Indonesian Journal of Electrical Engineering and Computer Science
Vol. 40, No. 2, November 2025, pp. 883∼897
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v40.i2.pp883-897

Deep-learning-based hand gestures recognition applications for game controls

Huu-Huy Ngo, Hung Linh Le, Man Ba Tuyen, Vu Dinh Dung, Tran Xuan Thanh
Thai Nguyen University of Information and Communication Technology, Thai Nguyen, Vietnam

Article Info

Article history:

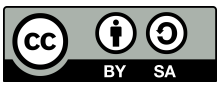
Received Jan 12, 2025
Revised Jul 21, 2025
Accepted Oct 14, 2025

Keywords:

Action recognition
Deep learning
Game controls
Hand gestures recognition
Human–computer interaction

ABSTRACT

Hand gesture recognition is among the emerging technologies of human-computer interaction, and an intuitive and natural interface is more preferable for such applications than a total solution. It is also widely used in multimedia applications. In this paper, a deep learning-based hand gesture recognition system for controlling games is presented, showcasing its significant contributions toward advancing the frontier of natural and intuitive human-computer interaction. It utilizes MediaPipe to get real-time skeletal information of hand landmarks and translates the gestures of the user into smooth control signals through an optimized artificial neural network (ANN) that is tailored for reduced computational expenses and quicker inference. The proposed model, which was trained on a carefully selected dataset of four gesture classes under different lighting and viewing conditions, shows very good generalization performance and robustness. It gives a recognition rate of 99.92% with much fewer parameters than deeper models such as ResNet50 and VGG16. By achieving high accuracy, computational speed, and low latency, this work addresses some of the most important challenges in gesture recognition and opens the way for new applications in gaming, virtual reality, and other interactive fields.



Corresponding Author:

Hung Linh Le
Thai Nguyen University of Information and Communication Technology
Thai Nguyen, Vietnam
Email: lhlinh@ictu.edu.vn

1. INTRODUCTION

Hand gesture recognition is a fundamental element of contemporary human–computer interaction (HCI) that offers a more natural, touchless, and intuitive control paradigm than traditional input devices such as keyboards, touchscreens, or mice. Its application is extensive and covers areas such as virtual and augmented reality (VR/AR), home automation technologies, assistive technology, robots, and gaming systems [1]-[4]. The advancement of sensing technology and computer vision algorithms has largely reduced most of the technical difficulties, e.g., partial occlusion, background clutter, and changes in lighting [5]-[7].

Deep learning, especially of convolutional neural networks (CNNs), has transformed the area of human gesture recognition with its proven capability of extracting spatial along with temporal features from images and video frames. VGG16, ResNet50, and DenseNet are some of the models that have been extensively utilized and modified for gesture recognition tasks with state-of-the-art accuracy on benchmarking datasets [4], [8], [9]. In particular, Sharma and Singh [4] applied CNNs and preprocessing methods (PCA, ORB, and histogram gradients) to improve the accuracy of recognition, while Mohammed *et al.* [8] fused color and depth data from Kinect sensors with hybrid models. Devineau *et al.* [10] also addressed temporal dynamics with the

use of skeletal joint data and parallel convolutions. While being precise, such CNN-based models typically accompany millions of parameters, resulting in an expensive computational cost that discourages their real-time implementation on low-resource devices.

Given these limitations, researchers have explored lightweight architectures. The combination of artificial neural networks (ANNs) with effective feature extraction offers a beneficial trade-off between accuracy and operational efficiency. Zhang *et al.* [11] and Nasri *et al.* [12] presented real-time gesture recognition systems using sEMG signals paired with ANN classifiers, demonstrating excellent performance with fast inference times. Similarly, Ozdemir *et al.* [13] and Cruz *et al.* [14] used spectral and inertial inputs to classify gestures. Mujahid *et al.* [15] used YOLOv3 with DarkNet-53 for real-time detection of static and dynamic gestures without preprocessing, whereas Aggarwal and Arora [16] used mobile-based HGR in game scenarios.

Meanwhile, recent gesture recognition research has focused on practicality, multimodality, and flexibility. Lee and Bae [17] suggested a deep learning-based glove using soft sensors for dynamic motion. Sen *et al.* [18] suggested a hybrid framework that fuses CNNs, ViT, and Kalman filtering for stable real-time control. Osama *et al.* [19] and Guo *et al.* [20] were concerned with the incorporation of gesture control in presentation and educational systems. Jiang *et al.* [21] were concerned with novel wearable HGR systems, whereas Naseer *et al.* [22] developed UAV control modules using gesture detection. Wen *et al.* [23] proposed an innovative mixed reality system aimed at enhancing sign language education through immersive learning experiences and comprehensive, real-time feedback mechanisms.

Despite such advancements, a major lack of gesture recognition systems that are both computationally lean and easily deployable in interactive systems such as gaming still exists. Most models either focus on attaining optimality in performance using heavier models or consider hardware-specific data (such as EMG or IMU) to be hardware-agnostic in consumer-level configurations. Therefore, this study proposes a novel hand gesture recognition platform aimed at interactive game control. The approach takes advantage of the MediaPipe hands framework for real-time landmark detection with efficiency optimization and combines it with a lightweight ANN model minimizing computational overhead and latency. The performance of the proposed ANN model is thoroughly tested and compared with state-of-the-art CNN architectures such as ResNet50 and VGG16. The comparative analysis determines the practical strengths and applicability of the ANN-based model in game applications.

2. METHOD

2.1. System architecture

Figure 1 illustrates an overview of the hand gesture recognition system being considered in this research. The system developed for controlling games consists of different steps that are essential in their own right to the correct identification and interpretation of the movements of the user. The process starts with video input, which is the primary source of information for the system. The video input may be from a webcam or another camera device capable of acquiring real-time visual depictions of the hand motion of the user. The video is necessary as it offers a continuous flow of visual data that records the dynamics and location of the hand, which is vital in sensing gestures intended for interaction with a game.

After the video input has been acquired, the process continues with processing of the video by decomposing it into frames. The individual frames are processed using the MediaPipe framework by first detecting the palm to draw a boundary around the hand area. After localizing the hand, MediaPipe applies its specialist landmark detection model to sample 21 important hand landmarks in real-time. The coordinates of the landmarks thus obtained are then converted to a systematic feature vector with maintained spatial relationships between the key points. Later, this vector serves as an input to a neural network responsible for gesture classification. Incorporating MediaPipe into the pipeline not only enhances the accuracy of feature extraction but also significantly reduces computational demands, thereby guaranteeing the viability of the system for real-time applications.

The hand skeleton input produced by MediaPipe, comprising the skeletal structure of the hand, is used as input for a CNN model. In this case, LeNet architecture—a traditional model in the field of image classification—is used to read the image and extract high-level features capturing spatial relationships between important hand landmarks. This extraction of features is crucial for the distinction between hand gestures and interpreting them as individual commands for controlling the game. The CNN then outputs a sequence of predictions that include the detected gesture and a confidence score measure of how certain the model is.

These predictions are passed as inputs to the application or game, thus enabling hands-free interaction without needing conventional input methods such as keyboards or controllers.

