Int J Syst Assur Eng Manag
https://doi.org/10.1007/s13198-024-02376-x

ORIGINAL ARTICLE

An open-source MP + CNN + BiLSTM model-based hybrid model for recognizing sign language on smartphones

Hayder M. A. Ghanimi[1,2] · Sudhakar Sengan[3] · Vijaya Bhaskar Sadu[4] · Parvinder Kaur[5] · Manju Kaushik[6] · Roobaea Alroobaea[7] · Abdullah M. Baqasah[8] · Majed Alsafyani[9] · Pankaj Dadheech[10]

Received: 10 September 2023 / Revised: 5 March 2024 / Accepted: 16 May 2024

Abstract The communication barriers experienced by deaf and hard-of-hearing individuals often lead to social isolation and limited access to essential services, underlining a critical need for effective and accessible solutions. Recognizing the unique challenges this community faces—such as the scarcity of sign language interpreters, particularly in remote areas, and the lack of real-time translation tools. This paper proposes the development of a smartphone-runnable sign language recognition model to address the communication problems faced by deaf and hard-of-hearing persons. This proposed model combines Mediapipe hand tracking with particle filtering (PF) to accurately detect and track hand movements, and a convolutional neural network (CNN) and bidirectional long short-term memory based gesture recognition model to model the temporal dynamics of Sign Language gestures. These models use a small number of layers and filters, depthwise separable convolutions, and dropout layers to minimize the computational costs and prevent overfitting, making them suitable for smartphone implementation. This article discusses the existing challenges handled by the deaf and hard-of-hearing community and explains how the proposed model could help overcome

✉ Sudhakar Sengan
sudhakarsengan@psncet.ac.in

Hayder M. A. Ghanimi
hayder.alghanami@uowa.edu.iq

Vijaya Bhaskar Sadu
sadhu.vijay@gmail.com

Parvinder Kaur
pkaur@ccet.ac.in

Manju Kaushik
mkaushik@jpr.amity.edu

Roobaea Alroobaea
r.robai@tu.edu.sa

Abdullah M. Baqasah
a.baqasah@tu.edu.sa

Majed Alsafyani
alsufyani@tu.edu.sa

Pankaj Dadheech
pankajdadheech777@gmail.com

[1] Information Technology Department, College of Science, University of Warith Al-Anbiyaa, Karbala 56001, Iraq

[2] Department of Computer Science, College of Computer Science and Information Technology, University of Kerbala, Karbala 56001, Iraq

[3] Department of Computer Science and Engineering, PSN College of Engineering and Technology, Tirunelveli, Tamil Nadu 627152, India

[4] Department of Mechanical Engineering, Jawaharlal Nehru Technological University, Kakinada, Andhra Pradesh 533003, India

[5] Department of Electronics and Communication Engineering, Chandigarh College of Engineering and Technology, Degree Wing, Chandigarh 160019, India

[6] Amity Institute of Information Technology (AIIT), Amity University Rajasthan, Jaipur 303002, India

[7] Department of Computer Science, College of Computers and Information Technology, Taif University, P. O. Box 11099, 21944 Taif, Saudi Arabia

[8] Department of Information Technology, College of Computers and Information Technology, Taif University, 21974 Taif, Saudi Arabia

[9] Department of Computer Science, College of Computers and Information Technology, Taif University, P. O. Box 11099, 21944 Taif, Saudi Arabia

[10] Department of Computer Science and Engineering, Swami Keshvanand Institute of Technology, Management and Gramothan (SKIT), Jaipur, Rajasthan 302017, India

Published online: 30 May 2024

these challenges. A MediaPipe + PF model performs feature extraction from the image and data preprocessing. During training, with fewer activation functions and parameters, this proposed model performed better to other CNN with RNN variant models (CNN + LSTM, CNN + GRU) used in the experiments of convergence speed and learning efficiency.

Keywords CNN · Sign language recognition · Gesture recognition model · BiLSTM, hand gesture recognition · Mobile app

1 Introduction

The Interest in creating technologies that can help people with disabilities has grown recently. The Sign Language Recognition Model (SLRM) is one tool used to help people with hearing impairment or who try to communicate with those who do not recognize Sign Language (SL) more successfully (Al-Sarayrah et al. 2021). There is a chance to create a smartphone-runnable SLRM on the go because smartphones are commonly available (Bragg et al. 2019; David et al. 2023). This novel idea addresses many current issues affecting the deaf and hard-of-hearing community, including communication difficulties that can frequently result in social isolation and restricted access to vital services (healthcare, employment, and education).

Deaf people find it challenging to communicate when an interpreter is not accessible due to the unavailability of real-time translation tools that translate SL into spoken or written language (Godwin-Jones 2017; Deng et al. 2009). The scarce supply and higher cost of SL interpreters, particularly in rural areas, may make these difficulties even more difficult. Also, there might be challenges for individuals who did not use it while developing, adding another communication obstacle (Donahue et al. 2015). However, a smartphone-runnable Sign Language Recognition (SLR) model presents unique challenges (Donahue et al. 2015).

The research on developing a smartphone-compatible Sign Language Recognition Model (SLRM) faces significant challenges, including optimizing for limited computational resources on mobile devices, accommodating the inherent variability in sign language due to individual differences in signing speed and style, ensuring real-time processing capabilities for live video data, overcoming the scarcity and diversity of sign language datasets, integrating temporal dynamics of sign language gestures, addressing user privacy and data security concerns, adapting to environmental variations like lighting and background, and ensuring the system is user-friendly and accessible. Addressing these challenges is crucial for creating a practical and efficient SLRM that can significantly improve communication accessibility for the deaf and hard-of-hearing community.

This work proposes a Gesture Recognition Model (GRM), which accurately detects and tracks hand movements by combining PF with Mediapipe hand tracking. The data and Feature Extraction (FE) are preprocessed using Mediapipe, a Google-developed open-source framework. The likelihood of hand motions is predictable using PF, a Bayesian Filtering Technique (BFT) established on the movement and location of the hand over time. This algorithm supports a more reliable and exact Hand Gesture Recognition (HGR) solution.

