Gesture control: real-time 3D hand recognition for touchless interfaces

Vasyl Teslyuk[†], Iryna Kazymyra[*,†] and Volodymyr Tsapiv[†]

Lviv Polytechnic National University, 12 S. Bandera Str., 79013 Lviv, Ukraine

Abstract

This paper presents a novel framework for real-time dynamic hand gesture recognition designed to enhance interaction with smart devices and touchless interfaces. The proposed system integrates Google Mediapipe for hand pose detection with a modified version of the DD-Net architecture, optimized for online classification of gestures using 2D and 3D data. Key innovations include introducing an auxiliary classification head to address the class imbalance and an attention mechanism to improve the recognition of partially observed gestures. The system is evaluated on the NVGesture and SHREC22 datasets, achieving an accuracy of 0.784 and 0.924, respectively, surpassing previous benchmarks.

Keywords

real-time gesture recognition, human-machine interactions, dynamic hand gestures

1. Introduction

Human-computer interaction (HCI) has evolved significantly over the years, moving from traditional input methods such as keyboards and mice to more intuitive and natural modes of interaction. Hand gesture recognition is prominent among these due to its close alignment with human communication habits. Hand gestures, being an integral part of non-verbal communication, provide a seamless and intuitive way for humans to convey commands and intentions. This makes them ideal for controlling machines, particularly when hands-free, touchless interaction is required.

The relevance of real-time dynamic hand gesture recognition lies in its broad applicability across several emerging fields, including augmented reality (AR), virtual reality (VR), robotics, and smart environments. In AR and VR systems, where users are immersed in virtual spaces, traditional input devices can be cumbersome and break immersion. Gesture control offers a more natural alternative, enabling users to interact directly with virtual objects. Similarly, gesture recognition can facilitate smoother interaction between humans and machines in robotics, enabling more intuitive control in industrial or assistive contexts.

One of the most pressing applications of this technology is in the development of touchless interfaces, which have gained significant importance due to public health concerns, particularly following the COVID-19 pandemic. In public spaces like elevators, ATMs, and kiosks, touchless interaction can help reduce the spread of infectious diseases by minimizing physical contact with shared surfaces. Gesture recognition provides an ideal solution for these scenarios by allowing users to interact without direct contact, ensuring convenience and hygiene.

Despite these benefits, real-time dynamic hand gesture recognition remains a challenging task. The primary obstacles include the high computational cost of processing continuous streams of data and the need for low-latency, high-accuracy systems that can function effectively on devices with

CIAW-2024: Computational Intelligence Application Workshop, October 10-12, 2024, Lviv, Ukraine
[*] Corresponding author.
[†] These authors contributed equally.
vasyl.m.teslyuk@lpnu.ua (V. Teslyuk); volodymyr.tsapiv.mknus.2023@lpnu.ua (V. Tsapiv); iryna.y.kazymyra@lpnu.ua (I. Kazymyra)
0000-0002-5974-9310 (V. Teslyuk); 0009-0008-2420-7483 (V. Tsapiv); 0000-0003-1597-5647 (I. Kazymyra)

limited processing power, such as those equipped with CPUs only. While accurate, traditional offline gesture recognition methods are not designed to handle real-time data processing, traditional offline gesture recognition methods typically require pre-segmented sequences of hand poses for analysis. This presents a significant gap in real-time applications, which require systems to make instantaneous decisions based on continuously incoming data.

This research aims to develop a real-time 3D hand pose recognition framework that balances high accuracy with computational efficiency, making it suitable for smart devices and touchless interfaces. To achieve this, the following tasks were undertaken:

- Adapting the DD-Net architecture for efficient online inference.
- Designing a dual-head prediction system to enhance the detection of gestures and non-gestures.
- Evaluating the system on the NVGesture dataset [11] and real-time input to validate its performance.

This framework advances the field of gesture recognition by offering a practical solution for real-time, low-latency applications with wide-reaching implications for smart environments and public interfaces.

2. Related works

Dynamic hand gesture recognition has seen significant advancements, with various models addressing offline and online recognition tasks using skeletal, depth, and motion data. However, real-time gesture recognition, particularly in streaming video, presents unique challenges due to the need for low-latency processing and robustness to environmental variations.

One of the prominent approaches in hand gesture recognition is the STA-GCN model [1], which employs a two-stream graph convolutional network with spatial-temporal attention for skeleton-based hand gesture recognition. The two-stream architecture processes pose and motion streams separately, using spatial-temporal graph convolutional layers to capture hand gestures over time. This method incorporates temporal pyramid pooling to extract features across multiple time scales, enhancing the model's ability to recognize gestures. The approach was evaluated on the DHG14/28 and SHREC2017 [5] datasets, demonstrating high accuracy. Still, the complexity introduced by spatial-temporal attention can increase computational overhead, making it less suitable for real-time applications where efficiency is vital.

