Received: 30 August 2025 | Revised: 11 February 2026 | Accepted: 11 June 2026 | Published online: 1 July 2026

RESEARCH ARTICLE

Artificial Intelligence and Applications
2026, Vol. 00(00) 1–15
DOI: 10.47852/bonviewAIA62027479

BON VIEW PUBLISHING

Dynamic Hand Gesture Recognition for Human–Computer Interaction

Chim Tun Sian[1], Suhardi Azliy Junoh[1,*] and Mohd Shahrimie Mohd Asaari[1]

[1] *School of Electrical and Electronic Engineering, Universiti Sains Malaysia, Malaysia*

Abstract: Hand gesture recognition (HGR) is a crucial technology for human–computer interaction, which allows users to control devices such as a smart TV by using only simple hand gestures without contacting them. However, getting real-time performance on resource-constrained devices remains a big hurdle. We introduce a keypoint-based dynamic HGR approach that leverages fixed-length sequences of 30 frames to obtain three-dimensional hand landmarks using MediaPipe Holistic. Data augmentation and standardization were used to make the model robust, and the dataset contained 10 classes of dynamic gestures. Several models, such as long short-term memory, gated recurrent unit (GRU), and one-dimensional convolutional neural network hybrids, both with and without attention mechanisms, were proposed, trained, and evaluated. Through experiments, it is discovered that GRU with attention has an optimal trade-off between classification accuracy and inference delay. To facilitate deployment onto embedded platforms, post-training quantization has been used to reduce the model size and accelerate inference. The optimized model is integrated with an Android TV application through a socket connection that provides real-time gesture-to-command mapping for navigation, selection, and volume control; evaluations of the optimized model using the emulator show that it is functional and has consistent performance characteristics, while the use of pointers does not entail a significant delay because of the small communication overhead. The results indicate that keypoint-based temporal modeling is a promising approach for real-time gesture-driven interaction with embedded systems in controlled deployment scenarios.

Keywords: human–computer interaction, gesture recognition, computer vision, deep learning, artificial intelligence

1. Introduction

Hand gesture recognition (HGR) has become an exciting field for natural and intuitive human–computer interaction (HCI) and in vision-based systems for touchless control in various applications. Static gestures are gestures with a definite pose of the hand in a single spatial moment, while dynamic hand gestures are collections of continuous hand movements in time, which provide deeper and more expressive means of interaction [1, 2]. However, it was pointed out that the temporal patterns are more similar to the real human communicative activities, making the dynamic hand gestures more suitable for real-world use in smart settings [3], games [4], healthcare [5], or consumer electronics [6].

The dynamic HGR pipeline consists of several vital components, namely, hand identification and tracking, gesture acquisition, feature extraction, and classification. In the past, researchers have developed vision-based and non-vision-based systems, which include camera-based systems based on computer vision and sensor-driven systems that use data gloves. Of these, vision-based solutions have gained wider application as they are nonintrusive and readily integrated. High-fidelity, real-time tracking of hand landmarks is provided by frameworks such as Media Pipe Holistic, which can detect 21 three-dimensional (3D) (x, y, z) keypoints per hand that enable multi-hand recognition and handedness classification [7].

The performance of dynamic HGRs has been greatly enhanced by deep learning (DL), specifically using a temporal model like a long short-term memory (LSTM) network that is able to capture the sequential dependence of the gesture execution. To represent spatial-temporal information, LSTMs are often integrated with a convolutional network, such as 3D-convolutional neural network (CNN), enhancing the accuracy. However, such models are very complex and difficult to use in real time on devices with limited resources, particularly in cases where the device is a low-resource computer [8]. A few lightweight architectures have been proposed to reduce the overhead of memory usage and inference latency while preserving good classification results, such as gated recurrent units (GRUs) and one-dimensional (1D)-CNNs [9].

Despite the significant advances made over the past few years in gesture-based HCI systems, there are some major challenges with these systems. Many solutions do not allow the user to express themselves as they can only interact with the system using basic or simple motions. Also, due to the absence of temporal modeling of these systems, they are not able to recognize gesture sequences that are essential to implement complex orders. Many problems are due to the misclassification of similar movements to each other, for example, the forward and backward rotations (14.0% and 12.1% error, respectively) or clockwise and

***Corresponding author:** Suhardi Azliy Junoh, School of Electrical and Electronic Engineering, Universiti Sains Malaysia, Malaysia. Email: suhardi@usm.my

© The Author(s) 2026. Published by BON VIEW PUBLISHING PTE. LTD. This is an open access article under the CC BY License (https://creativecommons.org/licenses/by/4.0/).

anticlockwise rotations (up to 19.1% error) [10]. The accuracy of identification is improved from 82.2% to 91% with an increasing number of input frames from 5 to 25. However, progress halts at this point, which shows how important it is to have more reliable and effective temporal feature modeling.

Another important limitation is the response time of the system. The perceptual threshold of human is that the system response should not be delayed by more than 100 ms after the gesture is completed; otherwise, it prevents the perception of real-time interaction [11]. Since limited CPU resources aggravate inference delays in DL models, latency issues must be studied for embedded or edge devices like Android televisions (TVs). The researchers must make an important compromise between making a model complicated and making it perform with low latency for efficient gesture-controlled interfaces. In order to solve these problems, the researchers of this study have investigated and assessed the dynamic HGR technique with the temporal DL model, LSTM and GRU, along with landmark extraction from the Media Pipe framework. The main purposes of this study were to increase the accuracy of the classification of standard dynamic gestures with an efficient temporal modeling process, to reduce the latency of inferences in real time for the CPU-only devices, and to test the usability of the optimized model on the Android TV platform to confirm its applicability to the actual HCI situations.

The study is based on vision-based dynamic HGR for single-handed gestures like swipes, pinches, and pointing gestures performed within a 1–3 m distance and under normal indoor lighting conditions using a 720p resolution camera. The studies that were considered did not include the RGB-based recognition methods, full-body motion analysis, and facial gestures. In this study, the designed system architecture guarantees the accurate and economical computational performance while keeping practically deployable on embedded devices without GPU acceleration.

Also, this study highlights important electronic engineering skills, such as machine learning model creation, embedded system optimization, and real-time video signal processing. Android TV deployment is done to focus on the interaction between hardware and software and low-power processing. However, gesture recognition demands effective use of filtering and feature extraction techniques. Furthermore, the application of LSTM and GRU models underscores the interdisciplinary character of contemporary methods in electrical engineering, emphasizing its vital role in algorithm development and modeling of control systems.

2. Related Works

Dynamic HGR has emerged as an essential component to improving HCI, thus creating more natural and intuitive interfaces. Early research on gesture-based interaction focused on static gestures [12, 13], but dynamic hand gestures have led to a number of problems in temporal modeling and in defining and segmenting gesture sequences in addition to latency issues. The nonintrusive feature of vision-based methods and the increased accuracy of depth cameras and real-time keypoint tracking systems have increased the importance of these methods. These systems can accurately capture the spatial and temporal characteristics of hand motion, for example, in virtual and augmented reality applications [14, 15].

Generally, gesture-based HCI systems make the distinction between static gestures and dynamic gestures [16]. Simple gestures are easily recognized and are called static gestures when the hand is in a fixed position. They are less expressive and flexible than dynamic gestures, however, which are constantly changing and of more semantic value. However, there are significant challenges in segmenting dynamic gestures, making real-time inferences, and ensuring the system doesn't tire out. People prefer static gestures for discrete tasks and favor dynamic gestures for complex or continuous interactions, as can be seen from studies done in virtual environments [17, 18].

The key components of any HGR system are data acquisition, data preprocessing, feature extraction, and classification. To achieve high-accuracy systems [19], recent progresses in computer vision technology, especially keypoint estimation methods like OpenPose [20–23] or AlphaPose [24–27] and MMPose [28–31] and Detectron2 [32–35], have been made. Of these, MediaPipe has become a lightweight and efficient 3D hand landmark-detection framework for real-time use. Comparative tests show that the detector is always more reliable and precise than other detectors, especially in the tracking of 3D keypoints in different situations [20].

In the area of classification, classic machine learning techniques like K-nearest neighbors (KNN), support vector machines (SVM), random forest, and logistic regression have proved successful for small-sized, well-curated datasets [21]. Since the advent of DL, however, DL models like CNNs, LSTM networks, GRUs, and transformers have been the backbone of powerful gesture recognition systems [22]. CNNs have been shown to be effective for spatial feature extraction in static gesture recognition [23], and LSTMs and GRUs are good for capturing temporal dependencies that exist in dynamic gestures [24].

To tackle the challenges of temporal modeling, some studies have suggested the use of hybrid models combining CNNs with LSTMs [25]. The frame sequences are passed to CNNs to capture spatial characteristics that are then given to LSTMs for the temporal evolution of gestures. CNNs and LSTMs have been used together to achieve high accuracy in real-time dynamic gesture recognition, leveraging both spatial and temporal information provided by frameworks like MediaPipe Holistic [26]. This comprehensive technique enables uniform performance with different gesture lengths and different techniques.