This study also recommends a GRM that mimics the temporal dynamics of SL gestures using CNN and BiLSTM. This model consists of Fully Connected (FC), BiLSTM, input, convolutional, and depthwise discrete convolutional layers. In order to minimize the size of variables and setup cost, the proposed model implies depthwise separable convolutions, a few layers, and filters. It is improved for smartphone locations because of the dropout layers that prevent overfitting. This study aims to develop a smartphone-runnable SLRM that can manage differences in signing speed and style, accurately recognize a standard range of signs, and work efficiently on a smartphone, thus not getting compromised with accuracy by integrating these two methods. For better access of deaf and hard-of-hearing people to education, employment, healthcare, and social relations, this model assists in removing the obstacles to communication.

The following are the subsections that make up this research article: The review of the relevant literature are found in Sect. 2, the contextual analysis of the recommended model is found in Sect. 3, the proposed model is found in Sect. 4, the experimental results and analysis are found in Sect. 5, and Sect. 6 provides a summary and conclusion.

2 Related works

There is a growing trend to use CNNs in different fields of Computer Vision (CV) because of their effectiveness in object identification and classification tasks (Bantupalli and Xie 2018; Simonyan and Zisserman 2014). To execute video analysis tasks, CNNs were first extended for video action and activity detection and reached innovative outcomes (Karpathy et al. 2014; Krizhevsky et al. 2017).

There are many methods to extract from video data all the information related to spatiotemporal using CNNs. The successful application of 2D + CNNs to static images led to the first use of CNNs in video analysis-based techniques. According to Wang et al. (2016), video frames are multi-channel inputs to 2D + CNNs.

In order to extract data from optical flow modalities and colour for each segment through 2D + CNNs, the Temporal

Segment Network (TSN) (Li et al. 2020) classifies video into multiple segments. Spatio-temporal modelling is used to apply action recognition. The authors proposed a convolutional Long Short-Term Memory (LSTM) model in Lionel et al. (2017).

The recommended model and the researchers first used a 2D + CNN to FE from video frames before using LSTM for global temporal modelling. All these methods' strength stems from the abundance of phenomenally successful 2D + CNN designs, which are trained using the colossal ImageNet dataset (Lionel et al. 2014).

Convolutional Neural Networks (CNNs) are trained to classify signs due to advancements in learning-based methods (Longlong et al. 2019). A 3D-CNN is used to recognize American SL in videos. Using a subsample of 50 labels from the ASLLVD dataset, the proposed model achieves 90% accuracy. Also, using 3D + CNNs on American SL achieved 92.88% accuracy across 100 classes.

Mercurio 2021) achieved 91.7% accuracy, for which 20 words were evaluated from the Italian SL using a CNN. (Oscar et al. 2016) achieve 50.6% accuracy across 60 classes when labelling signs from Danish SL using an iterative EM algorithm and CNN.

Panning et al. 2021) achieved a Top-10 accuracy of 73.5% and 56.4% when 100 signs were assumed from each Dutch and Flemish SL using a CNN with residual connections. Using the Inception network on an image American SL dataset, (Girshick et al. 2014) obtained 91% accuracy across 150 classes. HGR has been researched, but smartphone real-time gesture detection hasn't (Saleh et al. 2020; Subramanian et al. 2022; Tao et al. 2017) (Table 1).

3 Proposed model

3.1 Mediapipe framework

An open-source model-based hybrid proposal called MediaPipe is used to build processing pipelines for perceptual data like audio, videos, and images. This complete method uses ML for real-time hand tracking and Gesture Recognition (GR). By precisely identifying the sign gestures, it offers different hand and finger tracking solutions. In particular, the hands, face, and body pose signs are captured using the MediaPipe pipeline.

Table 1 Summary of comprehensive literature review

References	Methodology used	Research gap
Bantupalli and Xie 2018)	Video sequence analysis, Inception CNN and RNNs for American Sign Language Dataset	Lack of integration with real-time applications on smartphones
Simonyan and Zisserman 2014)	Two-stream ConvNet architecture for spatial and temporal information capture in videos	Limited exploration of combined spatial–temporal models for sign language on mobile devices
Karpathy et al. 2014)	CNNs for large-scale video classification, leveraging spatio-temporal information	Need for efficient, smartphone-compatible architectures
Krizhevsky et al. 2017)	Deep CNN with dropout for image classification and regularization	Challenges in deploying deep learning models on resource-constrained devices
Wang et al. 2016)	Temporal Segment Network (TSN) framework for video-based action recognition	Limited focus on sign language recognition in TSN applications
Li et al. 2020)	Introduction of large-scale Word-Level ASL video dataset, Pose-TGCN for gesture recognition	Dataset availability and handling in mobile contexts
Lionel et al. 2017; Lionel et al. 2014; Longlong et al. 2019)	Temporal convolutions, 3D CNNs, multimodality features (RGB-D videos, skeleton joints)	Exploration of multimodal inputs for mobile-based recognition systems
Mercurio 2021)	Sign language recognition for Italian Sign Language (LIS), LIS2Speech technology	Focus on LIS with limited application to other sign languages on smartphones
Oscar et al. 2016)	CNN within an iterative EM algorithm for hand shape recognition and continuous sign language recognition	Utilization of loosely labeled video data for mobile deployment
Panning et al. 2021)	Survey on audiologists' knowledge and motivation to learn ASL, highlighting the need for inclusive education	Integration of Deaf culture and sign language into mobile applications for healthcare
Girshick et al. 2014; Saleh et al. 2020; Subramanian et al. 2022)	R-CNN for object detection, D-talk system, and MOPGRU model for gesture recognition	Applying advanced object detection techniques to mobile sign language recognition
Tao et al. 2017)	Multi-view 3D-CNN model for extracting comprehensive spatiotemporal features from videos	Leveraging multi-view learning for sign language recognition on smartphones

3.1.1 Mediapipe poses landmarks

Using the BlazePose detector, the MediaPipe body pose model discovers the person/pose Regions Of Interest (ROI) inside the frame by inferring about 33 3D landmarks on the body with (x,y,z) coordinates from the video or input image. The division masks and pose landmarks within the ROI chronologically detect postures using the ROI-cropped frame input. As a result, it aptly localizes more vital locations and correctly matches SLR.