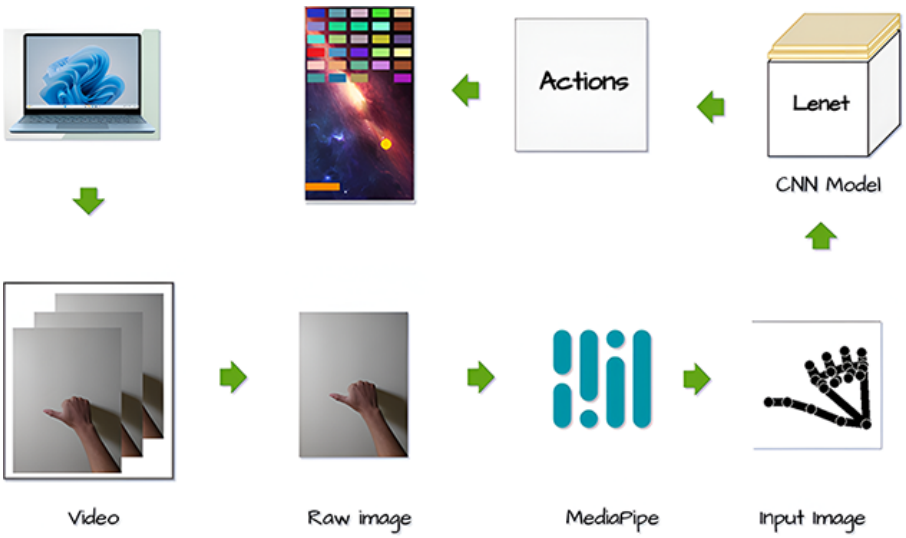


Figure 1. System overview

2.2. MediaPipe hands

MediaPipe hands [24] is an advanced framework utilized for tracking hands in real-time and landmark detection, which is important for human-computer interaction applications. It is able to localize and detect 21 key points on each hand (Figure 2), including fingertips, joints, and the palm bottom, thus enabling accurate hand pose estimation. This solution has significant applications in many areas such as gesture recognition, sign language interpretation, virtual reality (VR), and augmented reality (AR). The strong architecture of MediaPipe hands allows for seamless processing of the input data without compromising on the high level of accuracy, making it suitable for integration into real-time applications on numerous platforms.

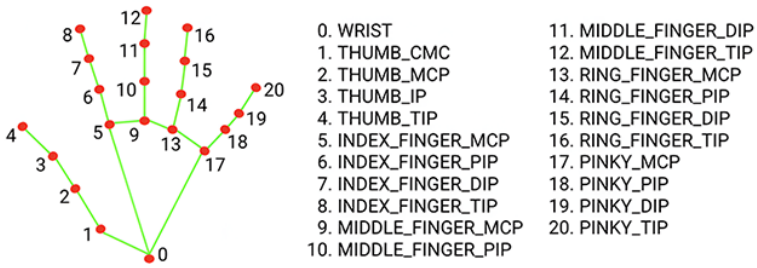


Figure 2. Hand landmarks

The MediaPipe hands solution works on a two-stage strategy that includes palm detection and hand landmark detection. In the initial stage, palm detection is applied for identifying regions of hands within the provided image. The palm detection step provides the base for stable keypoint extraction by defining a clear region for additional processing. Once the palm has been detected, the system enters the second stage: hand keypoint detection, where the 21 special keypoints on the cropped hand image are detected. This is essential in order to map the hand anatomy properly and extract significant details such as the fingers' tips, intermediate phalanges, and palm center.

One of the advantages of MediaPipe hands is that it can also track multiple hands at a time, even in cases where the hands overlap or where the hands change orientation. Multi-hand tracking capability is crucial for applications that need both hands to be involved or when there are multiple users. High tracking stability is attained by the system through the utilization of context information from successive frames to

predict keypoint positions even under conditions of rapid hand movement or temporary occlusion. Predictive style tracking enables smooth and continuous tracking required by applications demanding responsiveness, such as virtual reality/augmented reality interaction and gesture games.

Based on keypoints' coordinates identified by MediaPipe hands, it is possible to build a formal input for an artificial neural network. Namely, one-dimensional input vector can be built where every element corresponds to the Euclidean distance from the WRIST point to the remaining 20 keypoints. This method guarantees that input data preserves spatial relations among significant landmarks on the hand while minimizing complexity related to direct coordinate representation. The computation of these distances produces a normalized and invariant set of input features less sensitive to variation in hand size or orientation, therefore improving the robustness of the neural network model during training and inference phases. This then yields a feature vector as: $X = \{d_1, d_2, d_3, ..., d_{20}\}$. The Euclidean distance (d_i) between the WRIST point and the other 20 keypoints is calculated using (2.2.). In this equation, $i = 1, 2, \ldots, 20$, (x_0, y_0, z_0) are the coordinates of the WRIST point, and (x_i, y_i, z_i) represent the coordinates of the other keypoints.

$$d_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2} \tag{1}$$

By representing the input in such a manner, the output vector contains 20 elements which accurately represent the spatial relation of the anatomy of the hand. This vector is used as a significant feature for the neural network so that it can analyze and learn patterns of various hand gestures or motion. The Euclidean distance calculation guarantees that each vector element will be scaled equally, hence contributing to stabilization of the learning process and enhancement of the model's performance. As such, this structured representation not only reduces the complexity of the input data but also retains the critical geometric characteristics required for precise hand movement recognition. This method illustrates an effective way of converting raw landmark data into a meaningful format suitable for deep learning algorithms, hence enabling innovation in gesture-based interactive systems.

2.3. Artificial neural network model

After the extraction and vectorization of the 21 keypoints via MediaPipe hands, there is a generation of a structured feature vector comprising 20 distinct features. Each entry in the vector is the Euclidean distance between the WRIST keypoint and all the other keypoints and some other derived features. Figure 3 illustrates an ANN model that has three fully connected layers. There is a first hidden layer with 64 neurons and ReLU activation, followed by a hidden layer with 32 neurons and ReLU. The output layer has 4 neurons, one for each gesture class, and applies softmax to generate class probabilities. The ANN model is trained using the Adam optimizer with a learning rate of 0.001 for 20 epochs to achieve high recognition accuracy and low computation needs. This lightweight design and efficient training routine render the ANN model particularly amenable to real-time hand gesture recognition on the move, specifically for interactive game applications.

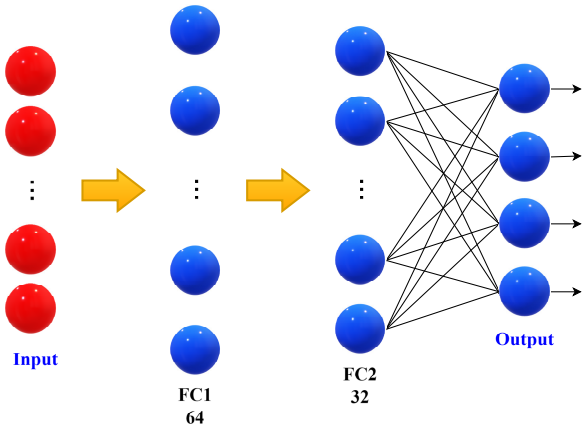


Figure 3. The structure of ANN model

2.4. The ResNet-50 model

The deep CNN architecture known as ResNet-50 (Figure 4), also known as ResNet-50, has emerged as an essential component in the field of contemporary computer vision. He *et al.* [25] presented ResNet-50 with the intention of addressing the obstacles that are associated with training very deep networks. In particular, the degradation problem is a problem that arises when increasing the depth of the network results in decreased accuracy owing to difficulty in optimizing the network. One of the most important innovations that ResNet-50 brings to the table is its utilization of residual learning through shortcut connections. This enables the network to acquire identity mappings and helps to alleviate the problem of disappearing gradients. In order to demonstrate its adaptability and efficiency, this architecture has been utilized extensively in a variety of applications, including semantic segmentation, object identification, and picture classification.