There have been some transformer-based models that have been explored for gesture recognition, including the use of vision transformer architectures [27, 28]. The models use self-attention mechanisms to capture long-range dependencies and complex spatial interactions efficiently. Robustness of transformers comes at a high cost: in order to avoid overfitting and to assure generalization, large-scale annotated datasets are necessary [29]. Recent changes to masked autoencoders for skeleton-based gesture identification add another layer of performance improvement, adding frequency domain transformations. This will enhance spatial sensitivity for gesture interpretation [30].

The GRUs have been popular for real-time applications due to their simpler structure and faster convergence than LSTMs [31]. Some studies find that GRUs can match or outperform LSTMs with a lower computational cost. They are particularly attractive models to be deployed in embedded systems, where both latency and efficiency are critical [32, 33]. The advantages and disadvantages of each model are presented in Table 1.

Many new studies have shown the application of these ideas in actual situations. Ahmad et al. [2] explored the application of LSTM networks with MediaPipe Hand landmarks and achieved a test accuracy of 99.71% on 27 dynamic gesture classes, demonstrating robustness and efficiency in training and inference. Constraints like class imbalance and dataset size were found, however, and they negatively affected gesture recognition for underrepresented gestures. Various sensor-based methods have been studied, such as 1D-CNN-ResBiGRU architectures that are applied to accelerometer/gyro-scope data from wearable sensors.

Table 1
A summary of the trade-offs between the different DL models

Model	Strengths	Weaknesses	Best for
CNN	Excellent at extracting spatial features; robust to displacement and distortion	Limited in modeling long-range temporal dependencies	Static gesture recognition
LSTM	Learns long-term temporal dependencies; effective for sequential gesture modeling	High computational cost	High-accuracy applications
GRU	Faster training than LSTM; fewer parameters	Less memory for long sequences	Real time systems
Transformer	Strong at learning global dependencies; handles complex relationships with self-attention	Requires large dataset and computation	Complex gesture relationships across long ranges

These systems reached an accuracy of 98.49% for benchmark datasets, outperforming traditional systems. But lack of variety in gestures and limited edge deployability remain the challenges to scalability [34].

Another research proposed an improved LSTM system combined with MediaPipe Holistic with the inclusion of a reset mode to remove the transitional noise among the gestures. The system's word-level recognition accuracy was 99.4%, and it was able to perform real-time training, but the sentence-level recognition accuracy was affected by the temporal transition error and hand posture variation [35].

However, gesture recognition technologies of today suffer from a number of drawbacks. This encompasses co-articulation, epenthesis, and the ability to detect the end of a gesture in a fluent performance and to identify a signer irrespective of his or her gestures [36]. Besides, several models with latency and memory overhead, especially in real-time inference scenarios, have been proposed in the past. As the complexity grows, high complexity models might yield better accuracy in certain situations; however, they are not suitable for time-sensitive applications because of increased processing delay [37].

In recent years, research efforts have been directed toward solutions for these problems through lightweight and efficient structures. For example, MS-MobileNet, a mobile-friendly CNN variant, has been proposed that reduces the inference time to 0.115 s per gesture, meeting the requirement for real-time responses [37]. Moreover, data augmentation methods such as simulated camera angle variation, finger length alteration, and keypoint dropout have been extensively utilized to improve model generalization and resilience in scenarios with low training data [38, 39].

The recent progress in dynamic HGR demonstrates their trend toward spatiotemporal models that are able to leverage keypoint information to learn complex gesture structures. MediaPipe can extract keypoints efficiently in real time, but to get high accuracy for identification, the temporal models like LSTM, GRU, and transformers are required. The study highlights the significance of lightweight models for the latency vs performance trade-off and also the critical role of lightweight models in data augmentation and data shortage. However, challenges like gesture ambiguities, constraints on datasets, and real-time systems pose a need to investigate this area further.

3. Research Methodology

3.1. Research design

The entire process of the proposed dynamic HGR system is shown in Algorithm 1. The training data for each gesture class is aggregated in Step 1. In Step 2, data preprocessing involves tasks such as data labeling, data normalization, and splitting the data into a testing set and training set.

The researchers created a DL neural network that was trained to classify the dynamic hand gestures. The trained model was then tested by assessing the model's performance, which includes accuracy, loss, F1-score, and prediction delay. If the model failed to satisfy the pre-determined deployment requirements, data normalization and data augmentation were done, and the model was retrained and re-evaluated. They made use of iterative hyperparameter tuning to further enhance the performance of their models.

After meeting the required parameters, they did real-time testing to ensure the visual accuracy of the system. If the evaluation is not satisfactory, the workflow changes with different model architectures and retraining to obtain satisfactory results.

The finalized model is then added to the Android TV platform. The system is validated by doing gestures in real time in front of the camera, and every gesture is mapped into a TV control command. The iterative methodology ensures stable and seamless interaction between the user and Android TV, with hands-on movement from both sides.

3.2. Data collection and data preprocessing

The nine dynamic hand gestures chosen were *swipe left, swipe right, swipe up, swipe down, pinch, pointer, scissors, open palm,* and *close palm*. An extra "*no gesture*" class was added to cover idle poses and invalid hand gestures. Table 2 provides a summary of the descriptions of each gesture, and Figure 1 provides visual examples of the gesture set. Table 2 and Figure 1 give a complete representation of the dynamic hand gestures that were utilized in this study.

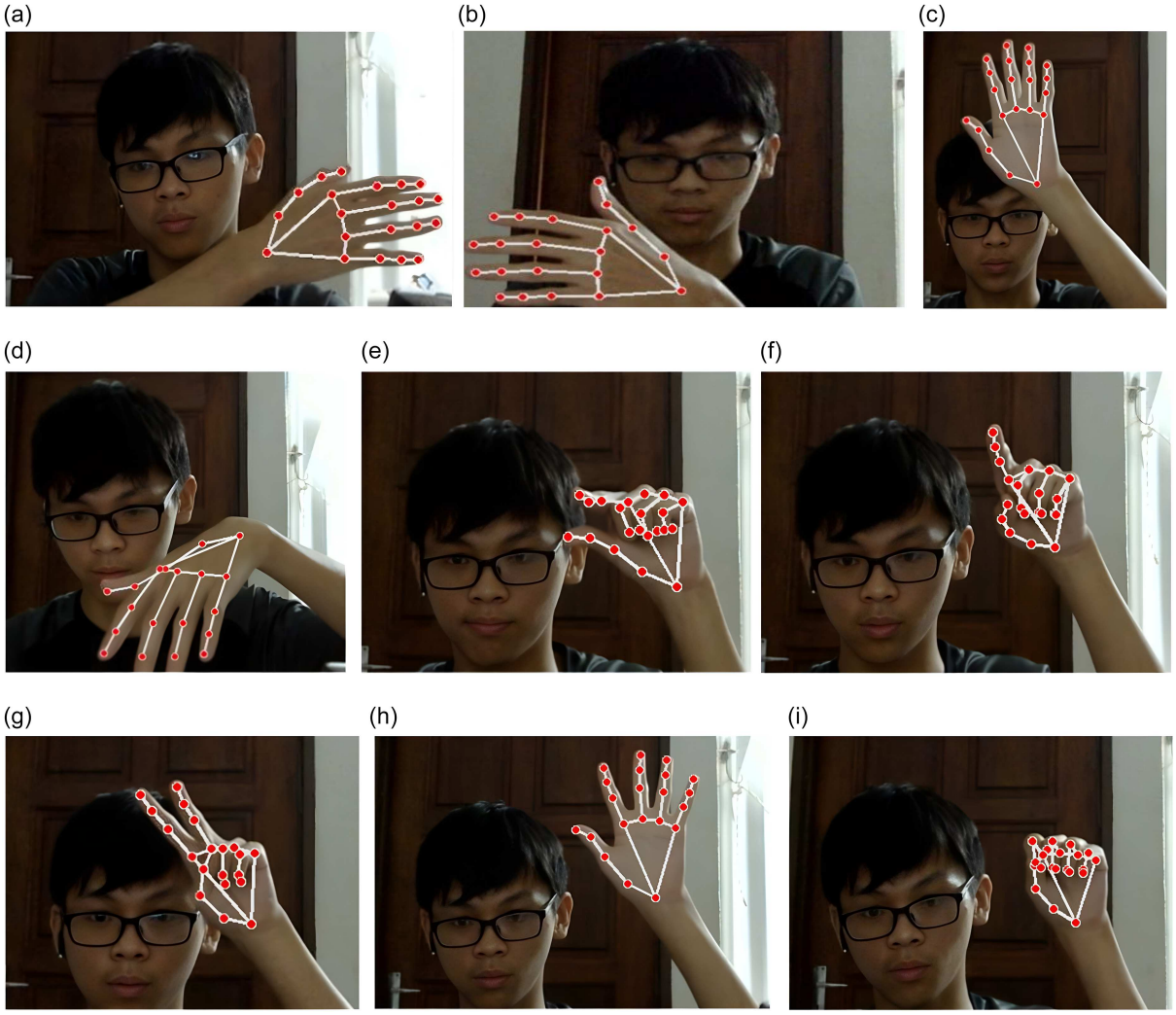
A Logitech C270 webcam was used to record gesture sequences, and the MediaPipe Holistic model was used to extract 21 hand landmarks. Specifically, light-brown to light-yellowish skin tone people were asked for the information using a Logitech C270 webcam. Also, the researchers used various background conditions and hand orientations to collect data for the added positional and environmental heterogeneity. The participants did various gestural movements at different angles and distances from the camera to better model the gestural movements in real-life situations.