3.1.2 MediaPipe hand landmarks

MediaPipe Hand Landmarks in Hand. By integrating two models, in MediaPipe hands, there has been an inference of about (Saleh et al. 2020) -3D hand landmarks made up of coordinates like (x, y, z) in one frame alone: the hand keypoint localization and the palm detection model. The model was initially put to use with a single-shot detector by the name of Blazing Palm. This detector uses MediaPipe to reduce the time complexity of detecting palms when the input image contains a complete dataset of hand sizes. This model analyses the image, producing a focused bounding box. This bounding box emphasizes the rigid regions—like the fist and palm—instead of focusing on extraneous objects to recognize palms. The model then performs hand keypoint localization using the palm detection outputs a result.

3.1.3 The following three outcomes are possible:

- 21 points on the hand's knuckles in a 2D and 3D area.
- Hand flag exhibiting that if, in the input image, a hand is likely to be present.
- Classification of left and right hands as "binary".

3.2 Particle filtering (PF) algorithm

A technique for finding the state of a system based on erratic observations is known as Particle Filtering (PF), commonly referred to as Monte Carlo Filtering (MCF). It is used in areas such as control theory and signal processing.

Each set of particles in PF represents a potential state-of-the system, which is how the system's state is denoted. The particles are updated at each time step following the experimental data and the system model. The low-probability particles are removed to keep representativeness, and the remaining particles are resampled. The following are the fundamental steps of PF.

Algorithm 1: PF Algorithm

Input: x_o: Initial state value.

$z_{1:t}$: Sequence of observed quantities

Output: x_t^{est}: Estimated state values at each time step

Initialization:

- Generate N particles, $\{x_0^i\}$, i=1...N, beginning the prior distribution $p(x^0)$.
- Set the first particle weights $\{\omega_0^i\}$, i=1...N, to be equal, ω_0^i=1/N.
- Set t=1.

Prediction Update:

- For Each particle i, sample a new state from the importance probability density function $q(x_t \mid x_{t-1}^i, z_t)$.
- Update the particle weight based on the likelihood of the new state given the earlier state and observation, $\hat{\omega}_t^i = \frac{\omega_{t-1}^i p(z_t|x_t^i) p(x_t^i|x_{t-1}^i)}{q(x_t^i|x_{t-1}^i, z_t)}$
- Normalize the particle weights, $\{\omega_t^i\}$, by computing: $\omega_t^i = \frac{\hat{\omega}_t^i}{\sum_{j=1}^N \hat{\omega}_t^j}$

Resampling:

- If the adequate number of particles $\left(1/\sum_{i=1}^N (\omega_t^i)^2\right)$is less than a threshold value, resample particles with a substitute from the current set of particles, $\{x_{t-1}^i\}$, i=1...N, using the normalized weights as the sampling probabilities.
- Otherwise, continue with the current set of particles.

State Estimation:

- Compute the estimated state at 'a' time 't' as the weighted average of the particles, $x_t^{est} = \sum_{i=1}^N \omega_t^i . x_t^i$
- Increment t by 1 and repeat from step 2 until t = T.

Result:

- Return the estimated state values, $\{x_t^{est}\}$, for each time step.

3.3 Depth-wise separable convolution

CNN employs a technique called depth-wise separable convolution (Fig. 1) that can aid in lowering the convolution layers' memory demands and computational difficulty. It entails splitting a convolution into two distinct functions: Convolutions that are point- and depth-wise (1×1 convolution).

Point-wise convolution blends feature maps across channels, whereas depth-wise convolution uses a different convolution kernel over each image data channel. Point-wise convolution creates new spatial traits, and depth-wise convolution retrieves them from each channel.

The output channels (filters) in the point-wise convolution are epitomized by 'N', and the kernel size used in the depth-wise convolution (assumed to be a square kernel with width and height equal to 'k') is meant by 'k'. The depth-wise separable convolution decreases the number of parameters and computations required when compared to

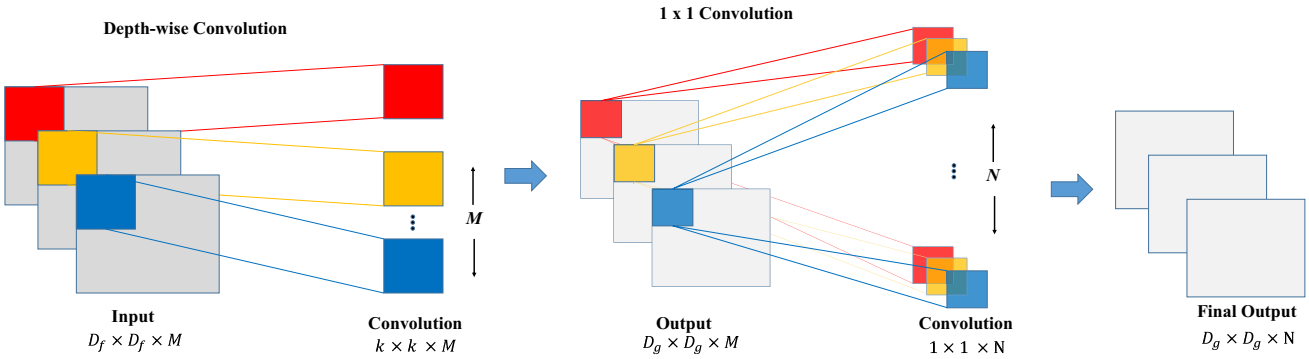

Fig. 1 Separable convolution process. Convolution depth-wise. Depth-wise and 1×1 convolution is involved

a conventional convolution, with the number of variables being only $\frac{1}{N} + \frac{1}{k^2}$ times that of a standard convolution. It is beneficial in contexts with scarce resources, like mobile devices, where memory and computation are restricted.

4 Proposed methodology

4.1 Smartphone runnable sign language recognition model (SRSLR)

4.1.1 Stage 1: feature extraction (FE)

A MediaPipe + PF model performs FE from the image and data preprocessing. MediaPipe could control distinct models for the components such as the face, hands, and pose by employing a region-appropriate image-pixel resolution for every single bit of data input from the camera. A Mediapipe system, created by Google, provides different pre-made components to form pipelines for Machine Learning (ML) and CV. A crucial part of this system is the Mediapipe hand identification and tracking algorithm, which uses ML and tracking methods to reliably and effectively detect and track hands in real-time video streams. Yet, HGR involves naming the specific hand gestures and detecting and tracking the hand. It is where PF and Mediapipe integration can be helpful.