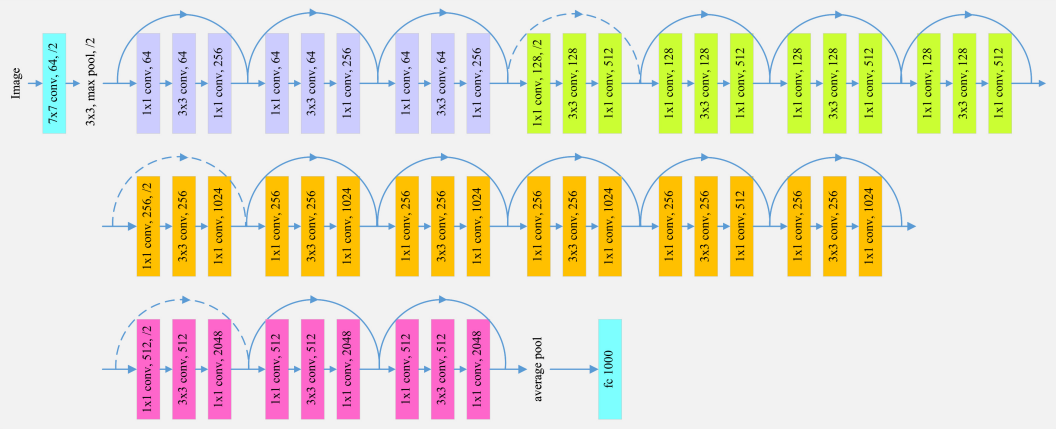


Figure 4. The ResNet-50 model architecture

In the ResNet-50 architecture, there are a total of fifty layers, which include convolutional layers, batch normalization layers, ReLU activation functions, and fully linked layers. The usage of residual blocks, in which identity connections bypass one or more layers, is one of its distinguishing characteristics. This allows the network to learn residual functions rather than direct mappings, which is a significant advantage. Each of the sixteen residual blocks that make up ResNet-50 is composed of three convolutional layers: a 1×1 convolution for dimensionality reduction, a 3×3 convolution for spatial feature extraction, and another 1×1 convolution for restoring dimensionality. These blocks are constructed utilizing bottleneck designs. This approach to bottlenecks decreases the load of computing work while retaining the capacity for representation. Additionally, the design employs strided convolutions and pooling layers to gradually lower the spatial dimensions, which guarantees the capture of hierarchical information across a variety of levels.

2.5. The VGG-16 model

Simonyan and Zisserman [26] initially presented the CNN architecture known as VGG-16. The architecture places an emphasis on having a basic and modular style. By taking this method, the network is able to extract intricate hierarchical properties while preserving its computational efficiency. The VGG-16 architecture (Figure 5) consists of 16 weight layers, including 13 convolutional layers and 3 fully connected layers, interspersed with max-pooling and activation functions. The hallmark of VGG-16 is its use of small 3×3 convolutions with a stride of 1 and padding to maintain spatial resolution. Stacking these tiny kernels in sequence allows the network to simulate the receptive field of bigger filters, which in turn enables the network to collect more detailed spatial data. Max-pooling layers separate five convolutional blocks in a hierarchical design of the architecture. This allows for the gradual reduction of spatial dimensions while simultaneously increasing the depth of feature maps. The fully linked layers at the very end of the network are responsible for aggregating these features in order to arrive at a final categorization. The consistent architecture and depth of VGG-16 make it an excellent choice for feature extraction and transfer learning, despite the fact that it has rather high processing requirements.

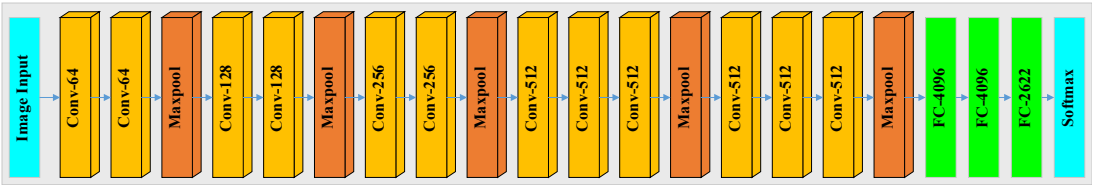


Figure 5. The VGG-16 model architecture

3. RESULTS AND DISCUSSION

3.1. Game application design

Game description:

Bricks, balls, and boards will be the three components that make up this game. We will arrange the bricks in rows at the very top of the screen. The bricks will vanish each time the ball makes contact with them. Any item that the ball comes into contact with will cause it to go in the opposite direction. Users can block the ball using the left or right board controls. If there are no bricks, the player is considered to have won the game; if they are unable to stop the ball, they will lose and the game will end.

Game activity diagram:

Figure 6 illustrates the game activity diagram. An initial user interface is presented to players at the beginning of the game, from which they can select various choices such as "Start" and "Exit." After selecting the "Start" option, the software will transition to the game interface and begin the process of initializing all of the essential components. These components include the board, the ball, and a set of bricks that are organized in a pattern that has been planned out beforehand. Other variables, including as the score, the velocity of the ball, and the status of the game, are also initialized in order to guarantee a seamless gameplay experience. It is the player's responsibility to manage the board, which is moved horizontally in order to interact with the ball, which is constantly traveling across the screen.