Another relevant work focuses on robust feature extraction from skeletal and depth data. The Robust Hand Shape Features for Dynamic Hand Gesture Recognition study [2, 14, 15, 16] proposes a multi-level feature LSTM model that extracts 3D geometric transformations from skeletal data and segmentation-based depth shape features. This method, tested on the DHG14/28 dataset, achieved state-of-the-art (SOTA) results by leveraging Conv1D and Conv2D pyramid structures with LSTM blocks. While this approach enhances accuracy, its reliance on depth and skeletal data increases computational complexity. It makes it less feasible for real-time applications where only skeletal data might be available.

The DD-Net architecture [3], designed for skeleton-based action recognition, takes a different approach by simplifying the network structure to improve efficiency. DD-Net introduces a double-motion feature extraction method that captures joint distances and global motion variations. This lightweight model was tested on SHREC17 [5] and JHMDB datasets, achieving competitive results for offline gesture classification. However, its design could be optimized for real-time continuous recognition, as it assumes the gesture's start and end are predefined, making it less adaptable to streaming data.

The challenge of continuous gesture recognition is addressed by methods presented in the SHREC 2022 Track on Online Detection of Heterogeneous Gestures [4]. The contest evaluated online recog-

nition methods where gestures are embedded in continuous sequences, interspersed with non-gestural motions. One such method is the Two-stage ST-GCN [4, 12, 13], which uses a sliding window technique to identify gesture candidates before refining them with a larger classification model. This two-stage approach improves the system's ability to handle continuous data and boundary detection between gestures and non-gestures. However, the sliding window approach can introduce latency, especially if the window size is large, impacting the real-time responsiveness of the system.

Another approach, Transformer Network + Finite State Machine (TN-FSM) [4], uses temporal convolutional networks (TCNs) and transformers to classify gestures within continuous data streams. The model includes a logical state machine that helps delineate gesture boundaries. This method excels in handling non-gesture segments and works well for continuous streams. Still, transformers and state machines introduce computational complexity, limiting their use in real-time applications unless carefully optimized.

The DeepGRU architecture [6] also contributes to this domain by employing gated recurrent units (GRUs) combined with a global attention mechanism. Designed for gesture recognition, DeepGRU achieves SOTA results on SHREC17 [5] and SHREC19 [7] datasets, offering an end-to-end approach for gesture classification from raw skeletal data. While highly accurate, the attention mechanism increases computational demand, and like many deep learning models, its performance in real-time settings depends on hardware capabilities.

Despite the advancements in accuracy and feature extraction, many of these models struggle to balance computational efficiency with real-time performance. Our work builds on these methods, particularly DD-Net and the SHREC competition models, by adapting the DD-Net architecture for online inference while integrating 3D and 2D hand pose data for robust recognition. By introducing residual connections and employing a sliding window approach optimized for low latency, our model aims to achieve real-time gesture recognition in CPU-constrained environments, ensuring accuracy and efficiency in practical applications.

3. Methods

3.1. Overview of the proposed approach

This study presents a real-time hand gesture recognition system that modifies an existing offline classification model to function online. The core model is based on the double-feature double-motion network (DD-Net) architecture, a 1D convolutional neural network (CNN) optimized for sequential data and designed originally for offline tasks. Our modifications adapt DD-Net for real-time inference, allowing it to classify continuous hand gestures while maintaining high accuracy and computational efficiency. The method works with Google Mediapipe [8] for hand pose detection and 3D hand landmark acquisition, reflecting the integration of advanced technologies observed in previous studies [9]. Also it is compatible with other tracking devices like Intel RealSense or Hololens 2.

The architecture captures temporal hand dynamics by buffering and preprocessing 3D landmarks into windows of 16 frames, making predictions continuously, and reducing spurious gestures through a post-processing step. Figure 1 presents a pipeline architecture overview.

3.2. Data acquisition and preprocessing

Data acquisition

The system employs various hardware options, such as cameras and depth sensors, to acquire 3D hand landmarks:

- Google Mediapipe provides real-time hand pose estimation, outputting 3D joint coordinates for each frame.
- Other supported sensors include Intel RealSense or Hololens 2, which provide hand pose data in formats compatible with the model.

Each captured frame consists of the 3D positions of multiple hand joints. The system buffers these into windows of 16 frames to match the training configuration of the network.