PF is a BFT used to calculate a system's noisy size-based state. Based on the hand's present and previous positions, PF can be used in HGR to determine the likelihood that a particular hand gesture is presented. By incorporating PF with Mediapipe, it is possible to offer the first assessment of the hand location and follow its movement over time using Mediapipe's hand detection and tracking capabilities. PF is used to compute the probability that a particular hand gesture will be made based on changes in the location and movement of the hand over time. This method is effective when the hand is masked (or) the gesture is complex and challenging to recognize using a pure ML-based method. For HGR, Mediapipe and PF's combination deliver a more reliable and accurate solution.

This model integrates a PF with the Mediapipe hand detection for hand tracking. The PF helps to smooth out the hand tracking and keeps a constant reference point for HGR while the PF is updated, and the hand tracking accuracy is increased using Mediapipe landmarks.

Algorithm 2: Applying MediaPipe + PF for FE

Step 1	**First,** use the Mediapipe hand detection model to detect the hand and extract the hand landmarks.
Step 2	**Initialize** a PF with particles being hand positions in the image or video frame.
Step 3	**For Each** subsequent frame, use the Mediapipe hand detection model to detect the hand and update the hand landmarks.
Step 4	**Use** the PF to predict the hand's position in the current frame based on the positions of the particles in the previous frame.
Step 5	**Calculate** the likelihood of each particle given the updated hand landmarks.
Step 6	**Resample** the particles based on their likelihoods, with higher likelihood particles having a higher probability of being selected.
Step 7	**Repeat** Steps 3-6 for each successive frame, updating the PF and the hand landmarks.
Step 8:	**Use** the final set of particles to estimate the hand's position and recognize gestures based on the hand position.

4.1.2 Stage 2: proposed CNN and BILSTM-based SLR

Recognition model The CNN + BiLSTM comprises the HGR Model. The proposed model is described below and shown in Fig. 2.

Input Layer The model receives its input as a series of color frames from the smartphone's camera. The input frames are then resized after being normalized to have a zero mean and a unit variance.

Convolutional layer A standard convolutional layer with few filters is used for FE. The layer that comes after the batch normalization layer is the ReLU activation layer, and the layer that comes after that is the ReLU activation layer. Following the convolutional layer, a max-pooling layer reduces feature map spatial dimensionality.

Depthwise separable convolutional layer The computational cost and the number of parameters are reduced using two depthwise separate convolutional layers. It is necessary to perform a 1x1 convolution in order to combine the outputs of the depthwise convolution before performing a pointwise convolution. A depthwise convolution applies a filter to each input channel. Batch normalization and ReLU activation are both applied following each depthwise and pointwise convolution that is performed.

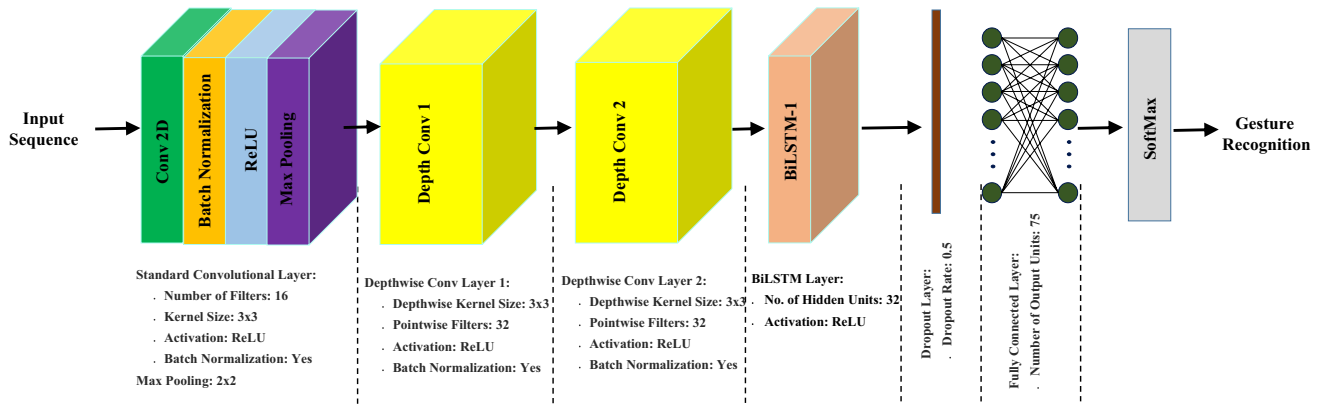

Fig. 2 The architecture of CNN + BiLSTM

BiLSTM layer One BiLSTM layer models SL gesture temporal dynamics. The BiLSTM layer receives the depthwise separable convolutional layer's feature maps. A dropout rate of 0.2 prevents overfitting in the BiLSTM layer, which has 16 or 32 hidden units.

Dropout layer A 0.5 dropout layer prevents overfitting.

FC layer A FC layer converts BiLSTM layer output to gesture classes. Gestural classes determine FC layer neuron count.

SoftMax layer Classification is achieved with a SoftMax layer. The SoftMax layer generates a gesture class probability distribution from the FC layer's output.

AOA optimizer with a 0.001 learning rate reduces each iteration's errors. The proposed model uses depthwise separable convolutions, a few layers, and filters to reduce computational costs and parameters. Dropout layers consist of to avoid overfitting. This approach can drastically reduce the model's computational cost and memory footprint, making it more dependable for smartphone use. In order to accomplish this, a more accessible model is used in order to optimize the model for the hardware found in smartphones. When stages 1 and 2 of the SRSLR model are integrated, the process flow looks like this:

Step 1: Data collection and preprocessing

(1) Collect hand gesture videos
(2) Preprocess the hand gesture data, such as cropping and resizing the images and normalizing and scaling the hand landmarks extracted using the Mediapipe framework. This step primarily confirms the data's integrity and accuracy and improves the model's performance during training and testing.
(3) Built-in data augmentation techniques extract keypoints and landmarks from a camera's input frames of the face, hands, and body.

