Article

Multidisciplinary ML Techniques on Gesture Recognition for People with Disabilities in a Smart Home Environment

Christos Panagiotou * , Evanthia Faliagka * , Christos Antonopoulos , Nikolaos Voros

Posted Date: 7 November 2024

doi: 10.20944/preprints202411.0452.v1

Keywords: gesture detection; edge computing; Internet of Things system)

Article

Multidisciplinary ML Techniques on Gesture Recognition for People with Disabilities in a Smart Home Environment

Christos Panagiotou * , Evanthia Faliagka *, Christos P. Antonopoulos and Nikolaos Voros

Electrical & Computer Engineering Department, University of Peloponnese, M. Alexandrou 1, 22100 Patras, Greece

* Correspondence: ch.panagiotou@esdalab.ece.uop.gr (C.P.); e.faliagka@esdalab.ece.uop.gr (E.F.)

Abstract: Gesture recognition has a crucial role in Human-Computer Interaction (HCI) and in assisting the elderly to perform automatically their everyday activities. In this paper three methods for gesture recognition and computer vision were implemented and tested in order to investigate the most suitable one. All methods, Machine learning using IMU, Machine learning on device and were combined with certain activities that were determined during a needs analysis research. The same volunteers took part to the pilot testing of the proposed methods. The results highlight the strengths and weaknesses of each approach, revealing that 60 while some methods excel in specific scenarios, the integrated solution of MoveNet and 61 CNN provides a robust framework for real-time gesture recognition.

Keywords: gesture detection; edge computing; Internet of Things system

1. Introduction

Gesture recognition is increasingly critical for the elderly, as it provides a seamless and intuitive way to interact with technology and perform daily activities, enhancing their safety, independence, and overall quality of life. As physical mobility or dexterity may decline with age, traditional interaction methods like pressing small buttons or navigating complex interfaces can become challenging. Gesture recognition, such as waving to turn on lights or signaling for assistance, simplifies these interactions and reduces the need for physical strain. Additionally, it allows for real-time monitoring, where specific gestures can alert caregivers to falls or other emergencies, enabling rapid response in potentially life-threatening situations. By minimizing the barriers to technology, gesture detection empowers older adults or people with disabilities to live more comfortably and safely within their environments, bridging the gap between their needs and available smart solutions [1]. Gesture recognition also plays a significant role in Human-Computer Interaction (HCI) by enabling more natural, intuitive, and efficient ways to interact with digital systems. Unlike traditional input methods like keyboards, mice, or touchscreens, gesture recognition allows users to control devices and interfaces with simple body movements, making interaction more fluid and accessible. This ability is especially crucial in environments where hands-free control is advantageous, such as in medical settings, industrial workplaces, or smart homes. The importance of gesture recognition in HCI lies in its potential to bridge the gap between humans and technology. By recognizing and interpreting gestures, computers can respond to users' needs in real time, often without requiring physical contact, which enhances accessibility and usability [2]. For example, in virtual reality (VR) and augmented reality (AR), gesture recognition allows users to manipulate objects in a virtual environment as they would in the real world, creating a more immersive experience [3]. Additionally, it supports inclusivity by providing alternative interaction methods for people with disabilities or limitations that prevent traditional device use. In summary, gesture recognition in HCI not only facilitates smoother, more natural interactions but also expands the scope of who can effectively use and benefit from technology, thus advancing the field of HCI toward more user-centered and accessible designs. Gesture recognition has diverse applications, from gaming and virtual reality, where it enhances immersive experiences, to accessibility technology, where it provides alternative input methods for those with limited mobility. It's also valuable in AAL environments, where it allows users to control lighting, appliances, or security systems through simple movements [4]. This technology is increasingly being integrated into sectors like healthcare and rehabilitation [5], transforming how we interact with digital environments. There are various gesture recognition techniques that are used depending on the application needed. The primary objective of this paper is to investigate the more effective technique to detect the gesture and make the respective action. Therefore, hands will act as a remote control for all household electronics when hand gestures are used instead of button presses. Three methods were tested for the gesture recognition: using machine learning classification techniques with a Sensor Tag with BLE technology, using deep learning techniques on device with an Arduino and computer vision that extracts features from upper body joints and trains a respective AI model to identify gestures. Several daily activities can become challenging for elderly individuals due to physical limitations, cognitive changes, or chronic health conditions. Automating these tasks using gesture recognition can improve their quality of life, safety, and independence. We conducted a research using questionnaires to explore which activities seniors find difficult and would like to do with a gesture. The research included 30 seniors (over 60 years old), both men and women and noted that the critical tasks where automation could help are managing the lights, finding their mobile phone and managing locks to help them keep the home safe. So, in this paper, 4 gestures were studied and each one of them was correlated with an action (lights on, lights off, find mobile, activate locks).