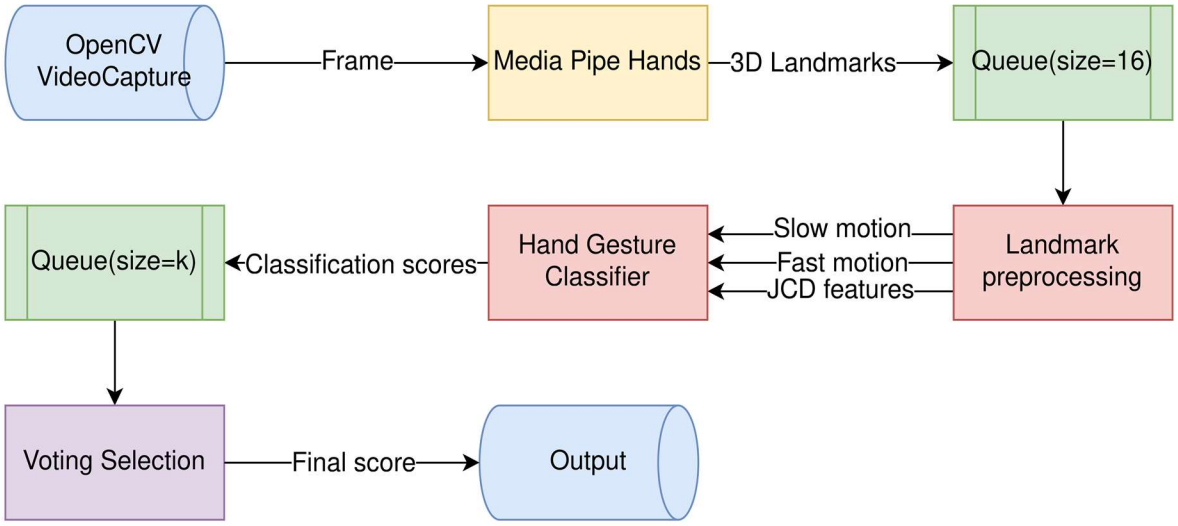

Figure 1: Overview of the pipeline architecture based on the Mediapipe hand pose estimator.

Preprocessing

To ensure robustness and compatibility across various frame rates, preprocessing steps are applied:

- Standardization: the 3D joint coordinates are standardized using the mean and standard deviation of the joint positions in the training dataset.
- Frame Rate Adjustment: an interpolation-based resampling is performed if the input frame rate differs from the training data frame rate. The system collects the required number of frames and then interpolates to ensure exactly 16 frames are used in each window. Without this, temporal inconsistencies would degrade the network's performance by introducing non-matching motion sequences.

The preprocessing step thus ensures the model receives data with consistent temporal properties, preserving gesture dynamics.

3.3. Model input representation

The input representation of hand gestures is crafted to capture both static and dynamic aspects of the movement. Each 16-frame observation window W is processed into three distinct views:

- Geometric Layout (Joint Collection Distances, JCD): This view represents the spatial relationship between hand joints by calculating the Euclidean distances between every pair of joints. The JCD features are location and viewpoint invariant, ensuring the model can recognize gestures regardless of hand orientation. The resulting tensor is of size $(\mathcal{J}\text{-}1) * \mathcal{J} / 2 \times W$, where $\mathcal{J}$ is the number of joints and W is the window size (16).
- Short-term Slow Motion (M_{slow}): This view captures short-term motion dynamics by computing the linear velocity of each joint between consecutive frames. This view reflects slow, continuous movements and is represented as a tensor of size $\mathcal{J} \times (W-1)$
- Short-term Fast Motion (M_{fast}): Similar to M_{slow}, this view computes linear velocity but skips every other frame, focusing on faster motions. This results in a tensor of size $\mathcal{J} \times (W/2-1)$. M_{fast} helps the model differentiate between quick, subtle movements and slower motions.

These three views are fed into separate embedding branches, creating a comprehensive multi-view description of the hand gesture.

3.4. Model architecture

As it was mentioned before the core model is based on the double-feature double-motion network (DD-Net) architecture. We adapt DD-Net for real-time inference, allowing it to classify continuous hand gestures while maintaining high accuracy and computational efficiency. The modified DD-Net model architecture is presented in Figure 2.

