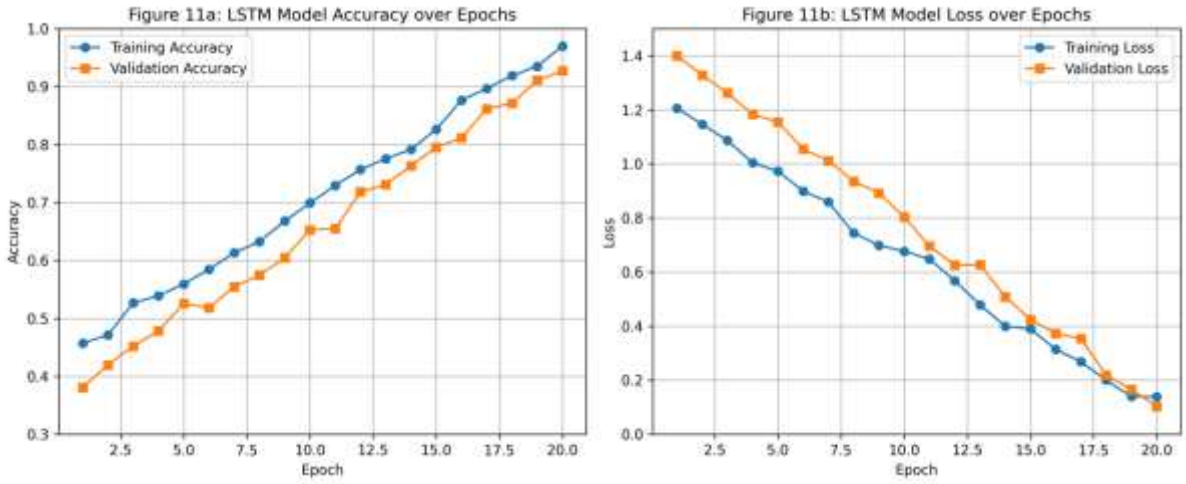

Figure 11. LSTM model accuracy and loss variation during the training process.

Training lasted 18 epochs with early pausing to prevent overfitting. If the model's validation performance didn't improve after five epochs, training would end. By keeping the algorithm from learning noise or extremely particular patterns from the training set, it generalizes effectively to unknown PSL gesture data.

5.3.3. Results

Table 4 shows the LSTM model's accuracy and loss throughout training, test, and validation datasets. Figure 12 shows training and validation confusion matrices.

Table 4. LSTM model Performance.

Metric/Observation	Result/Description
Training Accuracy	98.4%
Validation Accuracy	95.6%
Test Accuracy	94.9%
Training Loss	0.08
Validation Loss	0.12
Misclassified Gestures	"ظ" → "ذ" or "ز" (due to similar finger shapes)
Cause of Confusion	Low resolution of flex sensors for subtle finger configurations
Sensor Overlap Observed	s1/s2 values overlapping; some sensor outliers

Figure 12. Confusion matrices LSTM model.

The LSTM model classified the validation dataset with good accuracy and balanced predictions. The model may mistake the PSL sign for "ظ" → "ذ" or "ز". These signals are distinguished by subtle finger configurations, making flex sensors' accuracy difficult.

Figure 13 shows overlapping gesture distributions and outliers, which may lead to categorization uncertainty. Adding a more advanced finger-tracking sensor system that can capture fine-grained finger positions to the hardware is the same method as for the SVM model (Section 5.3.2). This would greatly enhance the model's capacity to distinguish signals with identical hand orientations but different finger articulations.