Each hand contributed 63 3D keypoints, resulting in 126 features for both hands. The data for each gesture were collected 100 times, sampling 30 frames of the landmark coordinates for each sample. The researchers collected the data using different hand

Table 2
A summary of the gestures and their descriptions

Gesture	Description of movement
Swipe left	Hand moves horizontally from right to left
Swipe right	Hand moves horizontally from left to right
Swipe up	Hand moves vertically upward
Swipe down	Hand moves vertically downward
Pinch	Index fingertip and thumb tip move together in a closing motion
Pointer	Index finger extended, remaining fingers folded
Scissors	Index and middle fingers extended in a V-shape
Open palm	Fully open hand with fingers spread Close palm Hand forms a fist
No gesture	Idle or unrelated hand pose

Figure 1
The dynamic hand gesture dataset used in this study, namely, (a) *swipe left*, (b) *swipe right*, (c) *swipe up*, (d) *swipe down*, (e) *pinch*, (f) *pointer*, (g) *scissors*, (h) *open palm*, and (i) *close palm*



orientations shown by different users to assess the robustness of the algorithm.

For this research study, the researchers chose to focus on single-handed interactions for the gesture set, for stable real-time performance, and for feasible deployment on Android TV devices with low resources. The use of two-hand gestures raises new issues, like increased inter-hand occlusion, higher synchronization complexity, and high processing needs, which can severely affect the real-time performance of constrained systems.

Two normalization procedures were used to improve generalization. The first method is the translation-invariant method, which translates landmarks to the wrist joint, thus minimizing the positional dependency. Additionally, the keypoint coordinates were converted to unit direction vectors (between adjacent joints)

using the vectorization method to be invariant to translation, rotation, and scale.

Then, data augmentation techniques were used to add some natural temporal and spatial pattern changes. Some of these techniques included spatial rotation, temporal masking, Gaussian noise injection, speed perturbation, and vector space transformations. All these enhancements resulted in an increased robustness of the model to inter-user variability, occlusions, and complex backgrounds. The data augmentation was done during the training of the model to artificially increase the diversity and the number of the training datasets. This process also enhanced the generalization and performance of the model when the researchers changed the main data, as it did not allow the model to learn the patterns of the training set. Multiple exposures of the model lead to the addition of annotation variability in order to make the model more resilient in real-world scenarios with complex backgrounds, user variations, and real-world imperfections. The end goal of data augmentation is to enhance the accuracy and reliability of the model in real-world scenarios.

3.2.1. Temporal warping

This method is based on the idea that the sequence is occasionally played at a random speed (0.9–1.1)×. Bilinear interpolation is used to warp the timeline.

Shorter sequences are zero-padded, while longer sequences are truncated to ensure a fixed-length representation. This is a way of mimicking natural variations in the speed of motion and robustness to temporal inconsistencies in dynamic actions. The sample data for Table 3 were interpolated in order to speed them up, and the empty timesteps were padded with zeros, keeping the 30-frame dimension.

Table 3
The temporal masking used to augment the sample data

Description	Time step 1	Time step 2	Time step 3	Time step 4
Time step	1	2	3	4
Sample input data	[0.1, 0.2, 0.3]	[0.4, 0.5, 0.6]	[0.7, 0.8, 0.9]	[1.0, 1.1, 1.2]
Augmented output data	[0.4, 0.5, 0.6]	[0.7, 0.8, 0.9]	[0.0, 0.0, 0.0]	[0.0, 0.0, 0.0]

3.2.2. Jitter (additive noise)

All of the keypoint coordinates are corrupted by adding some Gaussian noise, the level of which is sampled randomly from the range $\sigma \in [0.01, 0.03]$. This augmentation emulates sensor noise as well as small anatomical irregularities.

This augmentation introduces variations that prevent overfitting to exact coordinate values and encourage the model to learn robust spatial relationships. The keypoint coordinates were perturbed randomly to simulate realistic noise conditions within ±0.03 as shown in Table 4.

Table 4
The jitter (additive noise) used to augment the sample data

Description	Time step 1	Time step 2
Time step	1	2
Sample input data	[0.1, 0.2, 0.3]	[0.4, 0.5, 0.6]
Augmented output data	[0.102, 0.198, 0.305]	[0.403, 0.502, 0.594]

3.2.3. Rotation

This augmentation technique rotates all hand keypoints around their Z axis (vertical axis) by ±15°, resulting in changing the coordinates (x, y, z) of each keypoint. This mimics the changes in camera position, enabling the model to learn and see motions such as small hand tilts and left and right facing without having to shift the camera's position. Therefore, the model is exposed to the movements, which can be observed from various angles, in order to reinforce the model. Table 5 shows sample data for this rotation of the x y coordinates of each feature.

Table 5
The rotation used to augment the sample data

Description	Time step 1	Time step 2
Time step	1	2
Sample input data	[0.1, 0.2, 0.3]	[0.4, 0.5, 0.6]
Augmented output data	[0.12, 0.18, 0.30]	[0.45, 0.52, 0.60]

Algorithm 1: Dynamic HGR Workflow

Input: Training dataset D, Deployment thresholds T = {Acc, Loss, F1, Latency}
Output: Trained model M^* integrated into Android TV system
Initialization: Set candidate architecture A
Preprocess D (labeling, normalization, partitioning)
Construct and train model M based on A
repeat
  Compute evaluation metrics {Acc, Loss, F1, Latency}
  Compare results with thresholds T
  if performance(M) < T **then**
    Apply data augmentation and normalization
    Tune hyperparameters
    Retrain M using D
  end if
until performance(M) ≥ T
Conduct real-time testing of M
if visual performance unsatisfactory **then**
  Select alternative architecture A'
  Set $A \leftarrow A'$ and retrain M
  Repeat evaluation and testing
end if
Integrate final model M^* into Android TV system
Map gestures to TV control commands
Validate real-time functionality through user interaction

3.2.4. Temporal masking

This augmentation method randomly masks 1–3 consecutive time steps in a gesture sequence, thereby simulating realistic scenarios in which parts of the hand are temporarily occluded from the camera view, for example, when an object or another

Table 6
The temporal masking used to augment the sample data

Description	Time step 1	Time step 2	Time step 3	Time step 4
Time step	1	2	3	4
Sample input data	[0.1, 0.2, 0.3]	[0.4, 0.5, 0.6]	[0.7, 0.8, 0.9]	[1.0, 1.1, 1.2]
Augmented output data	[0.1, 0.2, 0.3]	[0.4, 0.5, 0.6]	[0.0, 0.0, 0.0]	[0.0, 0.0, 0.0]

body part obstructs the camera. Introducing such artificial temporal gaps enables the model to learn from discontinuous temporal features and improves robustness to missing or incomplete data during inference. Sample data illustrating the masking of time steps 3 and 4 after augmentation are shown in Table 6.

3.2.5. Vector space transform

This augmentation applies independent random scaling factors in the range [0.95×, 1.05×] and translations of ±0.03 to all 2D and 3D hand keypoints. The method is designed to introduce natural biomechanical variability across users, such as differences in hand size and finger length, while also accounting for minor positional shifts or camera drift during gesture execution. By allowing controlled variations in hand scale and position while preserving inter-finger spatial relationships, the augmentation improves model generalization across users and viewing conditions and enhances robustness to real-world variability. Sample data in Table 7 illustrates scaling factors of [1.02, 0.98, 1.0] and translation offsets of [−0.01, 0.01, 0.0] applied to the keypoints.

Table 7
The vector space transform used to augment the sample data

Description	Time step 1	Time step 2
Time step	1	2
Sample input data	[0.1, 0.2, 0.3]	[0.4, 0.5, 0.6]
Augmented output data	[0.92, 0.206, 0.3]	[0.398, 0.5, 0.6]

3.3. Model development and training

For the purpose of spatiotemporal gesture recognition, many DL architectures were investigated. While the LSTM and GRU networks represented temporal connections, CNNs were used to extract spatial features from landmark sequences. Additionally, fusion architectures such as CNN-LSTM and CNN-GRU were studied. The models were able to select important frames within a sequence by integrating attention processes to enhance temporal discrimination.

The structures of the LSTM and GRU models are illustrated in Figures 2 and 3, respectively. As shown, the LSTM architecture is designed to capture long-term temporal dependencies by maintaining cell states across time steps [40], while the GRU architecture provides a more compact alternative by utilizing update and reset gates to regulate information flow [41]. The researchers investigated the suitability of these complementary designs for dynamic gesture recognition activities.