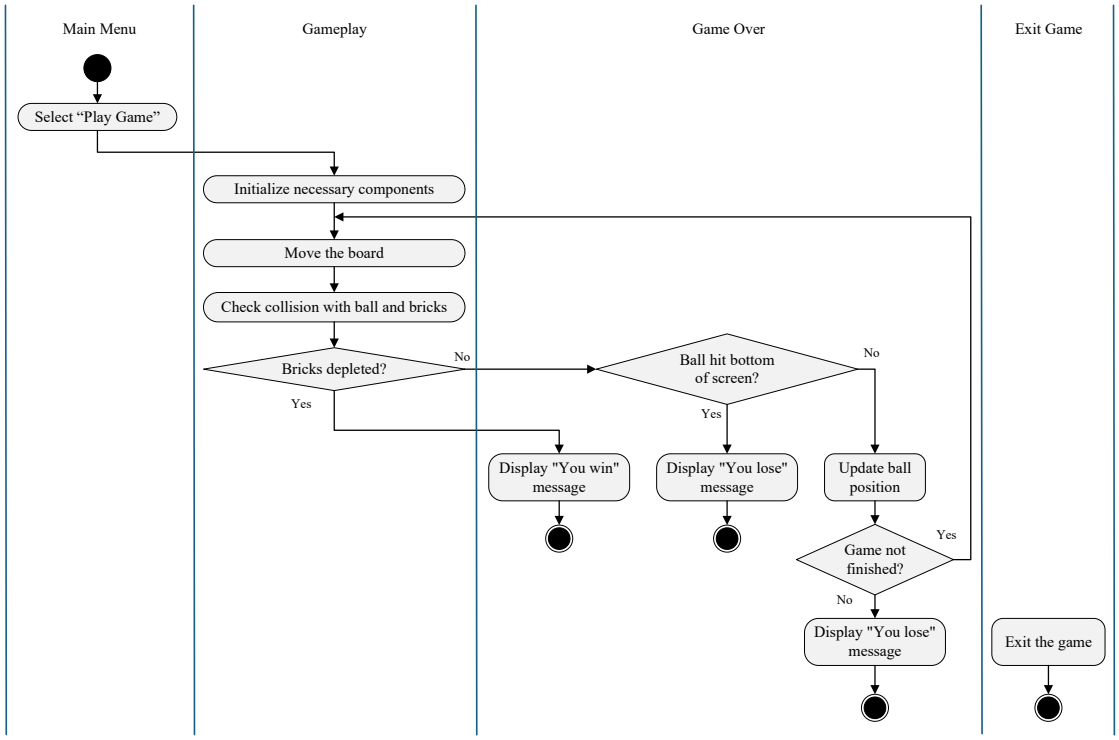


Figure 6. Game activity diagram

The program is responsible for managing the motion of the ball and ensuring that it does not collide with the board, bricks, or walls while the game is being played. If the ball comes into contact with a brick, the brick will be demolished, and the score will be adjusted accordingly. In the event that the ball collides with the board or the walls, it will bounce back, so preserving the flow of the game. The game will continue until either all of the bricks are demolished, which would result in a victory, or the ball falls past the board, which would result in a negative outcome. After the finish of the game, a message that reads "You win" or "You lose" is displayed, depending on the outcome of the game.

Once the game has come to a conclusion, players have the choice to select the "Restart" option, which will either allow them to restart the game and play it again or stop the game altogether. A smooth gameplay experience is ensured by its straightforward yet captivating framework, which strikes a balance between interactive features and clear end conditions in order to keep players interested.

3.2. Control signal transmission from hand gesture recognition program to game application

One method to successfully convey control signals from recognition software to a gaming application is the utilization of sockets. Sockets, a reliable and frequently used technique, accomplish signal transmission in networking and operating systems. A program can create a socket and establish a connection with a corresponding socket in another program. After establishing a connection, the program that is delivering data can send data over the socket, while the program that is receiving the data can process the data that is coming in. It is possible to transmit signals using this technology across both local area networks (LANs) and the internet, which provides flexibility in deployment.

User datagram protocol (UDP) and transmission control protocol (TCP) are the two principal communication protocols that sockets are able to implement. The TCP ensures precise and sequential transmission of data packets. This feature enables applications like file transfers and protocols like HTTP and FTP to utilize it effectively. In contrast, UDP is unstable and does not require a connection. As a result, it provides lower latency and higher velocity. It is an excellent choice for applications such as online gaming, streaming multimedia, and DNS queries.

The UDP is often the protocol of choice for gaming applications that place a high priority on low-latency signal transfer. Its capacity to provide signals with low latency counterbalances its lack of dependability, ensuring a smoother and more responsive gaming experience. Therefore, the UDP is utilized in this study for the purpose of controlling the transmission of signals from the hand gesture detection program to the gaming application, as shown in Figure 7.

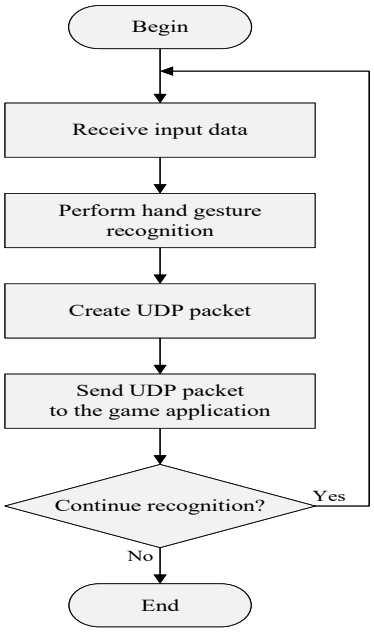


Figure 7. Diagram of control signal transmission from hand gesture recognition program to game application

3.3. Dataset description

Training dataset: the training dataset comprised 4,000 images that were labeled and categorized into four broad categories: thumbs-up, thumb-pointing-left, thumb-pointing-right, and a catch-all category for other gestures of the hand. Each gesture is associated with a distinct control signal within the respective gaming app, thereby enabling the user to control through gestures. The data has been separated into two segments: 70% for training and 30% for validation, thereby enabling the model's efficacy to be thoroughly verified. The training dataset was compiled from videos captured under varying conditions, including imaging viewpoints, lighting levels, and background environments, with the aim of increasing the model's generalization capacity. For efficient labeling and to reduce ambiguity, the videos were arranged in a way that each frame contained one clear and distinct hand gesture. The detailed breakdown of the number of images per class of hand gesture is outlined in Table 1, highlighting the balance and distribution of the dataset. Figure 8 provides representative images that show the four hand gestures, Figure 8(a) Thumbs up, Figure 8(b) Thumbs pointing left, Figure 8(c) Thumbs pointing right, and Figure 8(d) Other hand gestures.

Table 1. Description of the training dataset

Hand gestures	Control signal in game application	Training dataset	Testing dataset	Total
Thumbs-up	Start the game	700	300	1,000
Thumb-pointing-left	The board moves to the left	700	300	1,000
Thumb-pointing-right	The board moves to the right	700	300	1,000
Other hand gestures	None	700	300	1,000

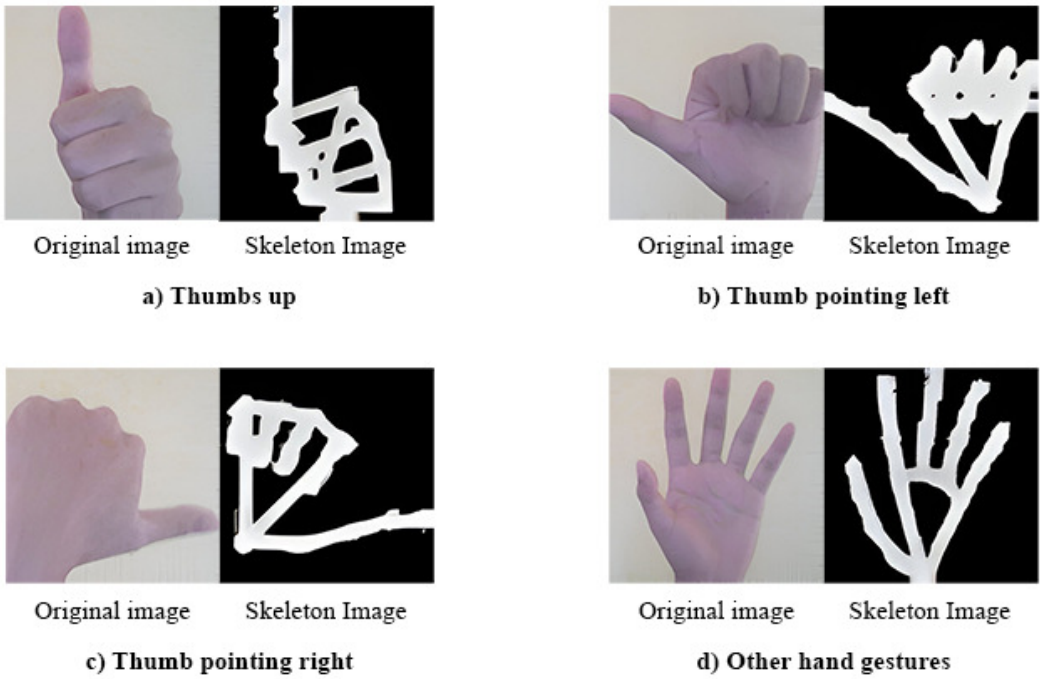


Figure 8. Snapshots of four hand gestures from the dataset, (a) Thumbs up, (b) Thumbs pointing left, (c) Thumbs pointing right, and (d) Other hand gestures

A few preprocessing procedures were carried out prior to inputting the data into deep learning models for enhancing data integrity and model strength. For consistency across the dataset, every raw image was resized to a uniform size of 224×224 pixels. Then, to normalize the input features, the pixel intensity values were normalized within the range [0, 1]. Throughout the training, we employed a range of data augmentation techniques, such as random rotation, horizontal flip, changes in brightness, and subtle zoom adjustments. All these augmentation techniques not only introduce variety into the training data but also counteract overfitting and thus enhance the model's capability to generalize to new, unseen data in real-time gesture recognition applications.