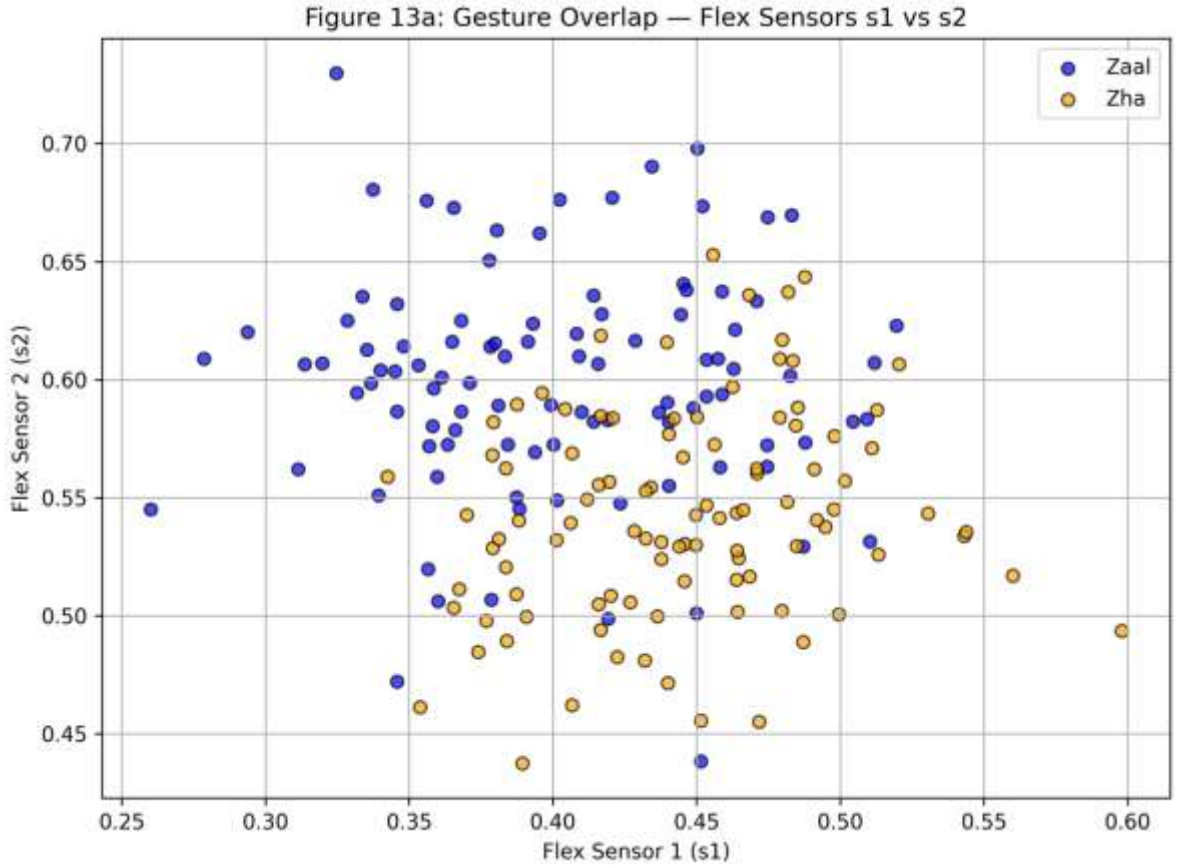

Figure 13. overlapping of flex sensor data for gestures.

Despite being trained and validated on the identical PSL gesture dataset, the SVM and LSTM models misclassified differently. The SVM uses feature-extracted summaries from time-series data, while the LSTM uses the complete sequential input, which may explain this mismatch. Thus, each model learns from separate temporal and spatial gesture inputs, resulting in complementing PSL recognition strengths and limitations.

6. Real-Time Applications

The learned machine learning model was incorporated into a real-time PSL identification system. Figure 14 shows the whole real-time implementation pipeline flow.

The algorithm loops until stopped by the user. Initial sensor data capture from the Arduino-based glove comprises flex sensor and IMU measurements. To guarantee dependability, incoming data is examined for mistakes and missing information.