Step 2: Hand tracking and FE

(1) Use the Mediapipe model to track the hand and extract the hand landmarks. The Mediapipe framework provides a robust and efficient method to track and extract hand landmarks from video or image data.
(2) Apply PF to the hand landmarks to improve tracking accuracy. PFis is a statistical method for tracking objects in video or image data.
(3) It can improve the tracking accuracy of the hand landmarks, uniquely in cases where the hand is occluded or partially visible.

Step 3: Data cleaning and labelling

(1) Following this step, the hand landmark points for each frame are flattened, then concatenated, and finally saved to a file. This step is essential if you want to get rid of null entries. A failed feature detection occurs when a fuzzy image is sent to the detector, resulting in an invalid dataset entry. Failing feature detection is avoided by cleaning the data.
(2) Each class's frame sequences and labels are saved for the subsequent training, testing, and validation phase.
(3) Preprocess the hand landmarks for input to the CNN + BiLSTM model, such as reshaping them into a tensor of proper dimensions.
(4) The CNN + BiLSTM model requires a 3-D tensor with the first dimension pointing to the number of instances, the second corresponding to the image's dimension, and the third showing the height of the image.

Step 4: CNN + BiLSTM training and testing

(1) Build a CNN + BiLSTM model consisting of a Depthwise convolutional and BiLSTM layer.
(2) The CNN layer manages FE, while the BiLSTM layer allows the model to learn the temporal dependencies between hand gestures. Train the CNN + BiLSTM model on the preprocessed hand gesture data.
(3) The weights and biases of the model have been optimized so that the loss function is reduced to its smallest value. Validate the trained model on a validation set to evaluate its accuracy and fine-tune it as needed.
(4) In order to prevent overfitting and keep an eye on how well the model is doing while it is being trained, the validation set is used.

Step 5: Evaluation and real-time recognition

(1) Evaluate the final hand GRM on a test set to measure its accuracy. The test set assesses the model's performance on unseen data.
(2) Use the final model for real-time recognition of hand gestures in video or image data.
(3) The model is implemented in real-time applications such as gesture-based interfaces or SL recognition.

5 Experimental analysis

5.1 Description of dataset

INCLUDE-50: Dataset for the Indian Lexicon in SL. It has frames of 0.27 million with 263 classes from 15 distinct word categories, making it the first publicly accessible ISL dataset that performs well compared to datasets on other SL. By getting aid from the deaf community, the recording of the videos INCLUDED is performed. The background, resolution, and lighting of the movie were all chosen to resemble

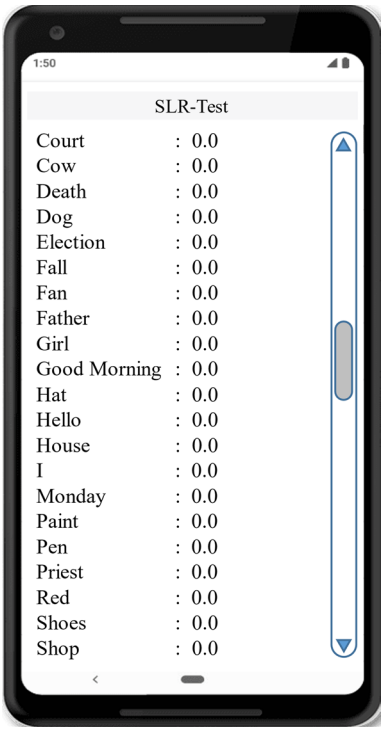

Fig. 3 HAR app's screenshot depicting each activity class's probability

real-world situations closely. INCLUDE-50, a smaller subset that has 50 classes, is selected to facilitate the Deep Learning (DL) model quick assessment of the dataset. The following classes have been selected: {Bank, Bird, Black, Boy, Brother, Car, Cell Phone, Court, Cow, Death, Dog, Election, Fall, Fan, Father, Girl, Hello, Hat, Good Morning, House, I, Monday, Paint, Pen, Priest, Red, Shoes, Shop, Summer, T-Shirt, Teacher, Thank You, Time, White, Window, Year, Large, Dry, Good, Happy, Hot, It, Long, Loud, New, Quiet, Short, Small, Train Ticket, You (plural)}. The classes covering all 15 were selected based on how frequently they were applied. In INCLUDE-50, there are 958 videos and 60,897 frames in total. Training and testing are the two classes of datasets with ratios of 0.8 and 0.2, respectively. 5% of these samples were randomly selected for validation at each epoch. The test dataset, which includes 20% of unseen samples, was then used to evaluate the trained model.

5.2 Experimental setting

Without Graphics Processing Unit (GPU) acceleration, the experiment was evaluated on a MacBook Pro with 16 GB of RAM and a 2.2 GHz Intel i7 processor. A 1.9 MB protocol buffer *(.pb)* in the pre-trained prototype model format was implemented. Combining operations, compressing the FlatBuffer format, and generating a TensorFlow Lite *(tflite)* model with high-speed efficiency and more memory for conducting inference on edge devices, the TensorFlow Lite Python converter was applied.

On a Personal Computer (PC), the Python interpreter Application Program Interface (API) was used to evaluate the TensorFlow Lite model, and the Java API was incorporated into an Android application. The Android SLR app was designed in Android Studio and evaluated on a smartphone Google Pixel 2 (API 30) model using the Android Emulator. It suits well on at least an API version 24-based Android smartphones. Every 2.56 *Sec.,* for each activity class, the output predictions are updated by the application. It runs on a single-thread CPU and does not use GPU (or) Android Neural Networks API (NNAPI) acceleration (Fig. 3). The

Table 2 Parameters used to train the proposed model

Parameter	Value
Learning rate	0.001
Dropout rate (BiLSTM)	0.2
Number of hidden units (BiLSTM)	32
Number of particles (PF)	100
Filters per convolutional layer	16, 32, 64
Hand landmarks	21
Body pose landmarks	33
Depthwise convolution kernel size	3×3 for depth-wise, 1×1 for point-wise
Pointwise convolution filters	Varies
Optimizer	AOA