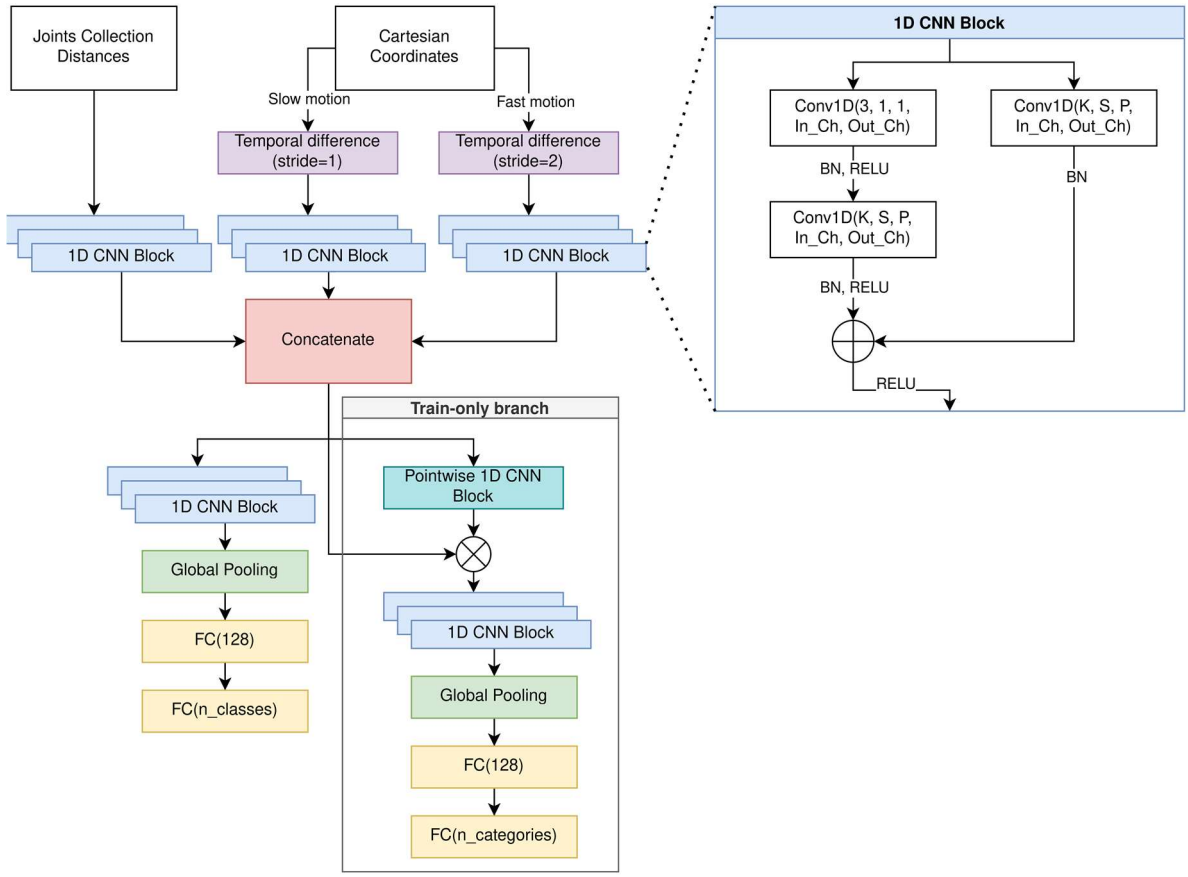

Figure 2: Overview of the modified DD-Net model architecture.

Embedding branches

Each input view (JCD, M_{slow}, M_{fast}) passes through its own embedding branch composed of several Convolutional Blocks, each containing:

- Conv1D Layers: These layers extract temporal features from the sequential data, learning to recognize patterns in hand movement over time.
- Batch Normalization: Applied to ensure stability during training by normalizing activations, reducing internal covariate shifts.
- Leaky ReLU Activation: Introduces non-linearity while avoiding the vanishing gradient problem, allowing for better gradient flow in deeper layers.

An important enhancement over the original DD-Net is the introduction of residual connections in the convolutional blocks. See Figure 2. These connections, inspired by the ResNet architecture [10], allow the network to bypass layers when beneficial, facilitating better learning and addressing the vanishing gradient problem that can occur in deep networks.

Feature concatenation and classification

After each embedding branch processes its respective input, the features are concatenated to form a unified representation of the hand gesture. This concatenated representation flows into two classification heads:

Primary Classification Head: This head performs fine-grained gesture classification, predicting specific gesture classes. It consists of several convolutional blocks followed by max-pooling layers for feature extraction and translation invariance. Afterwards, a global max-pooling layer is applied, followed by two fully connected layers and a Softmax activation to output the final gesture class.