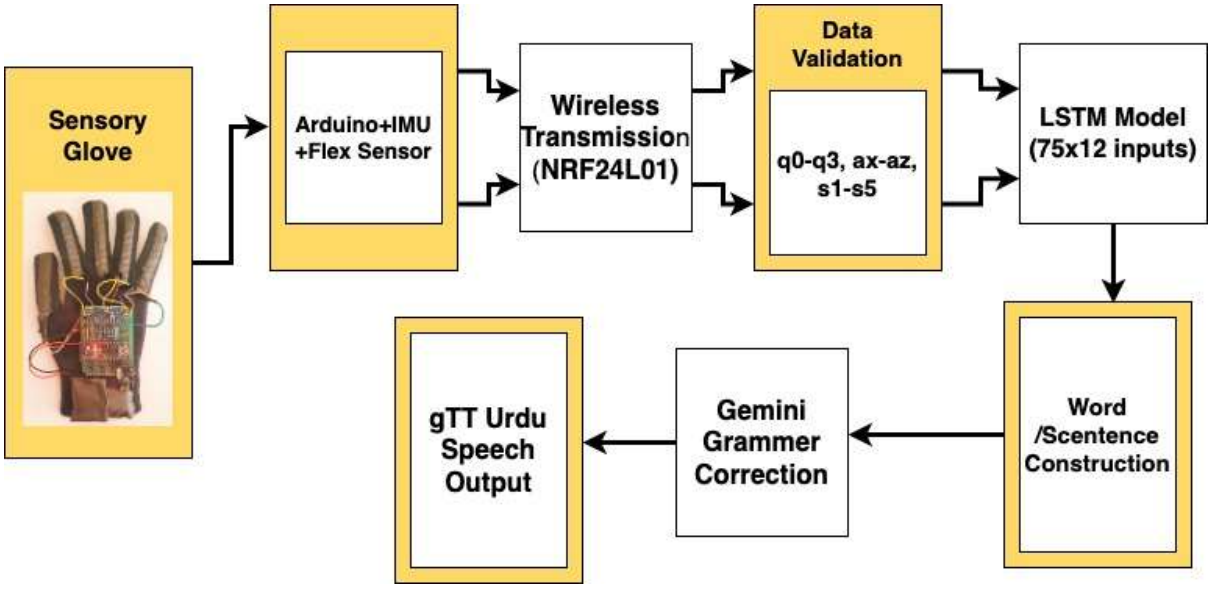

Figure 14. Flowchart of the real-time prediction application.

After validation, the LSTM model predicts the PSL alphabet letter or word in real time using sensor inputs. These predictions are processed to produce words and sentences based on sign sequence and timing.

Next, the raw output is transferred to Gemini [13], a text-refinement module that corrects grammatical and structural errors in deaf sign language communication, including direct translation artifacts from PSL to Urdu or English syntax.

The improved text is then transferred to a text-to-speech (TTS) system [14] to vocalize the statement. This end-to-end architecture allows real-time translation from Pakistan Sign Language to spoken language, helping Pakistani deaf and hard-of-hearing people communicate.

This flow's algorithms and processing stages are discussed in the next section.

6.1. Arduino Error Handling and Machine Learning Prediction

By collecting sensor data from the receiver Arduino module, the laptop processing unit is crucial to the PSL identification system. Flex sensor (finger positions) and IMU (hand orientation and movement) measurements from the sensory glove worn by the PSL sign performer are wirelessly communicated.

The technique checks for 12 comma-separated float values in a data packet to ensure its completion. They represent the 4 quaternion components (q0–q3), 3 accelerometer components (ax, ay, az), and 5 calibrated flex sensor values (s1–s5).

The system discards incomplete or faulty data packets and waits for the next valid reading. After confirming a successful reading, the program collects 75 full and valid readings, representing 1.5 s of signing activity at 50 Hz.

The data is reorganized into a 75 × 12 matrix to meet LSTM model input criteria. This redesigned matrix lets the model distinguish PSL gesture timing patterns and predict the signed letter or word.

6.2. Constructing Sentences

The system verifies sign confidence after the LSTM model processes the input matrix and predicts. The prediction is reliable and utilized to create Pakistan Sign Language words and phrases if the confidence level is greater than 0.85. These validated signals are sequenced to help the system comprehend continuous signing as text.

If the confidence score drops below 0.85, the algorithm discards the forecast and starts over to collect sensor data. This method improves PSL identification accuracy and reliability, especially in real-time situations where misclassification might impair communication.

6.2.1. Constructing Words from Letters

The letter buffer stores expected gestures that are PSL letters, not control signs like 'space', 'p' for pause, or 'n' for noise. This buffer stores recognized letters that compose a word.

If the expected gesture is 'space' and the letter buffer is not empty, the algorithm connects the letters to form a word. That word goes into the word buffer, which organizes words for sentences. The letter buffer is cleaned after storing the word to receive the next anticipated letters.

6.2.2. Constructing Sentences from Words

The word buffer stores expected gestures that are not control signs (' ', 'p', or 'n') and do not represent a single letter as whole word gestures.

If the expected gesture is 'p' (pause) and the word buffer is not empty, the algorithm concatenates the words to produce a sentence. Gemini then polishes this statement. Clearing the word buffer prepares it for the following phrase.

6.2.3. Hand Resting

The projected gesture of 'n' is eliminated since it indicates a resting or neutral hand. The system does not save the forecast and returns to the Arduino input stage for new sensor data.

6.3. Sentence Correction with Gemini

Google's Gemini LLM understands and generates human-like text from numerous inputs [13]. It provides grammatically accurate and relevant replies using powerful natural language processing. Gemini's open API makes it appropriate for many applications, including deaf aids.