Table 3 Comparing the original model with the changes

Model 1	MP with CNN + BiLSTM (proposed model without PF)
Model 2	PoseNet with CNN + BiLSTM
Model 3	OpenPose with CNN + BiLSTM
Model 4	MP + PF with CNN + LSTM
Model 5	MP + PF with CNN + GRU
Model 6	MP + PF with CNN + BiGRU

Table 4 Comparison of MAE. MSE and R^2

Network	MAE	MSE	R^2
Proposed Model	0.21	1.26	0.93
Model 1	0.22	1.31	0.91
Model 2	0.88	6.20	0.42
Model 3	0.83	5.24	0.58
Model 4	0.74	4.85	0.61
Model 5	0.43	1.35	0.85
Model 6	0.39	2.45	0.81

Table 5 Comparison of accuracy, recall, precision, and F1-score

Network	Accuracy	Recall	Precision	F1-Score
Proposed model	0.9744	0.9593	0.9940	0.9763
Model 1	0.9098	0.8947	0.9107	0.9027
Model 2	0.8043	0.7750	0.7750	0.7750
Model 3	0.8241	0.7660	0.8182	0.7912
Model 4	0.8713	0.7963	0.9556	0.8687
Model 5	0.8807	0.8750	0.8571	0.8660
Model 6	0.9010	0.8846	0.9200	0.9020

program that consumes power consumption in its light, medium, and superior levels was tracked using the Android Studio profiler. The proposed model was trained using the parameters as shown in Table 2.

This work compares the model with the modified models provided in Table 3 to analyze the proposed architecture, viz., Mediapipe with PF (MP) and CNN + BiLSTM's (MP + PF with CNN + BiLSTM) efficiency. A comparison of several SLR architectures is shown in Table 4, and the proposed architecture, MP + PF with CNN + BiLSTM, is assessed for its efficiency. Three evaluation metrics—Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2) values—are used to compare the results. They are compared based on how well the models perform on these metrics. According to Table 5, the proposed MP + PF with CNN + BiLSTM model surpasses all the other models with a 0.21 MAE, 1.26 MSE, and 0.93 R^2 value.

(1) *Model 1* MP with CNN + BiLSTM (Proposed without PF), the proposed model without PF, performs similarly with 0.22 MAE, 1.31 MSE, and 0.91 R^2 value.
(2) *Model 2* (PoseNet with CNN + BiLSTM) performance is much lower than Model 1, with 0.88 MAE, 6.20 MSE, and 0.42 R^2 value. Model 2 deploys a PoseNet with CNN + BiLSTM.
(3) *Model 3* (OpenPose with CNN + BiLSTM), which combines CNN + BiLSTM and OpenPose, performs lower than the other two models, with 0.83 MAE, 5.24 MSE, and 0.58 R2. Performance levels vary between Models 4, 5, and 6, combining CNN and MP + PF with different RNN models (LSTM, GRU, BiGRU).
(4) *Model 4* (MP + PF with CNN + LSTM), the MAE, MSE, and R^2 Model 4 with CNN + LSTM values are 0.74, 4.85, and 0.61.
(5) *Model 5* (MP + PF with CNN + GRU), the MAE, MSE, and R^2 values with CNN + GRU are 0.43, 1.35, and 0.85.
(6) *Model 6* (MP + PF with CNN + BiGRU), the MAE, MSE, and R2 values with CNN + BiGRU are 0.39, 2.45, and 0.81.

According to the results, the MP + PF with CNN + BiLSTM proposed architecture is the most effective in terms of SLR performance, with the lowest MAE and MSE values and the highest R^2 value. Comparing other models shows that combining the CNN + BiLSTM model and PF may significantly improve the SLR's performance.