Auxiliary Classification Head: Designed to handle the major class imbalance in online scenarios where the majority of frames contain no gestures. This head classifies gestures into three categories: static gestures, dynamic gestures, and no-gesture frames. A small attention unit enhances this branch, helping the model ignore irrelevant background movements and focus on meaningful hand gestures. This is particularly useful for frames at the start and end of gestures, where only part of the gesture is in the window. The attention unit uses pointwise convolution to generate a soft attention mask multiplied by the concatenated embeddings to focus on relevant features selectively.

3.5. Model training

The dataset used for training is adapted to the sliding window format required by the network. For each frame t, the system collects joint data from frames $t-W+1$ to t, creating a 16-frame observation window. The corresponding label is the gesture class at frame t.

The primary classification head is trained to predict the fine-grained gesture class using a cross-entropy loss. Using a separate cross-entropy loss, the auxiliary classification head is trained on the simplified three-class task (static, dynamic, no-gesture).

Both classification heads are optimized simultaneously, with cross-entropy loss applied to each. The model is trained using the Adam optimizer with a learning rate schedule that decays over time to ensure convergence.

3.6. Model inference

During inference, the system processes each frame in real time, updating its prediction every 16 frames using the following steps:

1. Observation Window Sampling: For each frame t, a 16-frame observation window W is created. The 3D joint data is preprocessed, and the multi-view description (JCD, M_{slow}, M_{fast}) is extracted.
2. Preliminary Classification: The multi-view embeddings are fed into both classification heads, producing a preliminary gesture class prediction for the current window.
3. Post-processing: A post-processing step is applied to reduce false positives and spurious gesture classifications. The last k preliminary predictions are stored in a buffer, and a majority voting mechanism is used to determine the final class. This helps smooth out predictions over time and eliminates noise from accidental or incomplete gestures.

4. Experiments

4.1. Datasets

The proposed hand gesture recognition system was evaluated using two datasets: an adapted version of the NVGesture dataset [11] and the SHREC'22 benchmark [4]. These datasets cover various gestures, comprehensively analyzing the model's performance on static and dynamic hand gestures.

NVGesture dataset

The NVGesture dataset consists of over 1,500 video sequences depicting 25 unique hand gestures performed by 20 different participants. The dataset includes a variety of gestures, with both static gestures (e.g., "thumb up," "ok") and dynamic gestures (e.g., "stop," "right"). Originally, this dataset was designed as an image-based gesture dataset [Figure 3], and thus, a significant amount of preprocessing was required to adapt it for 3D hand pose recognition using the proposed model.

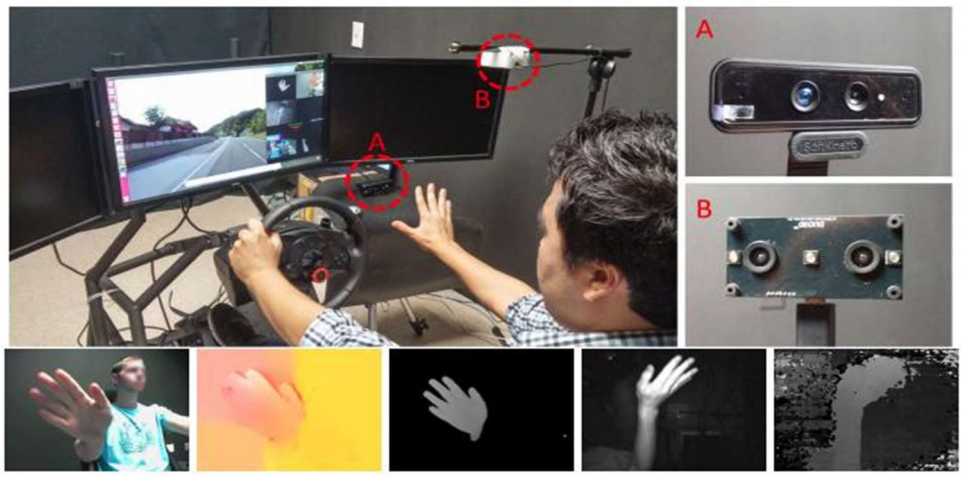

Figure 3: NVGesture data collection setup and a sample of multimodal data.

A custom annotation procedure was developed to prepare the dataset for model training. This procedure utilized Google's Mediapipe framework to automatically extract 2D and 3D joint positions for each gesture sequence. Given that some sequences contained background participants, which resulted in multiple hand detections, a size-based filter was applied. This filter ensured that only the largest detected hand, assumed to belong to the main performer closest to the camera, was retained for the training process.

Another preprocessing step involved resolving issues caused by motion blur, which led to missed detections in certain frames. A windowed interpolation method was applied to address this. In cases where the first and last frames of a 5-frame window were recognized, but intermediate frames were missing, linear interpolation was used to estimate the joint trajectories within the window. This process effectively filled in gaps in the data, ensuring more complete gesture annotations.