This article is a revised and expanded version of a paper entitled "Gestures detection and device control in AAL environments using machine learning and BLEs", which was presented at the 12th Mediterranean Conference on Embedded Computing (MECO), 2023 [4]. The paper expands the adopted methods by presenting a solution that employs MoveNet and CNN in a deep learning technique. The results highlight the strengths and weaknesses of each approach, revealing that while some methods excel in specific scenarios, the integrated solution of MoveNet and CNN provides a robust framework for real-time gesture recognition. This combination not only capitalizes on the strengths of deep learning for feature extraction but also enhances accuracy through precise keypoint detection.

The rest of the paper is organized as follows. In Section 2 there is a review of the literature, in Section 3 the three methods are described, in Section 4 we discuss about the findings and the future work challenges and Section 5 concludes the paper.

2. Related Work

Numerous research efforts have suggested the implementation of wearable devices integrated with sensors to recognize hand gestures and relay commands to a central hub or gateway device. For instance, Zhang et al. [6] created a wearable device that incorporates an accelerometer and a gyroscope for the detection of hand gestures, which are subsequently communicated to a hub equipped with Bluetooth Low Energy (BLE) for the purpose of device management. The system demonstrated a recognition accuracy exceeding 90% for a range of gestures, such as swipes, taps, and twists. Additionally, other research has investigated the application of machine learning algorithms for gesture recognition within BLE-based systems.

Rashed et al. [7] introduced a system that employs a neural network for the recognition of hand gestures, enabling the control of devices within a smart home setting. Their system attained a gesture recognition accuracy of 94% and demonstrated the capability to manage a range of devices, including lights, fans, and televisions. Bui et al. [8] developed a MEMS accelerometer based glove for gesture recognition. It consists of six MEMS accelerometers, which greatly improves the process of recognition. The current system is limited to gesture recognition in two dimensions. To address this constraint, Kim et al. [9] created the KHU-1 data glove, which incorporates three tri-axis accelerometer sensors, enabling it to execute three-dimensional rule-based hand motion tracking and gesture recognition. A significant drawback of these methods is their reliance on specialized equipment, which may not be user-friendly for elderly individuals.

In addition to wearable devices, some studies have also investigated the use of cameras for gesture detection in BLE based systems. For example, Kim et al. [10] developed a system that uses a camera and a BLE-enabled hub to detect hand gestures and control a robotic arm. Their system achieved an accuracy of 96.7potential of using BLE for controlling robotic systems. Several challenges and limitations of BLE-based gesture detection and device control systems have also been identified in the literature. One challenge is the limited range of BLE, which may restrict the use of these systems to small areas or rooms. Another challenge is the need for a reliable and secure communication protocol between the wearable device and the hub or gateway device.

Hand gesture recognition has become a pivotal area within computer vision, facilitating intuitive human-computer interaction. Moreover, computer vision solutions in contrast to the wearable IMU-based systems, attract interest due to unobtrusive way they operate. Various approaches have been developed over the years, ranging from traditional image processing techniques to advanced deep learning algorithms [11]. Early methods often relied on position markers or colored bands [12] to identify gestures, which proved cumbersome and impractical for real-world applications, particularly in dynamic environments like robotics.

Recent advancements have shifted towards more sophisticated techniques that leverage deep learning, particularly Convolutional Neural Networks (CNNs) [13,14]. CNNs excel in extracting hierarchical features from images, making them highly effective for tasks such as image classification, object detection, and segmentation. Numerous studies demonstrate the efficacy of CNNs in recognizing hand gestures by training on large datasets of labeled images, allowing them to learn intricate patterns associated with specific gestures [15–17].

While gesture recognition technology has advanced significantly, it faces ongoing challenges that necessitate further research and development. Addressing issues related to occlusion, variability among users and environmental influences is critical for enhancing the robustness and reliability of gesture recognition systems.