Gemini supports Urdu, which is important for this Pakistan Sign Language project. Sign language conveys meaning well, but its translation into written or spoken language typically lacks grammatical structure. PSL uses "I go market" whereas Urdu uses "میں بازار جا رہی ہوں."

Gemini helps refine translated outputs into well-structured, flowing language. This eliminates communication failures and deaf people feeling ashamed or misunderstood. After correction, a text-to-speech generator reads the statement.

This project for Gemini uses an Urdu prompt:

"آپ اپنے بہرے ساتھی کے لیے ترجمہ کرتے ہیں۔ آپ کو اُس کے ضروری الفاظ سے ایک مربوط اور بامعنی جملہ بنانا ہوگا۔ براہِ کرم نوٹ کریں کہ الفاظ میں املا کی غلطیاں یا نامناسب الفاظ ہو سکتے ہیں، لہٰذا آپ کو معنی کی بنیاد پر درست جملہ تیار کرنا ہوگا۔"

6.4. Text-to-Speech Generator

This document uses Google's gTTS service for TTS conversion [14]. Urdu support made this service ideal for a Pakistan Sign Language (PSL)-based communication system. Free and limitless access is another benefit of gTTS for systems that need constant use.

Other TTS systems generate more lifelike and expressive voices, although most charge or have restricted usage options.

After the machine learning model identifies PSL motions and the Gemini language model corrects the phrase for syntax and clarity, gTTS outputs the sentence as spoken

audio. Deaf people can clearly converse with hearing people. The system then returns to input to accept more gestures, generate new phrases, and continue the discussion in real time.

6.5. Timing Analysis

The time efficiency of each gesture recognition step may be determined using statistical data from repeated user trials of the Pakistan Sign Language (PSL) recognition system in real time.

The median Arduino data collection time (75 data points) is 1.586 s. Wireless transmission delays and packet loss cause small differences from the predicted 1.5 s.

The LSTM model processes input and predicts gestures in 0.14 s after data collection. To ensure clarity and system readiness, the following motion is captured after a 0.5-s wait to tell the user.

A typical PSL gesture recognition cycle takes 2.226 s.

Take the example of a user signing "Hello, my name is Tabassum. I am going to market

"ہیلو ، میرا نام تبسم ہے۔ میں بازار جا رہی ہوں"

The PSL gesture-based system requires the user to do the following sequence: "p, Hello, name, T, A, B, A, S, U, M me, market, p" where "p" is the punctuation or sentence-completion gesture.

Ten motions take 22.26 s (based on an average of 2.226 s per gesture). This estimate assumes a smooth process without Gemini-filtered low-confidence predictions or misclassifications.

The last 'p' gesture activates text-to-speech (TTS) after the gesture sequence. For a brief statement, the system requires ~3.62 s to comprehend and speak it. However, sentence length and complexity may affect its time.

7. Discussion

We outperformed previous research in model performance and evaluation. Unlike many earlier studies, our system was trained using 47 PSL gestures, including static (letter) and dynamic (word) signals. Our technique works better in real life with additional data.

Our real-time sensory glove technology transforms PSL motions into spoken Urdu, expanding prior studies. A glove-based technique avoids cameras and cumbersome hardware. Its little weight, simplicity of use, and natural hand movement let deaf individuals communicate daily.

The system uses IMU and flex sensor data and deep learning models like LSTM and SVM-FE for good recognition accuracy. Flex sensors can miss small finger motions, thus there are still some restrictions. Misclassifications may ensue. EMG (Electromyography) sensors for muscle signals or MPU9250 sensors on each finger for orientation tracking may help.

The method does not monitor hand location relative to the body or other hand, which is essential for capturing PSL's grammatical structure. Some PSL signals use hand location or two-hand interactions for context.

Both test and validation accuracy were marginally higher for the LSTM model than the SVM-FE. SVM-FE is still an excellent choice for resource-constrained applications because to its decreased computing load. Table 5 summarizes results.

Table 5. summarizes results.

Criteria	SVM-FE Model	LSTM Model
Model Type	Support Vector Machine	Recurrent Neural Network (LSTM)
Input Format	72-dimensional feature vector	75 × 12 time-series matrix
Training Accuracy	97.8%	98.4%
Validation Accuracy	94.2%	95.6%
Test Accuracy	93.6%	94.9%
Prediction Speed	Fast	Moderate
Computational Load	Low	Medium to High
Temporal Context Handling	No	Yes
Best Use Case	Mobile, low-power devices	Real-time, dynamic gesture understanding
Sensitivity to Noise	Higher (smooth features)	Moderate (benefits from temporal smoothing)
Scalability	Limited	High

LSTM is more scalable and suitable for adding more complicated movements and terminology. However, the SVM-FE model may work for low-power devices like mobile phones and embedded systems.