SHREC'22 benchmark

The SHREC'22 benchmark features continuous recordings of 3D hand poses captured in simulated Mixed Reality interactions using a Hololens 2 device. The dataset is structured with training and testing sets comprising 144 sequences. Each sequence contains 16 gesture classes interleaved with non-significant hand movements (referred to as "non-gestures"). The dataset includes gestures categorized into four types: static gestures, dynamic coarse gestures, dynamic fine gestures, and periodic gestures.

Each sequence is annotated with start frames, end frames, and gesture labels, making it well-suited for gesture segmentation and classification tasks. Notably, the subjects in the training and testing sequences are different, which ensures a challenging cross-subject evaluation.

4.2. Evaluation metrics

For the quantitative evaluation of the model's performance, the following standard metrics were employed:

Accuracy: Measures the percentage of correctly classified gestures.

Precision: Indicates the proportion of true positive predictions among all positive predictions, capturing the model's ability to avoid false positives.

Recall: Represents the proportion of true positive predictions among all actual positives, reflecting the model's capacity to identify all relevant gestures.

F1 Score: The harmonic mean of precision and recall provides a balanced metric for false positives and false negatives.

4.3. Evaluation process

The evaluation was conducted in two phases: offline evaluation on benchmark datasets and real-time evaluation using live input from a webcam.

Phase 1: offline dataset evaluation

The first phase involved testing the model's performance on the NVGesture dataset and SHREC'22 benchmark. For the NVGesture dataset, the model was trained on the 3D joint data extracted from Mediapipe's world coordinate estimates. To ensure robustness, the evaluation was performed on two types of keypoint representations:

3D keypoints: Using the world-coordinate hand pose estimates provided by Mediapipe.

2D keypoints: Normalized 2D joint coordinates projected from the camera view.

After preprocessing and annotating the NVGesture dataset, the modified model was trained and evaluated on the test partition. The dataset includes a balanced mix of static and dynamic gestures, which allowed for a comprehensive evaluation of the model's ability to recognize various hand motions.

Key challenges included the handling of motion blur and occasional multiple detections in the background, which were mitigated through the custom preprocessing steps described earlier. The results of this evaluation provided insight into the model's robustness in recognizing static and dynamic gestures under various conditions.

The SHREC'22 benchmark posed additional challenges due to the continuous nature of the recordings, where gestures were interleaved with non-significant movements. The model was trained to distinguish between gesture and non-gesture frames, focusing on identifying each gesture's precise start and end.

Due to the variability in subjects and the mix of gesture types (static, dynamic coarse, dynamic fine, and periodic), the SHREC'22 dataset served as a valuable test of the model's generalization capabilities across different individuals and gesture styles.

The evaluation results on both datasets, including accuracy, precision, recall, and F1 score, provided quantitative measures of the model's performance and demonstrated its effectiveness in recognizing static and dynamic gestures.

Phase 2: real-time evaluation

The model was tested in a real-time environment in the second evaluation phase to simulate practical application scenarios. The system was connected to a webcam, and the trained model was deployed to classify gestures as performed live. This evaluation was designed to mimic real-world usage, where the model must operate continuously and make online predictions.

Each gesture type (static, dynamic coarse, dynamic fine, periodic) was attempted three times, and the system's performance was recorded based on the number of successful attempts. A gesture was successful if the model correctly classified it during the real-time session without significant delay or misclassification.

Both 2D and 3D keypoint representations were used during the real-time tests to compare performance. The success rate for each gesture type provided a practical assessment of the model's usability in real-time applications, offering valuable insights into its responsiveness and robustness.

5. Results

5.1. NVGesture dataset evaluation

The evaluation of the model on the NVGesture dataset was conducted using both 2D and 3D hand pose data. Performance metrics such as accuracy, recall, precision, and F1 score were used to assess the model's effectiveness in gesture recognition. The results are presented in Table 1.

Table 1
NVGesture Dataset Evaluation Results

Data Type	Accuracy	Recall	Precision	F1 Score
2D	0.794	0.794	0.763	0.768
3D	0.784	0.758	0.784	0.751

The analysis of the results shows that the model trained on 2D data slightly outperforms the one trained on 3D data in terms of accuracy and F1 score. The 2D model achieved an accuracy of 0.794, compared to 0.784 for the 3D model. However, the 3D data model showed higher precision (0.784 vs. 0.763), suggesting it might be better at minimizing false positives. Overall, the 3D data approach outperformed the original results from the NVGesture dataset, which reported an accuracy of 0.74 using colour data only. This demonstrates the improved performance of our model and its potential for more accurate real-time hand gesture recognition using 3D landmarks.