They trained the models with an 80/20 training-validation split. They applied the Adam optimizer, a batch size of 32, and

Figure 2
The LSTM architecture

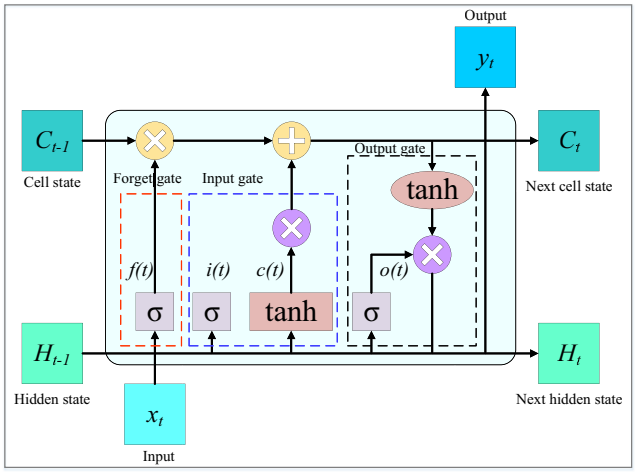
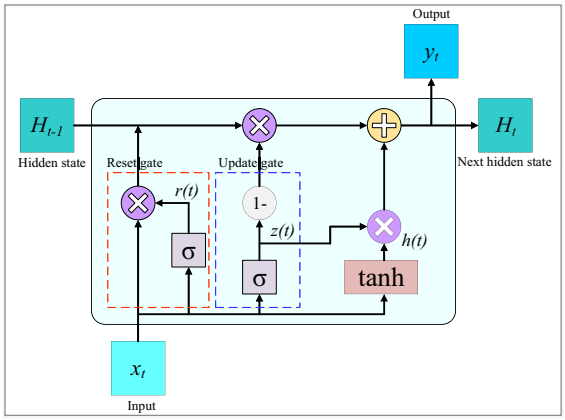


Figure 3
The GRU architecture



the *EarlyStopping* criterion to decrease model overfitting during training. The primary measures used for performance evaluation on a held-out test set were as follows: F1-score, accuracy, and prediction latency.

3.4 Optimization and deployment

By converting floating-point parameters into 8-bit integer representations, model quantization was used to make deployment on devices with limited resources easier. While preserving a respectable degree of accuracy, this modification dramatically decreased memory consumption and inference delay.

A middleware tool that converted recognized gestures into control instructions (such as navigation and volume adjustment)

was used to integrate the improved model into the Android TV platform. Socket programming was used to create communication between the Android TV and the recognition model, allowing for smooth, real-time interaction. Furthermore, a Windows platform prototype was developed that used the PyAutoGUI package to convert recognized gestures into system-level inputs (such as mouse and keyboard events).

3.5. Performance evaluation

Both quantitative and subjective metrics were used to evaluate the system's performance. Accuracy, precision, recall, F1-score, and latency were used to assess classification performance. Every experiment was performed several times by other users under the same recording settings in order to increase statistical reliability. To assess the stability and robustness of the model, performance measures were averaged over several trials, and the observed variance was analyzed. This recurrent multi-user evaluation allows for more trustworthy comparisons across the assessed models and lessens the volatility of single-run results. Additionally, interactive testing on the Android TV interface was used for real-world validation in order to guarantee responsiveness and resilience in practical usage scenarios.

4. Results and Discussion

The experimental findings and a thorough performance analysis of the suggested dynamic HGR system are presented in this part. From model training to real-world application, the outcomes are methodically arranged to assess the system's efficacy. The model training and validation methods are explained in Subsection 4.1, which also highlights the learning curves and model convergence efficiency. The researchers examined the relative effectiveness of the vectorization normalization and wrist centralization methods in Subsection 4.2. Furthermore, Subsection 4.3 presents the effect of data augmentation on the accuracy and robustness of the data classification technique. Finally, in Subsection 4.4, the researchers have compared the various classification metrics in various model architectures. They also studied the real-time performance of the proposed system, while emphasizing the prediction latency and the effect of the model optimization technique in Subsection 4.5. Practical applicability is further validated through usability testing on an Android TV emulator, as discussed in Subsection 4.6. A broader discussion of the findings and their implications is provided in Subsection 4.7.

All experiments were conducted locally on a workstation equipped with an AMD Ryzen 5 3550H CPU @ 2.10 GHz, 16 GB RAM, running the Windows 11 operating system.

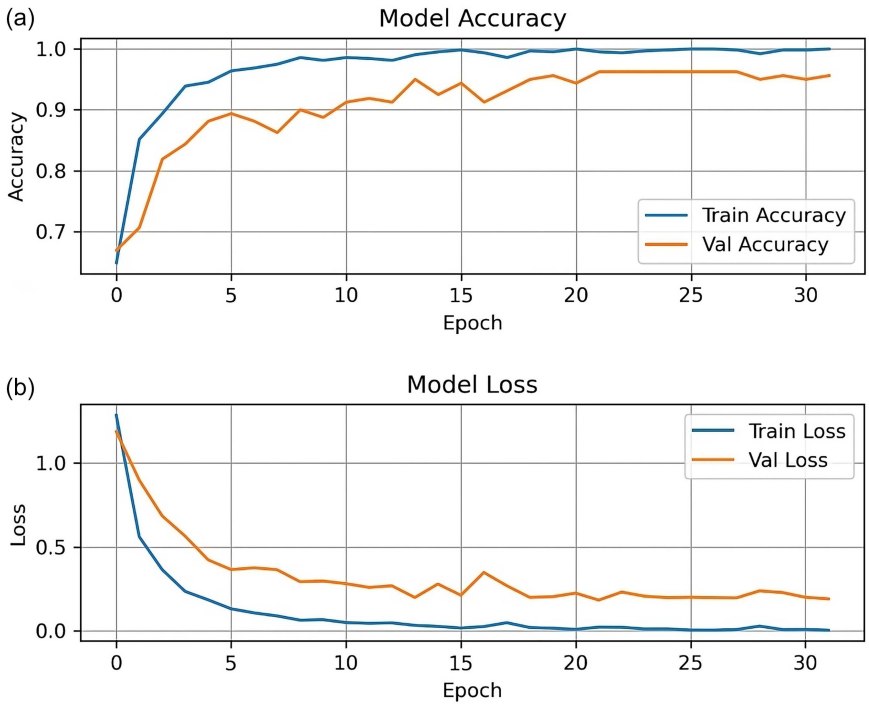
4.1. Training and validation

The learning curves of the GRU model with an attention mechanism, trained with a batch size of 32 for a maximum of 100 epochs, are shown in Figure 4. To reduce overfitting, an Early Stopping callback was used. The model weights were returned to those from epoch 22 when training ended at epoch 32, resulting in the lowest validation loss of 0.1827. This outcome resulted from setting the patience parameter to 10, which allowed training to continue without improvement for up to ten more epochs. The model's total training time was 29.45 s.

4.2. Comparison between wrist centralization and vectorization normalization

The GRU architecture was combined with an attention mechanism utilizing the same training data. Two normalizing techniques, specifically wrist centralization and vectorization, were employed to assess their influence on classification performance. The researchers used the vector-based normalization on the MediaPipe-extracted hand keypoints for improving the

Figure 4
The learning curves of the GRU model with attention mechanism, namely, (a) the accuracy curve and (b) the loss curve



model's robustness against illumination and occlusion-related changes. This format improves generalization across users and perspectives by reducing vulnerability to rotation, scaling, and small keypoint noise by transforming absolute coordinates into unit-length directional vectors between the physically linked joints.

Table 8 presents the results, which show that vectorization performed slightly better than the wrist centralization technique across all the evaluation criteria. The results indicate that despite its limitations, the vectorization technique presents a robust feature representation, which further increases the data classification reliability.

4.3. Impact of data augmentation

To assess the impact of data augmentation, the researchers trained the LSTM architecture with an integrated attention mechanism on similarly vectorized datasets with and without augmentation. Table 9 is a compilation of the classification metrics.

The data augmentation procedure produced a 1.5% relative accuracy gain and enhanced performance across all criteria, as Table 9 demonstrates. Significantly improved F1-score, accuracy, and recall values were observed by the researchers, suggesting that augmentation improves the model's capacity for generalization.

These results demonstrate the value of data augmentation in lowering overfitting and boosting the robustness of gesture recognition systems.

4.4. Architecture comparison

Eight DL models were developed and trained under the same experimental settings in order to thoroughly evaluate the effect of architectural variations on dynamic HGR. All designs employed the same preprocessing workflow, which includes batch normalization, vectorization, and data augmentation, to provide an impartial and fair comparison.

Every model was built as a sequential network that integrated convolutional feature extractors (1D-CNN), recurrent units (LSTM or GRU), and occasionally attention techniques. Figure 5 shows representative schematic designs for each of the eight architectures. The models are arranged side by side based on structural similarities for conciseness and clarity, minimizing repetition while upholding a professional presentation quality.

Table 10 shows the performance evaluation results. The baseline LSTM model has superior accuracy compared to the standard GRU, signifying its enhanced ability to capture temporal connections in sequential data. Second, the integration of the attention mechanism yields significant improvements in GRU-based models, narrowing the performance disparity with LSTM and, in certain instances, surpassing it in precision and F1-score. Third, enhancing recurrent models with a 1D-CNN layer marginally enhances classification performance for both LSTM and GRU, especially in terms of precision. However, these improvements come at the cost of increased architectural complexity and parameter size, which may limit deployment in real-time or resource-constrained environments. The confusion matrix of the 1D-CNN-GRU with Attention model for 10 hand gestures is shown in Figure 6.