3.4. Model training evaluation

Immediately following the collection of a significant dataset, the deep neural network model was trained by extracting features from the dataset. This activity is crucial because it has a direct impact on the overall quality of the system that is being offered. This section presents the results of model training, including VGG16, ResNet50, and ANN models.

Figures 9, 10, and 11 illustrate the training and validation accuracy (left) and the training and validation loss (right) during the training process of three models. The training results of the VGG16 model are shown in Figure 9. In the accuracy graph, both training and validation accuracy grow at a quick rate in the early epochs. After fifteen epochs, the accuracy stabilizes at high values that are near to 1.0, which indicates that learning is taking place effectively. The accuracy of the training varies slightly, but it converges in a consistent manner with the accuracy of the validation. At the end of twenty epochs, the levels of accuracy for training and validation were 0.9996 and 0.9994, respectively. On the loss graph, the training loss decreases sharply within the first few epochs and then stabilizes near zero. On the other hand, the validation loss follows a similar pattern with minor spikes, which demonstrates that the model generalizes well on the validation set.

Figure 10 shows the outcomes of the training process for the ResNet50 model. According to the accuracy graph, the levels of accuracy achieved by the model during training and validation are increasing with each epoch. After fifteen epochs, these accuracy levels remained continuously high and reached a saturation level. At the end of twenty epochs, the level of accuracy achieved throughout training and validation is equal to 1.0. During the first few epochs, the loss graph demonstrates a significant decrease in training and validation loss, which is then followed by a stability that is somewhat close to zero.

Figure 11 illustrates the training results of the ANN model. In the accuracy plot, both training and validation accuracy improve rapidly in the early epochs, reaching near-perfect values close to 1.0. The validation accuracy closely tracks the training accuracy, indicating effective learning. At the end of twenty epochs, the levels of accuracy for training and validation were 0.9992 and 0.9991, respectively. During the first few epochs, the training loss experienced a substantial decrease, and it eventually stabilized close to zero in the loss plot. A similar pattern may be seen in the validation loss. This set of findings demonstrates that the model is capable of generalizing well and maintaining steady performance throughout the training phase.

Although training and validation accuracy is high, as desired, in all three models, the limitations and potential failure causes in real-world applications need to be stated. During testing, the ANN model occasionally misclassified gestures when hands were partially occluded or under low lighting conditions, which affected the quality of the MediaPipe landmark detection. Moreover, gestures with similar shapes, such as a loosely held fist or a half-extended thumb, sometimes confused with the "thumbs-up" class. These challenges suggest the need for more robustness tests in different and uncontrolled environments.