3. Gesture Recognition Techniques

3.1. Wearable IMU- Machine Learning on the Cloud

In this scenario a wearable sensor module capable of detecting gestures was used. Specifically, a Texas Instrument CC2650 Sensor Tag with BLE technology was used that has an accelerometer, gyroscope and magnetometer. This sensor was fitted like a bracelet on the user's hand. The data was transmitted through Bluetooth to a computer, where artificial intelligence algorithms are employed to interpret the user's motion and issue corresponding commands for home automation.

3.1.1. Training Data Collection

An AI algorithm must first acquire knowledge of the movements it is intended to recognize. To facilitate this, an experiment was conducted involving 30 volunteers who were instructed to perform each gesture multiple times, thereby enabling the creation of a decision model. Consequently, whenever a user executes a gesture, the model is capable of identifying the specific gesture performed. The advantage of artificial intelligence algorithms lies in their ability to overlook minor variations in movements, allowing them to recognize recurring patterns—something that traditional deterministic algorithms have found challenging to accomplish. Initially, a suitable algorithm was developed using the Python programming language to capture motions from the Sensortag. This involved utilizing the accelerometer, gyroscope, and magnetometer from the IMU sensor to document particular hand movements. Then, the Orange machine learning application was employed, as shown in Figure 3. The model's training was conducted using a classification method across four distinct classes (0, 1, 2, 3), with each class corresponding to a specific gesture.

3.1.2. Training Process

Following the collection of data and its categorization into the specified classes, the information was imported into the Orange tool to commence the training process. To develop the classifier, three different classification algorithms were utilized to identify the most effective algorithm based on its accuracy. The three classification models involved in the training process included: a) Random Forest, b) Tree, and c) Neural Network, with their respective parameters detailed in Table 1. After model's training using the three different classification algorithms mentioned earlier, it was ultimately decided to use the Random Forest algorithm as it provides the best training curve. The classification results are shown in Table 2.

Table 1. Accuracy results of all cases.

Random Forest	Tree	Neural Network
Number of Trees:19	Min Number of Instances in leaves:2	Max number of iterations:200
-	Do not split subsets smaller than:9	Neurons in hidden layers:100
-	Limit the maximal tree depth to:100	Activation:Logistic
-	Stop when majority reaches:95% -	

Table 2. Accuracy results of all cases.

	Random Forest	Tree	Neural Network
CA	0,996	0,994	0,995
F1	0,997	0,991	0,992
Precision	0,998	0,899	0,996
Recall	0,999	0,887	0,996
0			

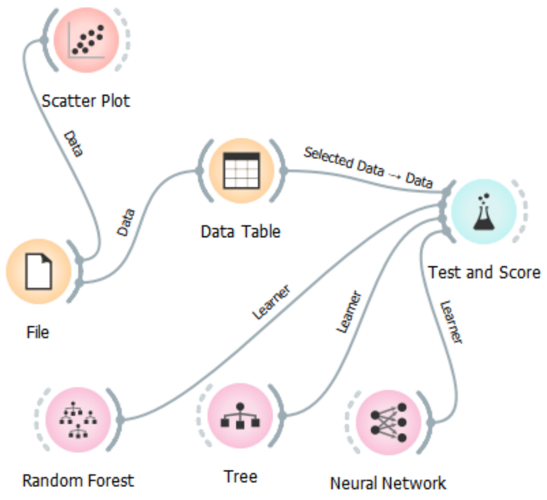

Figure 1. 10s sample from the rightward movement and from the upward movement.

3.2. Wearable IMU- Machine Learning on Device

To gather the necessary training data, an Arduino Nano 33 BLE sense was utilized. This board is deemed ideal for IoT applications as it provides an array of sensors (with IMUs and wireless interfaces being particularly relevant to our study) in compact dimensions (18 × 45 mm) and is compatible with models trained on the Edge Impulse platform. By utilizing the board's sensors, the arm's movements were tracked, with a focus on upward, downward, leftward, and rightward motions.