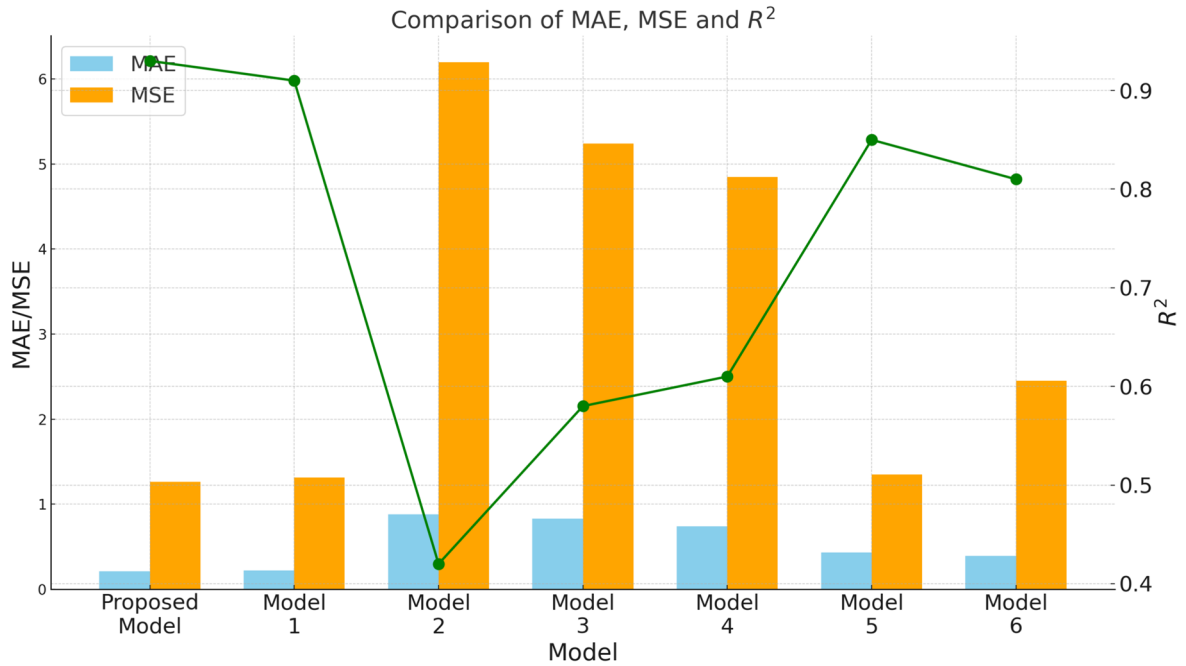

Fig. 4 Graphical Comparison of MAE. MSE and R^2

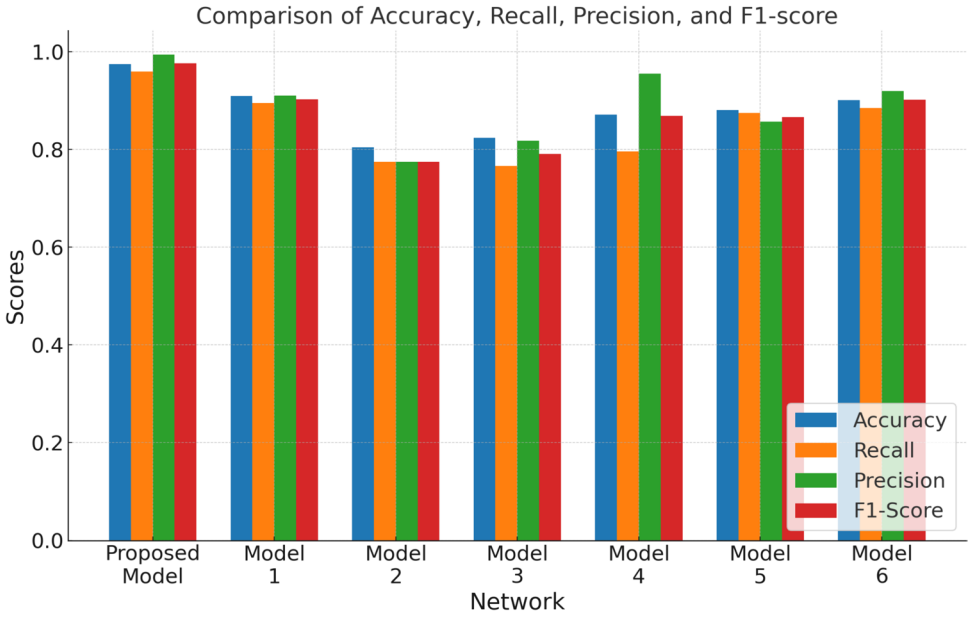

Fig. 5 Graphical comparison of accuracy, recall, precision, and F1-score

The proposed model's comparison for evaluation metrics includes accuracy, recall, precision, and F1-score, shown in Table 5. These measures are employed to assess the model's effectiveness. The proportion of successfully shown gestures is how accuracy is found. The proportion of predicted positive cases is valued by precision. When the dataset is imbalanced, the F1-score, which blends accuracy and recall to provide a fair assessment of the model's performance, is especially helpful (Fig. 4).

With an accuracy score of 0.9744, the proposed model (MP + CNN + BiLSTM with PF) could correctly accept significant gestures in the test dataset. The model also reached are high recall of 0.9593, indicating that a significant ratio of the actual positive cases is correctly identified (*i.e.,* correctly predicted the actual poses in the test dataset). The proposed model's precision, which was high at 0.9940, demonstrated its ability to limit False Positives (FP) and successfully identify most of the predicted positive cases (*i.e.,* correctly predicted the gestures present in the test dataset). The proposed model's F1-score of 0.9763 shows it successfully balanced precision and recall (Fig. 5).

Compared to the recommended model with PF, Model 1 (the proposed model without PF) has lower accuracy, recall, and F1 score. It shows that PF helped to make this model perform better.

Models 2 and 3 (PoseNet with CNN + BiLSTM and OpenPose with CNN + BiLSTM) performed lower than the proposed model with PF related to accuracy, recall, and F1-score. It implies that the proposed model, which encompasses Mediapipe and PF, outperforms the two models mentioned before.

When compared to Model (1, 2, 3) and Model (4, 5, 6) (MP + PF with CNN + LSTM, MP + PF with CNN + GRU, and MP + PF with CNN + BiGRU) had higher accuracy and recall but lower precision. It implies that, compared to the proposed model, these models tended to create more FP. The findings indicate that among the models assessed, the proposed model (MP + PF with CNN + BiLSTM) performed the best, obtaining high accuracy, recall, precision, and F1-score.

This work compared the proposed model's results with various existing advanced models, such as MediaPipe Optimized Gated Recurrent Unit (MOPGRU) and Pose-based Temporal Graph Convolutional Network (Pose-TGCN), in order to assess the effectiveness and performance of the proposed model. This work trained the model with the identical parameters as before and outperformed the current method regarding recognition accuracy. During training, with fewer activation functions and parameters, our proposed model performed similarly to other models used in the experiments of convergence speed and learning efficiency. Figure 6 displays the models' overall recognition accuracy.

6 Conclusion and future work

In conclusion, the development of a smartphone-runnable Sign Language Recognition Model (SLRM) marks a significant stride towards dismantling communication barriers for deaf and hard-of-hearing individuals. This model ingeniously integrates key components—Mediapipe hand