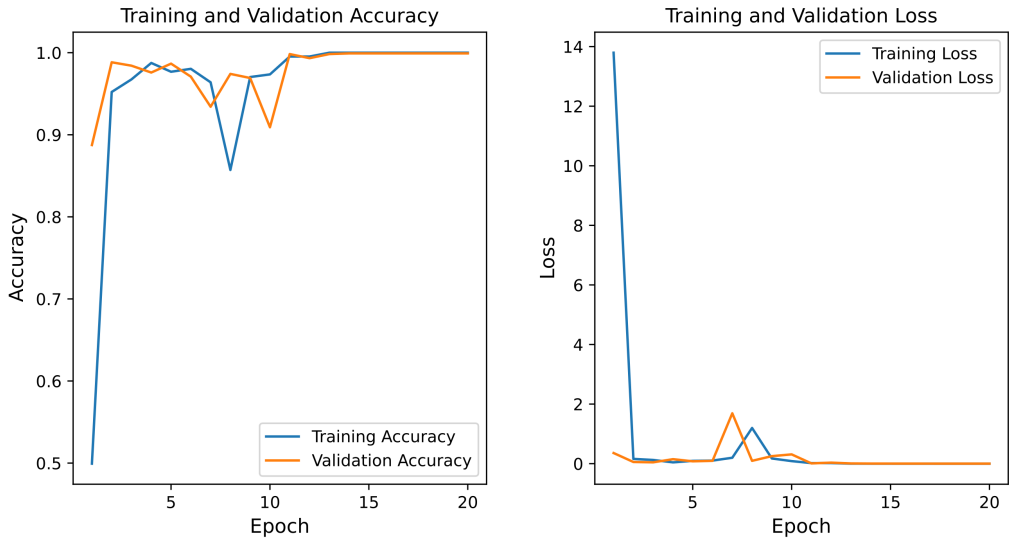


Figure 9. Accuracy and loss of the VGG16 model during training

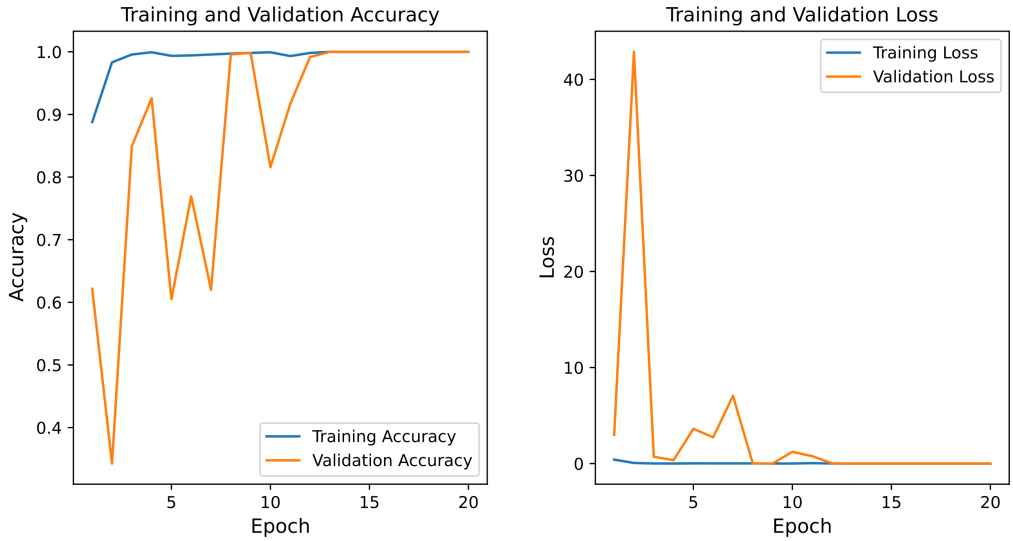


Figure 10. Accuracy and loss of the ResNet50 model during training

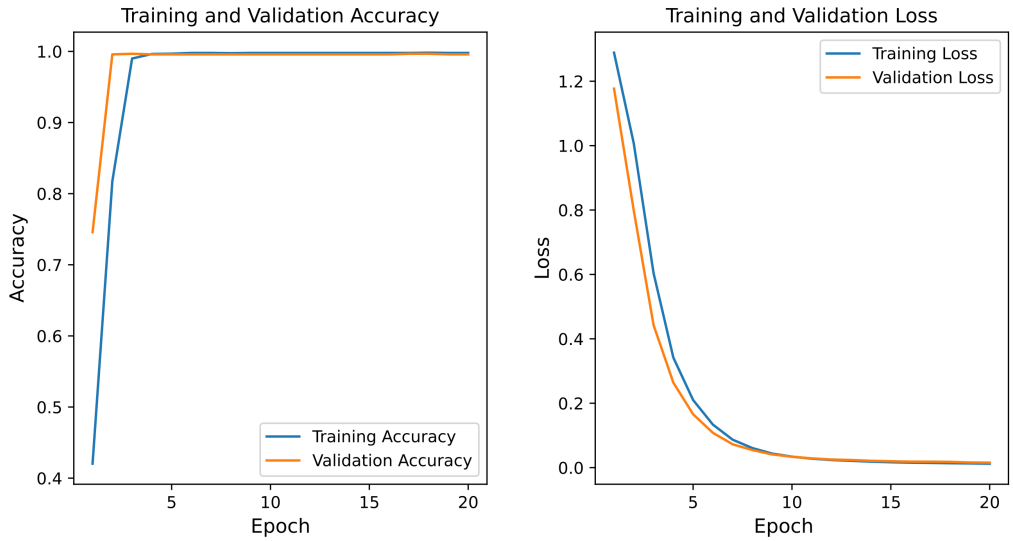


Figure 11. Accuracy and loss of the ANN model during training

3.5. Compare three models

Figure 12 presents an overall comparison of the three models (ANN, VGG16, and ResNet50) against key performance metrics such as recognition accuracy, number of parameters, and complexity of the model. The ANN model achieved 99.92% with only 3,556 parameters, whereas VGG16 and ResNet50 achieved 99.96% and 100% accuracy with 27,692,612 and 75,100,804 parameters, respectively. These results clearly show that, despite having a simpler structure, the ANN model performs as well as more complex models. To help explore these differences, we've added a bar chart comparing the accuracy and number of parameters for each model, illustrating the applicability of the ANN model where computational efficiency is most critical.

The ANN model, while being simple, was quite effective and compared to more complex models was equally accurate but with much less parameters and computational power. These qualities make it a highly viable candidate for use in edge devices or embedded systems where there is limited computational capability. Such a trade-off in terms of performance and efficiency enables the proposed solution to be realistic and usable in real-time gesture control systems, especially in mobile gaming or assistive technology.

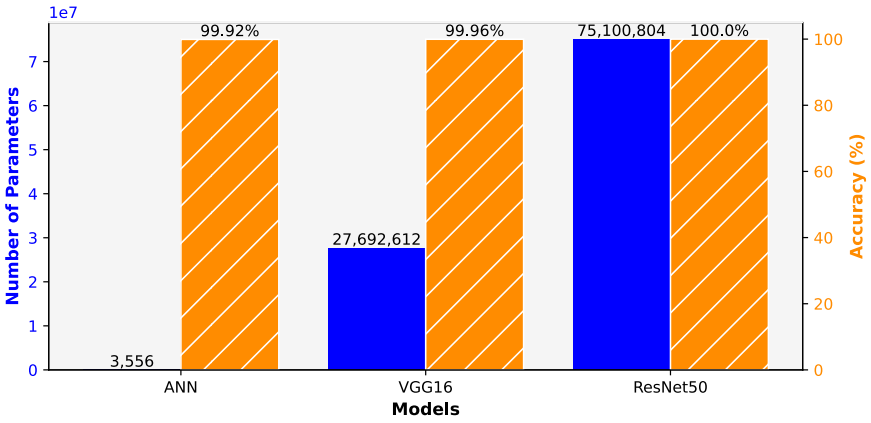


Figure 12. Results comparison of parameters and accuracy of three models

3.6. Integrating hand gesture recognition into game applications

Through the use of standard protocols or communication interfaces, the gesture recognition software that is written in Python may be easily included into applications that were created using other programming languages. Network communication standards like TCP/IP or HTTP, as well as software-specific protocols like RPC, might be included in this category of protocols. Integration of this kind makes it possible for systems to communicate with one another and makes it easier for them to share data.

In order to integrate Python-based gesture detection software into gaming applications, an application programming interface (API) may be written in Python. This API will enable data transmission and reception over conventional protocols. It is possible for the gaming application to communicate with the API by utilizing the libraries or tools that are associated with the programming language. This will make the processing of gesture recognition data easier, allowing for seamless integration.

Figures 13, 14, and 15 illustrate the control of the game application using hand gesture recognition. The initiation of the game is triggered by the player performing a thumbs-up gesture, as illustrated in Figure 13. During gameplay, the player interacts with the system to control the board's movement. To catch the ball on the left side, the player performs a thumb-pointing-left gesture, prompting the board to move left, as depicted in Figure 14. Similarly, catching the ball on the right side requires the player to execute a thumb-pointing-right gesture, which directs the board to the right, as shown in Figure 15. These gestures enable intuitive and responsive control within the game environment.

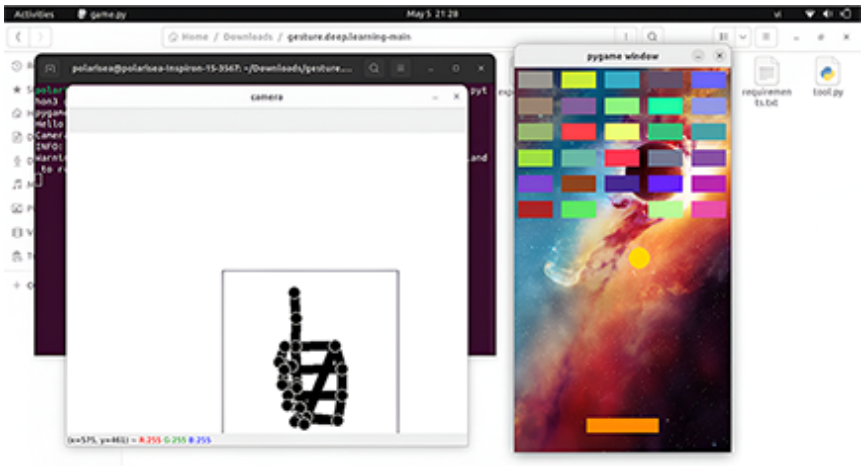


Figure 13. Start the game with the thumb-up gesture

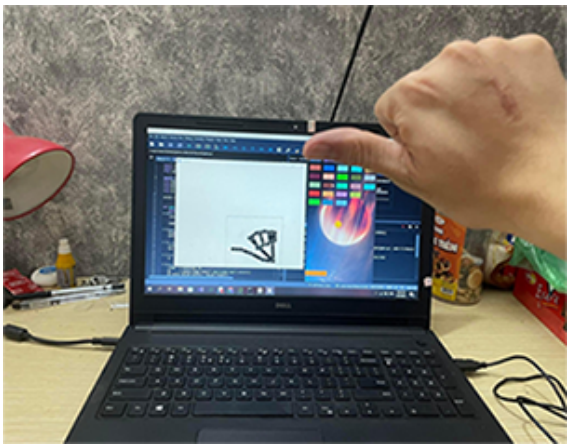


Figure 14. The board moves to the left with the thumb-pointing-left gesture

Figure 15. The board moves to the right with the thumb-pointing-right gesture

Overall, a straightforward training network model is used in the ANN model, which results in a high level of accuracy. It has achieved minimal reaction latency and quick recognition speed in order to be able to integrate into real-time control applications. This was accomplished by integrating the hand gesture recognition software into the gaming application where it was used.