5.2. Manual evaluation

A manual real-time evaluation was conducted using a webcam to assess the model's ability to generalize beyond the NVGesture dataset. This test involved classifying static gestures, dynamic gestures, and non-gesture frames. The results are summarized in Table 2.

Table 2
Manual Evaluation Results

Gesture Type	2D Data	3D Data
Static	0.345	0.893
Dynamic	0.567	0.712
Non-gesture	0.790	0.921

The manual evaluation highlights a significant performance gap between the 2D and 3D models in real-world applications. The model trained on 3D data demonstrated superior performance across all gesture types, especially notable improvements in static gesture recognition (0.893 accuracy in 3D vs. 0.345 in 2D). This suggests that the 2D model generalizes poorly to real-world data, whereas the 3D model performs well outside controlled dataset conditions. The 3D model's high accuracy for non-gesture frames (0.921) also reflects its ability to distinguish between gesture and non-gesture movements in live input effectively.

5.3. SHREC'22 dataset evaluation

The model's performance was also evaluated on the SHREC'22 dataset, which consists of continuous 3D hand pose recordings captured with a Hololens 2 device. The dataset only contains 3D data, so a direct comparison with 2D data was impossible. The results are provided in Table 3.

The model achieved high accuracy (0.9243) and F1 score (0.9241) on the SHREC'22 dataset, demonstrating its robustness in recognizing gestures in continuous, real-world recordings. The Static-Dynamic-Non-gesture (SDN) accuracy was particularly strong at 0.9497, highlighting the model's ability to accurately distinguish between different gesture types and non-gesture movements in mixed reality environments.

Table 3
SHREC'22 Dataset Evaluation Results

Metric	Value
Accuracy	0.924
Precision	0.926
Recall	0.924
F1 Score	0.924
SDN Accuracy	0.950

6. Discussions

This research demonstrates that using 3D hand pose data significantly improves real-time gesture recognition compared to previous approaches relying on 2D data or colour information. On the NVGesture dataset, our method achieved 0.784 accuracy, outperforming the original paper's 0.74 accuracy, which only utilized color data. The improvement can be attributed to the detailed spatial information captured by 3D hand pose data, allowing for a better understanding of hand movements. Furthermore, our modifications to the DD-Net architecture, including residual connections and an auxiliary classification head to handle class imbalance, proved effective in refining gesture classification.

A key finding is that models trained on 2D data generalized poorly in real-world scenarios, as evidenced by our manual evaluation, where the 2D model struggled with static gestures (0.345 accuracy), while the 3D model performed significantly better (0.893 accuracy). This highlights the robustness of 3D data in handling varying environments, lighting conditions, and background noise, making it more suitable for real-world applications like smart devices and touchless interfaces.

Our evaluation of the SHREC'22 dataset, which captures continuous gestures in mixed-reality environments using Hololens 2, further validated the approach. The model achieved a strong 0.924 accuracy and distinguished static, dynamic, and non-gestures, with 0.949 SDN accuracy. These results suggest that the method can be effectively integrated into augmented reality (AR) [17] and virtual reality (VR) applications, where continuous gesture recognition is critical.

7. Conclusions

This research has significantly advanced dynamic hand gesture recognition, presenting a robust real-time system that effectively utilizes 3D and 2D data for gesture classification. By adopting the DD-Net architecture and introducing modifications such as an auxiliary classification head and attention mechanisms, the system demonstrated improved accuracy, particularly when working with 3D data, achieving 0.784 accuracies on the NVGesture dataset—outperforming prior benchmarks based solely on 2D colour data. The automatic annotation procedure using Google Mediapipe allowed for efficient data preprocessing, further enhancing the system's performance.

The evaluation of the system was comprehensive, including tests on benchmark datasets like NVGesture and SHREC22, as well as real-time manual evaluations using a webcam. The results showed that the system, particularly when trained on 3D data, generalizes well to real-world environments, achieving superior performance in dynamic and static gestures and non-gesture categories.

The system remains within acceptable limits for real-time applications with an average inference time of 5 msec (combined with 22 msec per inference from Google Mediapipe). This low-latency performance makes it particularly valuable for practical use cases, such as controlling smart devices or providing touchless interfaces in public spaces, where health concerns and convenience are of growing importance.

References

[1] Zhang, Wei, Zeyi Lin, Jian Cheng, Cuixia Ma, Xiaoming Deng, and Hongan Wang. 2020. "STA-GCN: Two-Stream Graph Convolutional Network with Spatial-Temporal Attention for Hand Gesture Recognition." The Visual Computer 36 (10–12): 2433–44. https://doi.org/10.1007/s00371-020-01955-w.