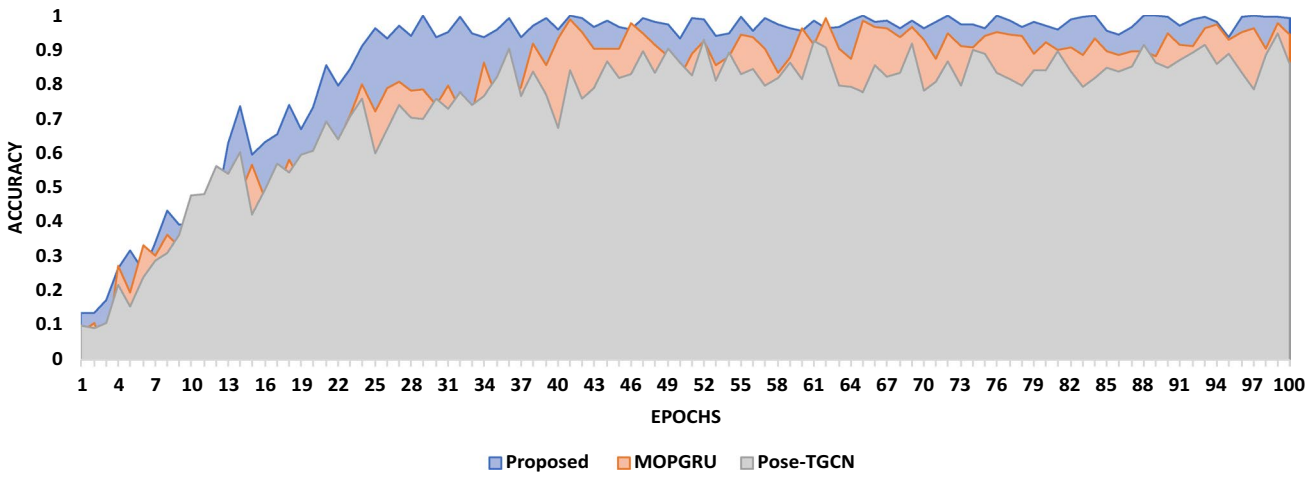

Fig. 6 Comparison of the accuracy of advanced models

tracking, Particle Filtering (PF), Convolutional Neural Network (CNN), and Bidirectional Long Short-Term Memory (BiLSTM)—to deliver a robust and accurate Hand Gesture Recognition (HGR) system. By capturing the temporal dynamics of Sign Language (SL) gestures, it adeptly accommodates a wide array of signs and adapts to variations in signing speed and style. Furthermore, the model's architecture, characterized by a minimalistic approach in layer and filter usage, depthwise separable convolutions, and dropout layers, significantly reduces computational demands, aligning perfectly with smartphone deployment criteria. Reiterating its core contributions, this model not only exemplifies superior performance in convergence speed and learning efficiency but also showcases a promising pathway towards real-time SL translation tools. These tools hold the potential to seamlessly translate SL into spoken or written language, fostering inclusivity and understanding within society.

Despite facing challenges such as environmental sensitivity and device performance variability, future research is geared towards enhancing the model's robustness, extending its gesture recognition capabilities, and embedding privacy-preserving features. The collective efforts in this work lay a foundational stone for enabling effective communication with the deaf and hard-of-hearing community, driving us closer to a more accessible world.

Funding Not Applicable.

Declarations

Conflict of interest Not applicable.

References

Al-Sarayrah W, Al-Aiad A, Habes M, Elareshi M, and Salloum SA (2021) Improving the deaf and hard of hearing internet accessibility: JSL, text-into-sign language translator for Arabic. In: Advanced machine learning technologies and applications: proceedings of AMLTA Springer International Publishing, pp 456–468

Bantupalli K, and Xie Y (2018) American sign language recognition using deep learning and computer vision. In: IEEE international conference on big data (big data), pp 4896–4899

Bragg D, Koller O, Bellard M, Berke L, Boudreault P, Braffort A, and Ringel Morris M (2019) Sign language recognition, generation, and translation: an interdisciplinary perspective. In: Proceedings of the 21st international ACM SIGACCESS conference on computers and accessibility, pp 16–31

David D, Alamoodi AH, Albahri OS, Zaidan BB, Zaidan AA, Garfan S, Malik RQ (2023) Landscape of sign language research based on smartphone apps: coherent literature analysis, motivations, open challenges, recommendations, and future directions for app assessment. Univ Access Inf Soc. https://doi.org/10.1007/s10209-022-00966-9

Deng J, Dong W, Socher R, Li LJ, Li K, and Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: Computer vision and pattern recognition, CVPR. IEEE conference, pp 248–255

Donahue J, Anne Hendricks L, Guadarrama S, Rohrbach M, Venugopalan S, Saenko K, and Darrell T (2015) Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2625–2634

Girshick RB, Donahue J, Darrell T, and Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Computer vision and pattern recognition, pp 580–587

Godwin-Jones R (2017) Smartphones and language learning

Karpathy AG, Toderici S, Shetty T, Leung SR, and Fei-Fei L (2014) Large-scale video classification with convolutional neural networks. In Computer vision and pattern recognition, pp 1725–1732

Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90

Li D, Rodriguez C, Yu X, and Li H (2020) Word-level deep sign language recognition from video: a new large-scale dataset and

methods comparison. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision, pp 1459–1469
Lionel P, Sander D, Pieter-Jan K, and Benjamin S (2015) Sign language recognition using convolutional neural networks. In: Computer vision-ECCV 2014 workshops, Lourdes Agapito, Michael M. Bronstein, and Carsten Rother (Eds.). Springer International Publishing, Cham, pp 572–578
Lionel P, Mieke Van H, and Joni D (2017) Gesture and sign language recognition with temporal residual networks. In: IEEE international conference on computer vision workshops, Venice, pp 3086–3093
Longlong J, Elahe V, Matt H, and Yingli T (2019) Recognizing american sign language manual signs from RGB-D videos. CoRR abs/1906.02851
Mercurio G (2021) LIS2SPEECH LIS translation in written text and spoken language (Doctoral dissertation, Politecnico di Torino)
Oscar K, Hermann N, and Richard B (2016) Deep hand: how to train a CNN on 1 million hand images when your data is continuous and weakly labelled. In: IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA, pp 3793–3802
Panning S, Lee RL, Misurelli SM (2021) Breaking down communication barriers: assessing the need for audiologists to have access to clinically relevant sign language. J Am Acad Audiol 32(4):261–267
Saleh B, Al-Beshr R, Tariq UM (2020) D-talk: sign language recognition system for people with disability using machine learning and image processing. Int J Adv Trends Comput Sci Eng 9:4374–4382
Simonyan K, and Zisserman A (2014) Two-stream convolutional networks for action recognition in videos. In: Advances in neural information processing systems, pp 568–576
Subramanian B, Olimov B, Naik SM, Kim S, Park KH, Kim J (2022) An integrated media pipe optimized GRU model for Indian sign language recognition. Sci Rep 12(1):1–16
Tao H, Hua M, Zhang Y (2017) Moving object recognition using multiview three-dimensional convolutional neural networks. Neural Comput Appl 28(12):3827–3835
Wang L, Xiong Y, Wang Z, Qiao Y, Lin D, Tang X, and Van Gool L (2016) Temporal segment networks: towards good practices for deep action recognition. In: European conference on computer vision, Springer, pp 20–36