Future integration plans can delve into expanding the gestural repertoire from the present four static gestures to composite or dynamic gestures. Furthermore, the system can be enhanced by adding multi-modal inputs, i.e., verbal speech or facial recognition, thereby enhancing the flexibility and usability of interactions. These innovations can enable greater immersion in various applications, including virtual reality, home automation control, and human-robot interactions.

3.7. The evaluation of processing times

Processing time significantly impacts system performance; hence, it is paramount for a system to achieve a good balance between accuracy and responsiveness. The comparative analysis of processing times of the three models—ANN, VGG16, and ResNet50—when deployed in a gaming application reveals that ANN is the most suitable option for real-time applications. In Figure 16, with a very low processing time of 0.26 milliseconds per frame, ANN is best placed to enable smooth gameplay and instantaneous responsiveness. In contrast, VGG16 and ResNet50 with their processing speeds of 0.71 milliseconds per frame and 3.7 milliseconds per frame, respectively, may introduce increased latency, affecting overall game performance. The simplicity of the ANN model also ensures that computational complexity is low, and thus deployment in low-resource gaming setups becomes extremely convenient. These results show that ANN is the most suitable model for developing games with a good mix of efficiency, real-time response, and ease of use.

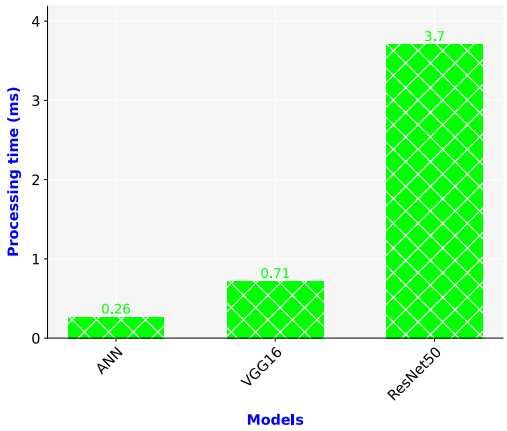


Figure 16. Compare results of the system's processing time

3.8. Discussion

The experimental results show that the suggested hand gesture recognition system, which consists of MediaPipe and a light-weight ANN model, achieves a good balance between computation cost and recognition accuracy. Even more complicated models such as ResNet50 and VGG16 have been found to have a bit better recognition accuracy compared to ANN, but this is incremental at a great computational cost. The ANN with merely 3,556 parameters achieves a high accuracy of 99.92% and a low processing time of 0.26 milliseconds per frame, making it well-suited for real-time applications like gaming. Our method is superior to other similar research that uses larger designs or technology, offering a real-world, low-latency solution that can be run on mid-range devices. The fact that the system can be easily integrated into game programs using UDP makes the system more desirable for interactive systems. Nevertheless, the present model is capable of identifying only four distinct gestures, which may limit the use of the system for more sophisticated interaction. Increase in the dataset and addition of other gesture types can render the system more versatile. Also, testing the system with a larger user sample would enable better determination of its robustness to different hand geometries and skin tones. In general, the results are in favor of the viability of employing light weight deep learning models in enabling real-time, gesture-based interaction that is natural in interactive systems.

4. CONCLUSION

This paper presents a real-time deep learning-based optimized hand gesture recognition system. By leveraging MediaPipe for efficient hand landmark extraction and a lightweight ANN for classification, our system achieves high recognition accuracy (99.92%) at significantly reduced computational complexity. In comparison to more complex architectures such as VGG16 and ResNet50, the ANN model has comparable performance with many fewer parameters and a lower processing time of 0.26 milliseconds per frame. Hence, the design is particularly well-suited for use in low-resource settings, for example, mobile phones, embedded systems, or assistive technology. The system discussed was integrated into a basic interactive game with seamless and natural gesture control using no physical input devices. These results point out the applicability of this research to real-time human-computer interaction applications, particularly for those applications where fast, natural, and touchless input is crucial.

In future work, we aim to enhance the present system in a number of ways. First, the database will be expanded to include a larger set of both static and dynamic hand gestures, captured in a wide range of real-world settings such as low illumination, noisy backgrounds, and various hand shapes. This should enhance the robustness and generalizability of the model. Second, we aim to fuse multi-modal inputs, e.g., speech commands and head pose, to allow for more flexible and richer user experiences. Further, large-scale experiments with varied users and environments will be carried out in order to compare performance under realistic conditions. Lastly, failure cases seen during testing—e.g., occlusion-based misclassifications or ambiguous gestures—will be tackled by increasing data variability and tuning the neural network structure. These future developments aim to push the frontier of real-time gesture recognition systems to more generalized use in gaming, virtual reality, robotics, and home automation.

ACKNOWLEDGEMENTS

This work was supported by the Science and Technology Development Fund of Thai Nguyen province, Vietnam (Grant Number: DT/KTCN/18/2023).

FUNDING INFORMATION

Authors state no funding involved.

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] M. J. Cheok, Z. Omar, and M. H. Jaward, "A review of hand gesture and sign language recognition techniques," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 1, pp. 131–153, Jan. 2019, doi: 10.1007/s13042-017-0705-5.

[2] O. K. Oyedotun and A. Khashman, "Deep learning in vision-based static hand gesture recognition," *Neural Computing and Applications*, vol. 28, no. 12, pp. 3941–3951, Dec. 2017, doi: 10.1007/s00521-016-2294-8.

[3] J. Singha, A. Roy, and R. H. Laskar, "Dynamic hand gesture recognition using vision-based approach for human–computer interaction," *Neural Computing and Applications*, vol. 29, no. 4, pp. 1129–1141, Feb. 2018, doi: 10.1007/s00521-016-2525-z.

[4] S. Sharma and S. Singh, "Vision-based hand gesture recognition using deep learning for the interpretation of sign language," *Expert Systems with Applications*, vol. 182, pp. 1–12, Jul. 2021, doi: 10.1016/j.eswa.2021.115657.

[5] S. Franceschini, M. Ambrosanio, S. Vitale, F. Baselice, A. Gifuni, G. Grassini, and V. Pascazio, "Hand gesture recognition via radar sensors and convolutional neural networks," in *IEEE Radar Conference (RadarConf20)*, Florence, Italy, Sep. 2020, pp. 1–5, doi: 10.1109/RadarConf2043947.2020.9266565.

[6] S. Ahmed, K. D. Kallu, S. Ahmed, and S. H. Cho, "Hand gestures recognition using radar sensors for human-computer-interaction: a review," *Remote Sensing*, vol. 13, no. 3, pp. 1–24, Feb. 2021, doi: 10.3390/rs13030527.

[7] J.-H. Kim, G.-S. Hong, B.-G. Kim, and D. P. Dogra, "Deepgesture: deep learning-based gesture recognition scheme using motion sensors," *Displays*, vol. 55, pp. 38–45, Dec. 2018, doi: 10.1016/j.displa.2018.08.001.

[8] A. A. Q. Mohammed, J. Lv, and M. S. Islam, "A deep learning-based end-to-end composite system for hand detection and gesture recognition," *Sensors*, vol. 19, no. 23, pp. 1–23, Nov. 2019, doi: 10.3390/s19235282.