The 1D-CNN-LSTM architecture demonstrates superior performance compared to the assessed models, with an accuracy of 98.00% and an F1-score of 97.98%. However, when prioritizing model efficiency, the GRU with an attention mechanism provides a trade-off, achieving competitive performance while preserving a reasonably lightweight architecture.

4.5. Real-time performance

4.5.1. Comparison before and after quantization using test samples

To assess the efficacy of model optimization, the GRU with architecture was examined before and after quantization. Three major performance factors were evaluated: model size, prediction delay, and classification accuracy. Prediction latency indicates the system's responsiveness in real-time applications, accuracy assesses the model's ability to categorize dynamic hand movements accurately, and model size represents a memory footprint, which is vital for deployment on resource-constrained devices.

Quantization significantly reduces model size and prediction delay, as seen in Table 11. Latency decreased from 102.16 ms to 1.74 ms (a 98.3% reduction), and model size reduced from kB to 176.82 kB (a 66.3% reduction). The results show that quantization preserves predictive performance while significantly improving computational efficiency, with only a 1% drop in classification accuracy (from 98% to 97%). These findings highlight the effectiveness of quantization in enabling real-time gesture detection on embedded and low-power systems.

4.5.2 Comparison before and after quantization in real-time testing

The researchers measured the frame delay in real time throughout multiple system phases, including frame capture, preprocessing, model inference (prediction latency), keypoint extraction, data post-processing, and rendering. They carried out

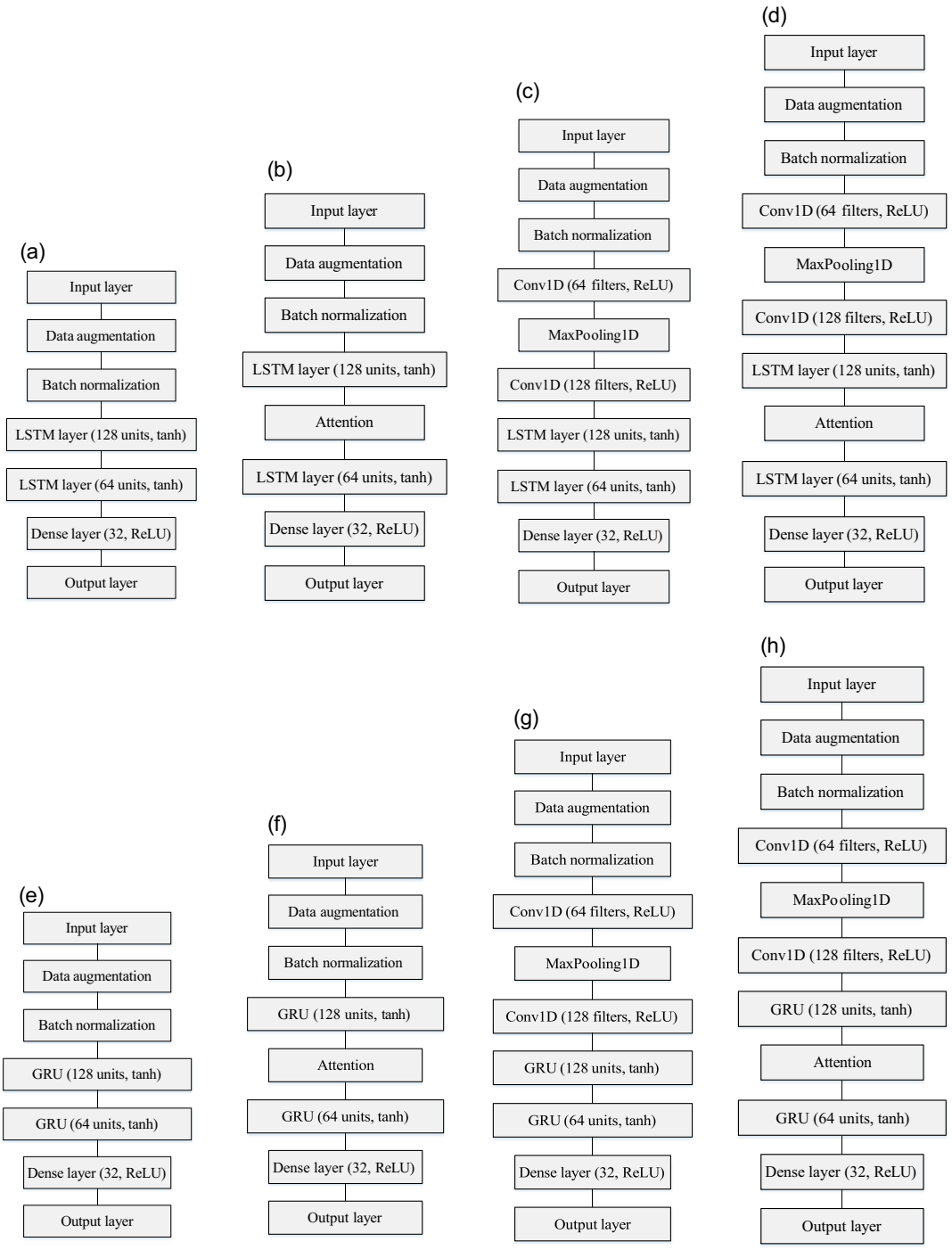
Table 8
A comparison of the classification metrics of wrist centralization and vectorization normalization methods

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Wrist centralization	97.00	97.06	97.00	97.00
Vectorization	98.50	98.49	98.50	98.48

Table 9
A comparison of the classification metrics with and without data augmentation

Condition	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Without augmentation	96.50	96.60	96.50	96.49
With augmentation	98.00	98.06	98.00	97.99

Figure 5
The DL models evaluated in this study, namely, (a) LSTM, (b) LSTM with attention, (c) 1D-CNN-LSTM, (d) 1D-CNN-LSTM with attention mechanism, (e) GRU, (f) GRU with attention mechanism, (g) 1D-CNN-GRU, and (h) 1D-CNN-GRU with attention mechanism



a comparative evaluation of the prediction and frame latencies, both before and after quantization, for determining their effect on the system responsiveness.

According to Table 12, the optimized model significantly reduced the prediction latency from 95.67 ms to 2.02 ms (a decrease of 97.9%), whereas the total frame latency decreased from 163.35 ms to 77.12 ms (a 52.8% reduction). These enhancements highlight how quantization improves end-to-end system performance during real-time testing in addition to improving inference. The significant reduction in frame latency is primarily attributed to faster model inference, confirming that quantization is an effective strategy for enabling low-latency, interactive gesture recognition applications.