[2] Do, Nhu-Tai, Soo-Hyung Kim, Hyung-Jeong Yang, and Guee-Sang Lee. 2020. "Robust Hand Shape Features for Dynamic Hand Gesture Recognition Using Multi-Level Feature LSTM." Applied Sciences (Basel, Switzerland) 10 (18): 6293. https://doi.org/10.3390/app10186293.

[3] Yang, Fan, Yang Wu, Sakriani Sakti, and Satoshi Nakamura. 2019. "Make Skeleton-Based Action Recognition Model Smaller, Faster and Better." In Proceedings of the ACM Multimedia Asia. New York, NY, USA: ACM.

[4] Emporio, Marco, Ariel Caputo, Andrea Giachetti, Marco Cristani, Guido Borghi, Andrea D'Eusanio, Minh-Quan Le, et al. 2022. "SHREC 2022 Track on Online Detection of Heterogeneous Gestures." Computers & Graphics 107: 241–51. https://doi.org/10.1016/j.cag.2022.07.015.

[5] Quentin de Smedt, Hazem Wannous, Jean-Philippe Vandeborre, Joris Guerry, Bertrand Le Saux, et al.. SHREC'17 Track: 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset. 3DOR - 10th Eurographics Workshop on 3D Object Retrieval, Apr 2017, Lyon, France. pp.1-6, 10.2312/3dor.20171049_xffff_. _xffff_hal-01563505_xffff_

[6] Maghoumi, Mehran, and Joseph J. LaViola Jr. 2019. "DeepGRU: Deep Gesture Recognition Utility." In Lecture Notes in Computer Science, 16–31. Cham: Springer International Publishing.

[7] Caputo, Fabio Marco, S. Burato, Gianni Pavan, Théo Voillemin, Hazem Wannous, Jean-Philippe Vandeborre, Mehran Maghoumi, Eugene Matthew Taranta, A., Razmjoo, Joseph J. Laviola, Fabio Manganaro, Stefano Pini, Guido Borghi, Roberto Vezzani, Rita Cucchiara, H. Nguyen, Minh-Triet Tran and Andrea Giachetti. "SHREC 2019 Track: Online Gesture Recognition." (2019).

[8] Zhang, Fan, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang and Matthias Grundmann. "MediaPipe Hands: On-device Real-time Hand Tracking." ArXiv abs/2006.10214 (2020): n. pag.

[9] Teslyuk, V., Tsmots, I., Gregus ml, M., Teslyuk, T., Kazymyra, I. Methods for the efficient energy management in a smart mini greenhouse Computers, Materials and Continua, 2022, 70(2), pp. 3169–3187

[10] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. "Deep Residual Learning for Image Recognition." In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE.

[11] Molchanov, Pavlo, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. 2016. "Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks." In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE.

[12] Song, Jae-Hun, Kyeongbo Kong, and Suk-Ju Kang. 2022. "Dynamic Hand Gesture Recognition Using Improved Spatio-Temporal Graph Convolutional Network." IEEE Transactions on Circuits and Systems for Video Technology: A Publication of the Circuits and Systems Society 32 (9): 6227–39. https://doi.org/10.1109/tcsvt.2022.3165069.

[13] Miah, Abu Saleh Musa, Md Al Mehedi Hasan, and Jungpil Shin. 2023. "Dynamic Hand Gesture Recognition Using Multi-Branch Attention Based Graph and General Deep Learning Model." IEEE Access: Practical Innovations, Open Solutions 11: 4703–16. https://doi.org/10.1109/access.2023.3235368.

[14] Kopuklu, Okan, Ahmet Gunduz, Neslihan Kose, and Gerhard Rigoll. 2019. "Real-Time Hand Gesture Detection and Classification Using Convolutional Neural Networks." In 2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019). IEEE.

[15] Zhan, Felix. 2019. "Hand Gesture Recognition with Convolution Neural Networks." In 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI). IEEE.

[16] Vaezi Joze, Hamid Reza, Amirreza Shaban, Michael L. Iuzzolino, and Kazuhito Koishida. 2020. “MMTM: Multimodal Transfer Module for CNN Fusion.” In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE.
[17] D. Ostrovka, T. Teslyuk, P. Veselý, and I. Protsko, «The Analysis and Comparison of File Formats for the Construction of iOS OS Three-dimensional Objects for Augmented Reality Systems», DCSMart, pp. 325–334, Lviv, Ukraine, 2019