[9] G. Li, H. Tang, Y. Sun, J. Kong, G. Jiang, D. Jiang, B. Tao, S. Xu, and H. Liu, "Hand gesture recognition based on convolution neural network," *Cluster Computing*, vol. 22, no. 2, pp. 2719–2729, Mar. 2019, doi: 10.1007/s10586-017-1435-x.

[10] G. Devineau, F. Moutarde, W. Xi, and J. Yang, "Deep learning for hand gesture recognition on skeletal data," in *13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, Xi'an, China, May 2018, pp. 106–113, doi: 10.1109/FG.2018.00025.

[11] Z. Zhang, K. Yang, J. Qian, and L. Zhang, "Real-time surface EMG pattern recognition for hand gestures based on an artificial neural network," *Sensors*, vol. 19, no. 14, pp. 1–15, Jul. 2019, doi: 10.3390/s19143170.

[12] N. Nasri, S. Orts-Escolano, and M. Cazorla, "An sEMG-Controlled 3D game for rehabilitation therapies: real-time time hand gesture recognition using deep learning techniques," *Sensors*, vol. 20, no. 22, pp. 1–12, Nov. 2020, doi: 10.3390/s20226451.

[13] M. A. Ozdemir, D. H. Kisa, O. Guren, A. Onan, and A. Akan, "EMG based hand gesture recognition using deep learning," in *Medical Technologies Congress (TIPTEKNO)*, Antalya, T¨urkiye, Nov. 2020, pp. 1–4, doi: 10.1109/TIPTEKNO50054.2020.9299264.

[14] P. J. Cruz, J. P. Vasconez, R. Romero, A. Chico, M. E. Benalcazar, R. Alvarez, L. I. B. Lopez, and A. L. V. Caraguay, "A Deep Q-network based hand gesture recognition system for control of robotic platforms," *Scientific Reports*, vol. 13, no. 1, pp. 1–18, May 2023, doi: 10.1038/s41598-023-34540-x.

[15] A. Mujahid, M. J. Awan, A. Yasin, M. A. Mohammed, R. Damaevicius, R. Maskeliunas, and K. H. Abdulkareem, "Real-time hand gesture recognition based on deep learning YOLOv3 model," *Applied Sciences*, vol. 11, no. 9, pp. 1–15, May 2021, doi: 10.3390/app11094164.

[16] K. Aggarwal and A. Arora, "Hand gesture recognition for real-time game play using background elimination and deep convolution neural network," in *Virtual and Augmented Reality for Automobile Industry: Innovation Vision and Applications. Cham*, Switzerland: Springer International Publishing, Feb. 2022, pp. 145–160, doi: 10.1007/978-3-030-94102-4 8.

[17] M. Lee and J. Bae, "Deep learning based real-time recognition of dynamic finger gestures using a data glove," *IEEE Access*, vol. 8, pp. 219 923–219 933, Nov. 2020, doi: 10.1109/ACCESS.2020.3039401.

[18] A. Sen, T. K. Mishra, and R. Dash, "Deep learning-based hand gesture recognition system and design of a human–machine interface," *Neural Processing Letters*, vol. 55, no. 9, pp. 12 569–12 596, Dec. 2023, doi: 10.1007/s11063-023-11433-8.

[19] N. Osama, Y. Ahmed, H. Mohamed, S. E. Hesham, Y. Ahmed, E. K. Elsayed, and D. Ezzat, "Virtual control system for presentations by real-time hand gesture recognition based on machine learning," in *9th International Conference on Advanced Intelligent Systems and Informatics*, Cairo, Egypt, Sep. 2023, pp. 327–335, doi: 10.1007/978-3-031-43247-7 29.

[20] L. Guo, Z. Lu, and L. Yao, "Human-machine interaction sensing technology based on hand gesture recognition: a review," *IEEE Transactions on Human-Machine Systems*, vol. 51, no. 4, pp. 300–309, Jun. 2021, doi: 10.1109/THMS.2021.3086003.

[21] S. Jiang, P. Kang, X. Song, B. P. Lo, and P. B. Shull, "Emerging wearable interfaces and algorithms for hand gesture recognition: a survey," *IEEE Reviews in Biomedical Engineering*, vol. 15, pp. 85–102, May 2021, doi: 10.1109/RBME.2021.3078190.

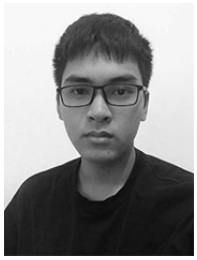
[22] F. Naseer, G. Ullah, M. A. Siddiqui, M. Jawad Khan, K.-S. Hong, and N. Naseer, "Deep learning-based unmanned aerial vehicle control with hand gesture and computer vision," in *13th Asian Control Conference (ASCC)*, Jeju, Korea, May 2022, pp. 1–6, doi: 10.23919/ASCC56756.2022.9828347.
[23] H. Wen, Y. Xu, L. Li, X. Ru, X. Wang, and Z. Wu, "Enhancing sign language teaching: a mixed reality approach for immersive learning and multi-dimensional feedback," *arXiv:2404.10490*, pp. 1–6, May 2024, doi: 10.48550/arXiv.2404.10490.
[24] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "MediaPipe hands: on-device real-time hand tracking," in *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*, Seattle, WA, USA, Jun. 2020, pp. 1–5, doi: 10.48550/arXiv.2006.10214.
[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Jun. 2016, pp. 770–778, doi: 10.1109/cvpr.2016.90.
[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR 2015)*, California, USA, May 2015, pp. 1–14, doi: 10.48550/arXiv.1409.1556.

BIOGRAPHIES OF AUTHORS

Huu-Huy Ngo received his B.S. and M.S. degrees from Thai Nguyen University of Information and Communication Technology, Vietnam, in 2010 and 2012, respectively, and Ph.D. degrees in Information Engineering and Computer Science from Feng Chia University, Taiwan, in 2021. Currently, he is a lecturer at the Thai Nguyen University of Information and Communication Technology, Vietnam. His research interests include computer vision, deep learning, embedded system, neural networks, and object detection. He can be contacted at email: nhhuy@ictu.edu.vn.

Hung Linh Le received his B.S. and M.S. degrees from VNU University of Engineering and Technology, Vietnam, in 2003 and 2007, respectively, and Ph.D. degrees in Control Engineering and Automation from Vietnam Academy of Science and Technology, Vietnam, in 2016. Currently, he is a lecturer at the Thai Nguyen University of Information and Communication Technology, Vietnam. His research interests include measurement system design, control system, embedded system, neural networks, and multimedia applications. He can be contacted at email: lhlinh@ictu.edu.vn.

Man Ba Tuyen is an undergraduate student in the Faculty of Information Technology at Thai Nguyen University of Information and Communication Technology, Vietnam. His research interests include computer vision, deep learning, human activity recognition, neural networks, and object detection. He can be contacted at email: dtc2054802010571@ictu.edu.vn.



Vu Dinh Dung received his B.S. from Thai Nguyen University of Information and Communication Technology, Vietnam, in 2024. His research interests include computer vision, deep learning, machine translation, natural language processing and object detection. He can be contacted at email: dtc1954801010001@ictu.edu.vn.

Tran Xuan Thanh received his B.S. from Thai Nguyen University of Information and Communication Technology, Vietnam, in 2024. His research interests include computer vision, deep learning, machine translation, natural language processing and object detection. He can be contacted at email: dtc1954801010007@ictu.edu.vn.