Despite high real-time performance and precise gesture-to-speech translation, the system only offers one-way communication from deaf to hearing users. Future systems should use voice recognition and animated PSL avatars or video-based interpretation for speech-to-sign translation to facilitate two-way communication.

For improved system effectiveness in Pakistan, future study should aim to:

- Increase PSL dataset quantity and variety to reduce latency.
- Improving recognition using Transformer-based models.
- Enhancing hardware ergonomics for daily usage.
- Supporting Urdu linguistic subtleties and regional PSL variants.

This research establishes a good foundation for PSL-based communication tools but requires additional refinement to properly overcome the communication gap between deaf and hearing Pakistanis.

8. Conclusions

This study tries to solve communication hurdles for deaf people. This research describes a real-time Urdu Sign Language Recognition (SLR) system using a sensory glove and machine learning algorithms. We discuss contemporary SLR technologies, our dataset's Urdu sign motions, and the system's hardware architecture and model design. The dataset and models are assessed for real-time gesture-to-spoken language translation. We also address system enhancements, including hardware restrictions and model upgrades, and future development. This study lays the groundwork for practical SLR systems, but it may be expanded and refined to better serve users and enhance sign language recognition technology.

Author Contributions: Conceptualization, T.K. and S.A.; methodology, T.K.; software, K.S.; validation, T.K., S.A. and R.M.Y.; formal analysis, T.K.; investigation, T.K.; resources, K.S.; data curation, R.M.Y.; writing—original draft preparation, T.K.; writing—review and editing, S.A.; visualization, R.M.Y.; supervision, K.S.; project administration, T.K.; funding acquisition, S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Our study was reviewed and approved by the Institutional Research Ethics Committee of PMAS-Arid Agriculture University, Rawalpindi.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The dataset generated and analyzed during this study is available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. World Health Organization. Deafness and Hearing Loss. 2021. Available online: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss (accessed on).
2. Kothadiya, D.; Bhatt, C.; Sapariya, K.; Patel, K.; Gil-Gonzalez, A.B.; Corchado, J.M. Deepsign: Sign language detection and recognition using deep learning. *Electronics* **2022**, *11*, 1780.
3. Zhang, Y.; Jiang, X. Recent advances on deep learning for sign language recognition. *Comput. Model. Eng. Sci. (CMES)* **2024**, *139*, 2399–2450.
4. Saeed, Z.R.; Zainol, Z.B.; Zaidan, B.B.; Almoodi, A.H. A systematic review on systems-based sensory gloves for sign language pattern recognition: An update from 2017 to 2022. *IEEE Access* **2022**, *10*, 123358–123377.
5. Alsharif, B.; Altaher, A.S.; Altaher, A.; Liyas, M.; Alalwany, E. Deep learning technology to recognize american sign language alphabet. *Sensors* **2023**, *23*, 7970.
6. Abdullahi, S.B.; Chamnongthai, K. American sign language words recognition of skeletal videos using processed video driven multi-stacked deep LSTM. *Sensors* **2022**, *22*, 1406.
7. Chai, J.; Zeng, H.; Li, A.; Ngai, E.W.T. Deep learning in com- puter vision: A critical review of emerging techniques and application scenarios. *Mach. Learn. Appl.* **2021**, *6*, 100134.
8. Bengio, Y.; LeCun, Y.; Hinton, G. Deep learning for AI. *Commun. ACM (CACM)* **2021**, *64*, 58–65.
9. Monk, S. *Programming Arduino: Getting Started with Sketches*, 3rd ed.; McGraw Hill TAB: USA, 2022.
10. Christianini, N.; Shawe-Taylor, J. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, 1st ed.; Cambridge University Press: Cambridge, UK, 2020.
11. Kelleher, J.D.; Namee, B.M.; D'Arcy, A. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*, 2nd ed.; The MIT Press: Cambridge, UK, 2020.
12. Geron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow*, 3rd ed.; O'Reilly Media Inc.: USA, 2022.
13. Google Gemini API Documentation. 2024. Available online: https://ai.google.dev/api?lang=python (accessed on).
14. Google Text-to-Speech Documentation. 2024. Available online: https://gtts.readthedocs.io/en/latest/ (accessed on).