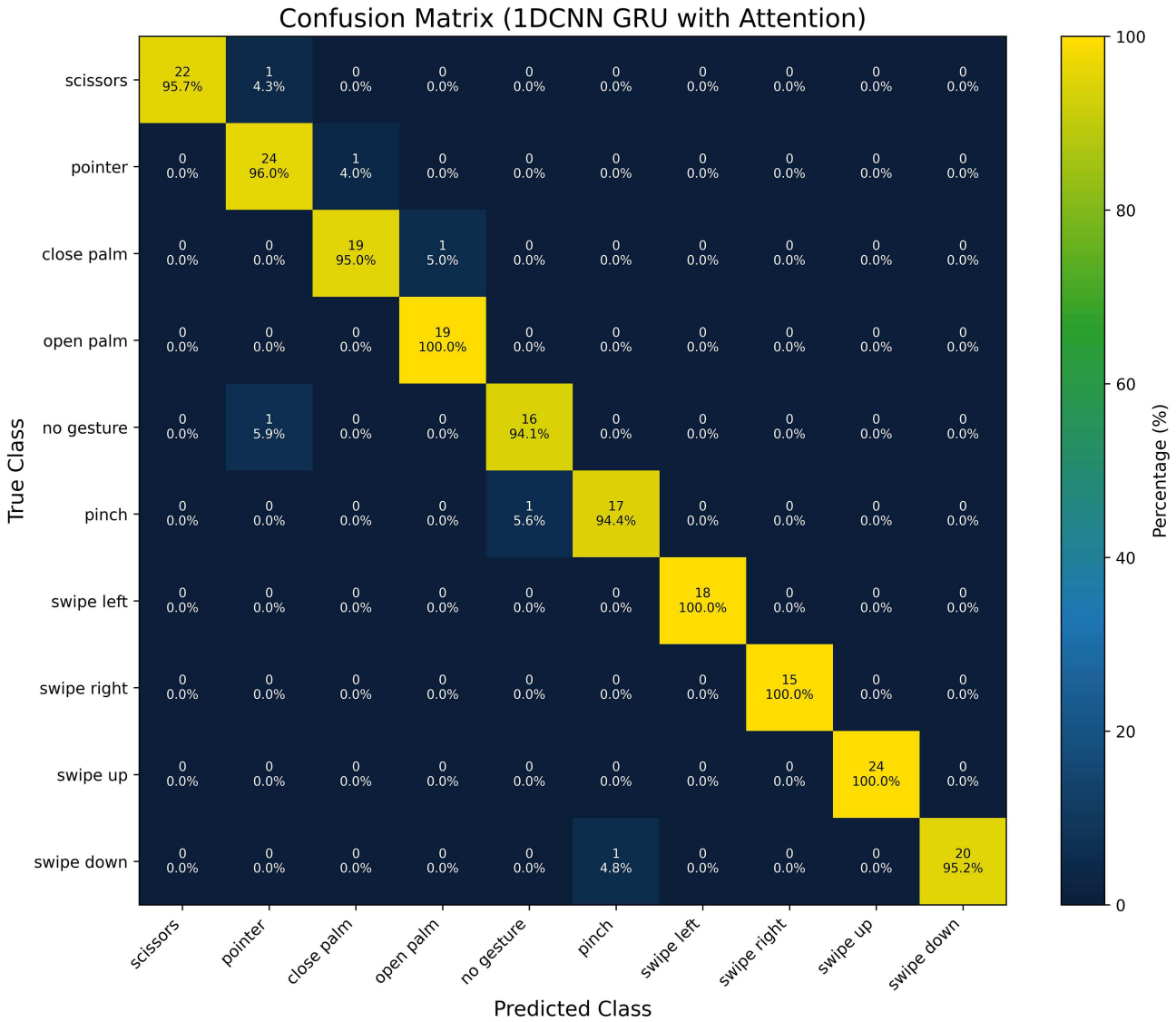
4.6. Usability testing

To evaluate the practical performance of the proposed system, the trained GRU with attention mechanism was integrated

Table 10
A comparison of the classification metrics across different architectures

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
LSTM	97.00	97.18	97.00	96.94
LSTM + Attention	97.00	96.99	97.00	96.87
1D-CNN-LSTM	98.00	98.16	98.00	97.98
1D-CNN-LSTM + Attention	97.50	97.64	97.50	97.47
GRU	95.50	95.43	95.50	95.34
GRU + Attention	97.50	97.54	97.50	97.47
1D-CNN-GRU	97.50	97.70	97.50	97.44
1D-CNN-GRU + Attention	97.00	96.99	97.00	96.92

Figure 6
The confusion matrix of the 1D-CNN-GRU with attention



into an Android application and tested on an Android TV emulator. To allow gesture-based control, a socket-based communication channel was designed between the client application and the inference server. The server activated the recognition model and analyzed the real-time gesture data when the client began the connection.

Dynamic hand motions were successfully determined by the system and transferred to the Android TV navigation and control features. Table 13 presents the mapping between assigned control actions and established gesture vocabulary. For example, *swipe* gestures supported directional navigation, whereas *open palm* and *close palm* were mapped to the back and enter functions,

Table 11
A comparison of model size, prediction latency, and accuracy before and after quantization

Method	Model size (kB)	Prediction latency (ms)	Accuracy (%)
Before quantization	524.91	102.16	98.00
After quantization	176.82	1.74	97.00

Table 12
A comparison of frame latency and prediction latency before and after quantization

Method	Frame latency (ms)	Prediction latency (ms)
Before quantization	163.35	95.67
After quantization	77.12	2.02

Table 13
Mapping the hand gesture vocabulary to control functions on an Android TV emulator

Hand gesture	Control function
Swipe left	Move selection left
Swipe right	Move selection right
Swipe up	Move selection up
Swipe down	Move selection down
Open palm	Back function
Close palm	Enter/confirm selection
Pinch	Adjust audio volume
Pointer	Control cursor movement
Scissors	Cursor click
No gesture	No operation

respectively. More advanced gestures such as *pinch, pointer*, and *scissors* enabled volume adjustment, cursor movement, and selection, thereby demonstrating the versatility of the system in supporting natural HCI interactions.

The results of usability studies revealed that the application showed a consistent response to all actions, which led to an intuitive and seamless user experience. A constant fingertip tracking yielded a slight latency in cursor-based gestures (like pointer), but the responsiveness was still suited for real-time interaction.

Figure 7 presents a list of representative examples for gesture execution. The researchers calculated the task completion time of 31 prediction examples, including preprocessing, model inference, frame collection, and socket communication. The system meets the real-time HCI requirements for smart TV platforms with an average task completion time of 605.84 ms.

The findings show that the proposed gesture-based control model is feasible for real-world HCI. The high recognition accuracy, low latency, and wide-ranging capabilities of the system show its potential to enhance the accessibility and user experience for a variety of interactive multimedia applications.

4.7. Discussion

The researchers in this study have proposed the use of a novel dynamic HGR system for HCI, which showed an excellent performance, with an average real-time prediction latency of 2.022 ms and a classification accuracy of 97.5%. The aforementioned findings suggest that our method was appropriate for interactive applications when compared to the existing recognition frameworks. With a little increase in processing cost, the GRU's resilience and temporal consistency of sequential gesture interpretation were greatly improved with the addition of the attention mechanism. The system did, however, maintain an interactive frame rate, which may suggest practical applications.

Compared to vision-based systems that employ raw input photos, this system offers a significant decrease in data dimension through the incorporation of keypoints created by MediaPipe. This method loads data quickly, uses less memory, and improves forward and backward propagation during training. By emphasizing the fundamental hand motions and reducing the unnecessary background information, the keypoint-based features also help to simplify the model design without sacrificing identification accuracy.

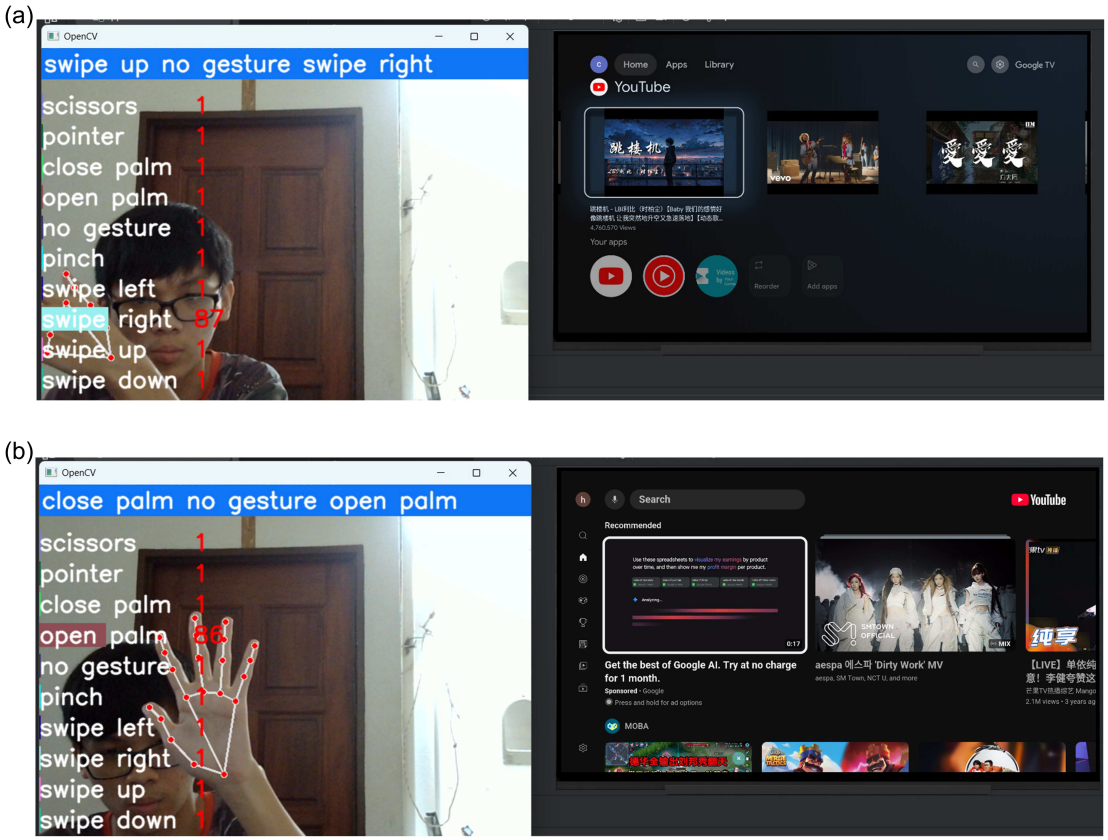
Additionally, normalization techniques were essential for enhancing model robustness. It was noted that the vectorization-based normalization techniques showed a better performance in comparison to the wrist-centric normalization process since it converted the absolute keypoint positions into unit directional vectors. This method minimizes the effects of keypoint instability that is initiated by the rapid motions or partial occlusions and is intrinsically robust to rotation, translation, and scaling. Consequently, the model showed increased generality across hand sizes, users, and camera views, resulting in more dependable and consistent performance in actual HCI scenarios.

The researchers conducted ablation tests by isolating vectorization, attention mechanisms, and data augmentation in order to evaluate the contribution of each component in the suggested pipeline. To lessen run-to-run variability, they averaged all data after training and assessing each configuration in a series of tests with various users. The study's findings demonstrate that vectorization aids in resilience to positional noise, attention aids in temporal consistency for dynamic gestures, and data augmentation aids in generalizing information across various execution styles and users. The ideal balance between accuracy, inference efficiency, and stability was consistently attained by the integrated architecture.

In order to assess effective temporal modeling and real-time deployment with minimal computational resources, this work focuses on dynamic hand gesture classification utilizing pre-segmented gesture clips. The system manages temporally ordered 3D hand keypoints collected frame by frame using MediaPipe Holistic, which recognizes motion dynamics over time, even though a consistent gesture stream segmentation was not addressed. These temporally structured 3D keypoint sequences, which capture both motion patterns and static hand locations across a number of frames, serve as the foundation for dynamic gesture detection.

In order to facilitate easier model training, steady temporal consistency, and dependable real-time performance on platforms with limited resources, such as Android TV, the researchers in this work employed a constant input length of 30 frames. They extracted hand markers from the motion sequences at a frame rate of 30 using the MediaPipe Holistic method. Fixed-dimensional feature vectors were produced as a result, and when

Figure 7
Examples of gesture executions on an Android TV emulator, namely, (a) *swipe right* for navigation and (b) *open palm* for the back function



no hands were found, they were zero-padded. Because the stated end-to-end delay accounted for preprocessing, frame collection, data inference, and communication overhead, this method made deterministic latency analysis possible. For example, the average end-to-end latency of the system for 31 prediction cases was 605.84 ms, which showed that it was suited for the real HCI applications. However, this fixed-length formulation provides reliable real-time deployment but may limit adaptability to naturally variable gesture dimensions.

The proposed dynamic hand gesture detection system showed good performance, but the researchers noted some limitations during the installation and assessment stages. Firstly, the quality of the keypoint extraction of MediaPipe Holistic significantly affected the accuracy of the system. It was observed that MediaPipe worked well in the ideal conditions, but some scenarios with fast hand movements, drastic lighting changes, or partial occlusions can generate jitters or uneven keypoints. Such variations on the input data led to inconsistent model predictions and resulted in a less reliable gesture classification system.

Model weights and activations were quantized and compressed to lower precision forms in order to further minimize inference delay. This modification accelerated the predictions by using efficient integer arithmetic to lower the amount of memory bandwidth needed. As a result, quantization maintained the model's functionality and enabled it to work effectively under resource constraints.

Among all the models tested, the attention-augmented GRU offered the best trade-off in the responsiveness, accuracy, and model compactness trade-off. Real-time evaluations revealed its better consistency and inference speed in comparison to its LSTM-based competitors. Its simple architecture makes it especially suitable for deploying on embedded devices like Android TV, despite the absence of convolutional layers.

The researchers identified a few drawbacks in spite of these benefits. The pointer function displayed notable delay due to the socket-based connection between the Android client and inference server, although the majority of gesture-mapped control functions operated dependably. The real-time responsiveness of cursor movements was reduced by this communication cost. Future research will concentrate on moving from a client-server architecture to on-device inference in order to overcome this limitation. This will entail deploying the trained model directly on the Android TV using TensorFlow Lite or other optimized frameworks, which will lower end-to-end latency and network overhead.

5. Conclusion and Future Work

This research proposes a real-time dynamic HGR system that uses DL models to classify ten gesture sequences and the MediaPipe Holistic system to extract 3D keypoints. The GRU with attention mechanism offered the best trade-off between model size, accuracy, and inference latency among the architectures being assessed. Post-training quantization decreased the model size by around 66% and enhanced responsiveness, making it easier to deploy on resource-constrained embedded systems. In controlled experimental circumstances, real-time interaction

and functional integration were shown through system validation utilizing an Android TV emulator.

The researchers identified a few shortcomings in spite of these successes. The validity of MediaPipe's keypoint extraction, which can be diminished by quick hand motions, dim lighting, or obstacles, can have a significant impact on identification accuracy. Pointer control was particularly impacted by latency in socket-based communication between the Android client and the inference server for real-time assessments. Furthermore, while the current gesture vocabulary is useful, it is still constrained and cannot handle complex interactions.

More intricate and organic gestures for usage in augmented reality (AR) and virtual reality (VR), gaming, and smart device applications will be added to the gesture set in the future. Multi-user scenarios, which are emphasized as a crucial subject for more research to allow dependable implementation in actual smart-home contexts, were not expressly taken into account by the solution suggested in this study. Rather, it is designed for single-user interactions in controlled contexts. In order to assess the real-time performance and viability of deployment on systems with restricted resources, the researchers conducted an experimental study on the internal dataset. Future research has been suggested to focus on cross-dataset robustness and generalization assessment on large-scale public datasets including SHREC, DHG14/28, and EgoGesture. Using an Android TV emulator under controlled conditions, the researchers examined the system's end-to-end latency, gesture-to-command mapping, and functional integration. The majority of gesture-driven controls performed well; however, emulator-based testing is unable to adequately represent actual network circumstances, camera behavior, and CPU load. Therefore, on-device testing on commercial Android TV hardware is a crucial topic for further research. To increase scalability and lower network latency, the transition from a client-server configuration to on-device inference utilizing TensorFlow Lite or other frameworks must be given top priority. In order to improve resilience in a variety of environmental and tracking settings, sensor fusion technologies—such as combining RGB data with depth sensing or inertial measurement units—will also be investigated. These advancements are likely to improve the usability and adaptability of dynamic HGR systems in real-world HCI.

Acknowledgment

The authors are pleased and thankful to Universiti Sains Malaysia, Seberang Perai Selatan, 14300 Nibong Tebal, Penang, Malaysia, for the support in terms of hardware, equipment, and lab facilities.

Funding Support

This work was supported by the Universiti Sains Malaysia PRIME Grant (R502-KRARP003-0000000897-K134).

Ethical Statement

The authors declare that this study did not require formal ethical approval because it involved voluntary participation in a noninvasive hand gesture data collection task and did not involve medical procedures, sensitive personal information, or vulnerable populations. All participants provided informed consent prior to participation. The collected data were anonymized and used solely for research purposes.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data available on request from the corresponding author upon reasonable request.

Author Contribution Statement

Chim Tun Sian: Conceptualization, Methodology, Software, Investigation, Data curation, Writing – original draft. **Suhardi Azliy Junoh:** Validation, Formal analysis, Resources, Data curation, Writing – review & editing, Supervision, Project administration. **Mohd Shahrimie Mohd Asaari:** Methodology, Validation, Formal analysis, Investigation, Resources, Writing – review & editing, Visualization, Funding acquisition.

References

[1] Rahman, M. M., Uzzaman, A., Khatun, F., Aktaruzzaman, M., & Siddique, N. (2025). A comparative study of advanced technologies and methods in hand gesture analysis and recognition systems. *Expert Systems with Applications*, *266*, 125929. https://doi.org/10.1016/j.eswa.2024.125929

[2] Ahmad, K. A., Higashi, T., & Yoshida, K. (2025). Dynamic hand gesture recognition by hand landmark classification using Long Short-Term Memory. *Pertanika Journal of Science & Technology*, *33*(S2), 73–84. https://doi.org/10.47836/pjst.33.s2.05

[3] Xiao, Y., Liu, T., Han, Y., Liu, Y., & Wang, Y. (2024). Realtime recognition of dynamic hand gestures in practical applications. *ACM Transactions on Multimedia Computing, Communications, and Applications*, *20*(2), 50. https://doi.org/10.1145/3561822

[4] Chang, V., Eniola, R. O., Golightly, L., & Xu, Q. A. (2023). An exploration into human–computer interaction: Hand gesture recognition management in a challenging environment. *SN Computer Science*, *4*(5), 441. https://doi.org/10.1007/s42979-023-01751-y

[5] Alonazi, M., Ansar, H., Al Mudawi, N., Alotaibi, S. S., Almujally, N. A., Alazeb, A., . . . , & Min, M. (2023). Smart healthcare hand gesture recognition using CNN-based detector and deep belief network. *IEEE Access*, *11*, 84922–84933. https://doi.org/10.1109/ACCESS.2023.3289389

[6] Tchantchane, R., Zhou, H., Zhang, S., & Alici, G. (2023). A review of hand gesture recognition systems based on noninvasive wearable sensors. *Advanced Intelligent Systems*, *5*(10), 2300207. https://doi.org/10.1002/aisy.202300207

[7] Abdallah, M. S., Samaan, G. H., Wadie, A. R., Makhmudov, F., & Cho, Y.-I. (2022). Light-weight deep learning techniques with advanced processing for real-time hand gesture recognition. *Sensors*, *23*(1), 2. https://doi.org/10.3390/s23010002

[8] Qi, J., Ma, L., Cui, Z., & Yu, Y. (2024). Computer vision-based hand gesture recognition for human-robot interaction: A review. *Complex & Intelligent Systems*, *10*(1), 1581–1606. https://doi.org/10.1007/s40747-023-01173-6

[9] Yu, Z., Lu, Y., An, Q., Chen, C., Li, Y., & Wang, Y. (2022). Real-time multiple gesture recognition: Application of a lightweight individualized 1D CNN model to an edge

computing system. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *30*, 990–998. https://doi.org/10.1109/TNSRE.2022.3165858

[10] Rodríguez-Moreno, I., Freire-Obregón, D. S., Martínez-Otzeta, J. M., & Castrillón-Santana, M. F. (2025). Real-time hand gesture recognition: Lightweight keypoint-based approach with medoid similarity. *Journal of Advances in Information Technology*, *16*(4), 447–457. https://doi.org/10.12720/jait.16.4.447-457

[11] Liu, X., Zhang, Y., Yan, Z., & Ge, Y. (2023). Defining 'seamlessly connected': User perceptions of operation latency in cross-device interaction. *International Journal of Human-Computer Studies*, *177*, 103068. https://doi.org/10.1016/j.ijhcs.2023.103068

[12] Filipowska, A., Filipowski, W., Raif, P., Pieniążek, M., Bodak, J., Ferst, P., . . . , & Grzegorzek, M. (2024). Machine learning-based gesture recognition glove: Design and implementation. *Sensors*, *24*(18), 6157. https://doi.org/10.3390/s24186157

[13] Shin, J., Miah, A. S. M., Kabir, M. H., Rahim, M. A., & Al Shiam, A. (2024). A methodological and structural review of hand gesture recognition across diverse data modalities. *IEEE Access*, *12*, 142606–142639. https://doi.org/10.1109/ACCESS.2024.3456436

[14] Crogman, H. T., Cano, V. D., Pacheco, E., Sonawane, R. B., & Boroon, R. (2025). Virtual reality, augmented reality, and mixed reality in experiential learning: Transforming educational paradigms. *Education Sciences*, *15*(3), 303. https://doi.org/10.3390/educsci15030303

[15] Evangelou, G., Georgiou, O., Brown, E., & Moore, J. W. (2025). Sense of agency in gesture-based interactions: Modulated by sensory modality but not feedback meaning. *Frontiers in Computer Science*, *7*, 1511928. https://doi.org/10.3389/fcomp.2025.1511928

[16] Sarma, D., & Bhuyan, M. K. (2021). Methods, databases and recent advancement of vision-based hand gesture recognition for HCI systems: A review. *SN Computer Science*, *2*(6), 436. https://doi.org/10.1007/s42979-021-00827-x

[17] Herbert, O. M., Pérez-Granados, D., Ruiz, M. A. O., Cadena Martínez, R., Gutiérrez, C. A. G., & Antuñano, M. A. Z. (2024). Static and dynamic hand gestures: A review of techniques of virtual reality manipulation. *Sensors*, *24*(12), 3760. https://doi.org/10.3390/s24123760

[18] Xie, J., Xu, Z., Zeng, J., Gao, Y., & Hashimoto, K. (2025). Human–robot interaction using dynamic hand gesture for teleoperation of quadruped robots with a robotic arm. *Electronics*, *14*(5), 860. https://doi.org/10.3390/electronics14050860

[19] Roggio, F., Trovato, B., Sortino, M., & Musumeci, G. (2024). A comprehensive analysis of the machine learning pose estimation models used in human movement and posture analyses: A narrative review. *Heliyon*, *10*(21), e39977. https://doi.org/10.1016/j.heliyon.2024.e39977

[20] Docekal, J., Rozlivek, J., Matas, J., & Hoffmann, M. (2022). Human keypoint detection for close proximity human-robot interaction. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots*, 450–457. https://doi.org/10.1109/Humanoids53995.2022.10000133

[21] Wang, P., Fan, E., & Wang, P. (2021). Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, *141*, 61–67. https://doi.org/10.1016/j.patrec.2020.07.042

[22] Khodabandelou, G., Jung, P.-G., Amirat, Y., & Mohammed, S. (2021). Attention-based gated recurrent unit for gesture recognition. *IEEE Transactions on Automation Science and Engineering*, *18*(2), 495–507. https://doi.org/10.1109/TASE.2020.3030852

[23] Yu, J., Qin, M., & Zhou, S. (2022). Dynamic gesture recognition based on 2D convolutional neural network and feature fusion. *Science Reports*, *12*, 4345. https://doi.org/10.1038/s41598-022-08133-z

[24] Mekruksavanich, S., Phaphan, W., & Jitpattanakul, A. (2025). Sensor-based hand gesture recognition using one-dimensional deep convolutional and residual bidirectional gated recurrent unit neural network. *Lobachevskii Journal of Mathematics*, *46*, 464–480. https://doi.org/10.1134/S1995080224608166

[25] Hax, D. R. T., Penava, P., Krodel, S., Razova, L., & Buettner, R. (2024). A novel hybrid deep learning architecture for dynamic hand gesture recognition. *IEEE Access*, *12*, 28761–28774. https://doi.org/10.1109/ACCESS.2024.3365274

[26] Mallik, B., Rahim, M. A., Miah, A. S. M., Yun, K. S., & Shin, J. (2024). Virtual keyboard: A real-time hand gesture recognition-based character input system using LSTM and MediaPipe holistic. *Computer Systems Science & Engineering*, *48*(2), 555–570. https://doi.org/10.32604/csse.2023.045981

[27] Tan, C. K., Lim, K. M., Lee, C. P., Chang, R. K. Y., & Alqahtani, A. (2023). SDViT: Stacking of distilled vision transformers for hand gesture recognition. *Applied Sciences*, *13*(22), 12204. https://doi.org/10.3390/app132212204

[28] Aly, M., & Fathi, I. S. (2025). Recognizing American sign language gestures efficiently and accurately using a hybrid transformer model. *Scientific Reports*, *15*(1), 20253. https://doi.org/10.1038/s41598-025-06344-8

[29] Tan, C. K., Lim, K. M., Chang, R. K. Y., Lee, C. P., & Alqahtani, A. (2023). HGR-ViT: Hand gesture recognition with vision transformer. *Sensors*, *23*(12), 5555. https://doi.org/10.3390/s23125555

[30] Ikne, O., Allaert, B., & Wannous, H. (2024). Skeleton-based self-supervised feature extraction for improved dynamic hand gesture recognition. In *2024 IEEE 18th International Conference on Automatic Face and Gesture Recognition*, 1–10. https://doi.org/10.1109/FG59268.2024.10581975

[31] Mienye, I. D., Swart, T. G., & Obaido, G. (2024). Recurrent neural networks: A comprehensive review of architectures, variants, and applications. *Information*, *15*(9), 517. https://doi.org/10.3390/info15090517

[32] Shiri, F. M., Perumal, T., Mustapha, N., & Mohamed, R. (2024). A comprehensive overview and comparative analysis on deep learning models. *Journal on Artificial Intelligence*, *6*(1), 301–360. https://doi.org/10.32604/jai.2024.054314

[33] Ilham, A. A., Nurtanio, I., Ridwang, & Syafaruddin. (2024). Applying LSTM and GRU methods to recognize and interpret hand gestures, poses, and face-based sign language in real time. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, *28*(2), 265–272. https://doi.org/10.20965/jaciii.2024.p0265

[34] Mekruksavanich, S., Phaphan, W., & Jitpattanakul, A. (2025). Sensor-based hand gesture recognition using one-dimensional deep convolutional and residual bidirectional gated recurrent unit neural network. *Lobachevskii Journal of Mathematics*, *46*(1), 464–480. https://doi.org/10.1134/S1995080224608166

[35] Ridwang, I., Ilham, A. A., Nurtanio, I., & Syafaruddin. (2023). Dynamic sign language recognition using MediaPipe library and modified LSTM method. *International Journal on*

Advanced Science, Engineering and Information Technology, *13*(6), 2171–2180. https://doi.org/10.18517/ijaseit.13.6.19401

[36] Mohamed, N., Mustafa, M. B., & Jomhari, N. (2021). A review of the hand gesture recognition system: Current progress and future directions. *IEEE Access*, *9*, 157422–157436. https://doi.org/10.1109/ACCESS.2021.3129650

[37] Niu, P. (2025). Convolutional neural network for gesture recognition human-computer interaction system design. *PLoS One*, *20*(2), e0311941. https://doi.org/10.1371/journal.pone.0311941

[38] Mumuni, A., & Mumuni, F. (2022). Data augmentation: A comprehensive survey of modern approaches. *Array*, *16*, 100258. https://doi.org/10.1016/j.array.2022.100258

[39] Kumar, T., Brennan, R., Mileo, A., & Bendechache, M. (2024). Image data augmentation approaches: A comprehensive survey and future directions. *IEEE Access*, *12*, 187536–187571. https://doi.org/10.1109/ACCESS.2024.3470122

[40] Junoh, S. A., & Pyun, J.-Y. (2024). Augmentation of finger-prints for indoor BLE localization using conditional GANs. *IEEE Access*, *12*, 30176–30190. https://doi.org/10.1109/ACCESS.2024.3368449

[41] Hussain, A. H. A., Taher, M. A., Mahmood, O. A., Hammadi, Y. I., Alkanhel, R., Muthanna, A., & Koucheryavy, A. (2023). Urban traffic flow estimation system based on gated recurrent unit deep learning methodology for Internet of Vehicles. *IEEE Access*, *11*, 58516–58531. https://doi.org/10.1109/ACCESS.2023.3270395

How to Cite: Sian, C. T., Junoh, S. A., & Asaari, M. S. M. (2026). Dynamic Hand Gesture Recognition for Human–Computer Interaction. *Artificial Intelligence and Applications*. https://doi.org/10.47852/bonviewAIA62027479