3.2.1. Training Data Collection

An image of two samples of raw data collected is shown below (Figure 2). This figure shows a sample data of 10s during the rightward movement and a sample data of 10s during the upward movement. As the accelerometer on the development board has three axes, three different lines are shown, one for each axis (accX, accY, and accZ). Accordingly, there were three lines for the gyroscope measurements (gyrX, gyrY, and gyrZ) and three more for the magnetometer measurements (magX, magY, and magZ). From Figure 2a, we can understand that the accelerometer y and z values did not have a crucial role in model creation, as they had small ripples, which is expected as the movement is on the horizontal axis. The movement we studied affected the accelerometer values in the x axis, as the movement was rightward, and these values were critical for model creation. Accordingly, in Figure 2b, the z values of the accelerometer have the crucial role in model creation. In order to gather the training data, 30 volunteers (the same volunteers of the first scenario) performed continuous movements, and several samples were collected lasting 10–12 s each. These samples were equally distributed throughout the movements (5 samples per user per movement). Arduino was held steadily on the wrist like a bracelet, and the position of it remained unchanged.

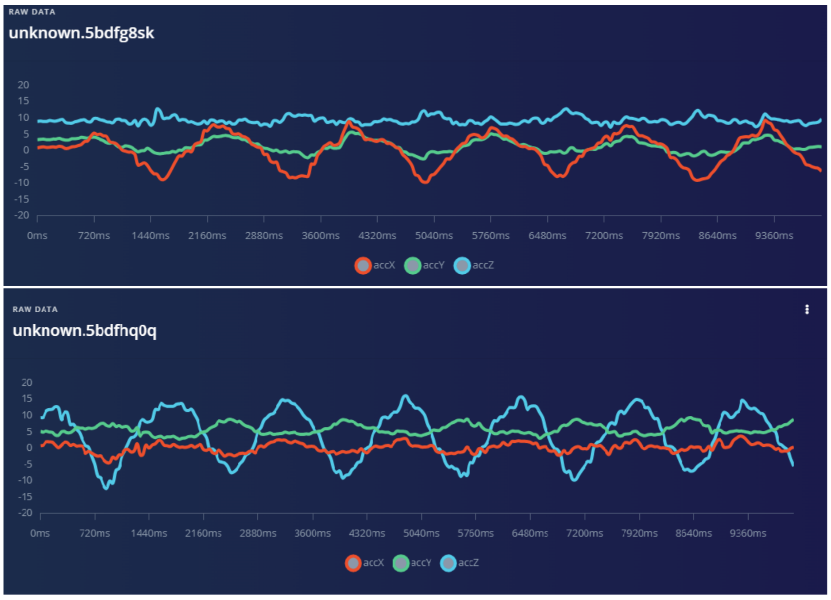

Figure 2. 10s samples from the rightward and uppward movement.

3.2.2. Data Preprocessing, Feature Extraction, and Training

Following the data collection phase, an annotation procedure was carried out to assign labels to each sample for the purpose of supervised classification. The raw data were divided into smaller segments, and signal processing techniques were applied to extract relevant features. In particular, spectral analysis was utilized, which is effective for examining repetitive movements, like those recorded by accelerometers. It is widely recognized [18,19] that extracting features from the frequency domain (i.e., the power characteristics of the signal) enhances performance. These features are particularly important for classifying repetitive motions using accelerometer data, as the FFT identifies periodic signals and decomposes them into their harmonic components, which helps to lower the data dimensionality. Microcontrollers utilize their DSP engines to perform feature extraction efficiently. The features identified in the proposed system included the RMS value as a time domain feature and the peaks, frequency, and power characteristics of the signal as frequency domain features. This approach produced a consistent dataset consisting of 33 features (11 for each axis). The initial window size was set to 5000 ms, and it was expanded by 50 ms. The window size represented the dimensions of the raw features utilized during training. The increase in window size was intended to artificially generate additional features, providing the learning block with more data.

3.2.3. Machine Learning Model Creation and Accuracy Evaluation

In the process of classification, an algorithm acquires knowledge from a specified dataset and categorizes new data into multiple classes or groups. This paper focused on employing neural networks to classify the hand movement into one of four predefined categories. The previously discussed steps were implemented. Initially, the training and test sets were established using an 80/20 train/test split ratio. The training set, consisting of 480 samples, was generated from the movements of the first 24 participants performing 5 repetitions for each of the four gestures, while the test set, comprising 120 samples, was created from the movements of the last 6 participants. On the Edge Impulse platform, the model was developed as a pipeline utilizing the neural network classifier, with appropriate parameters selected to achieve optimal accuracy through heuristic methods. A neural network is composed of layers of interconnected neurons, each connection assigned a specific weight. For instance, a neuron in the input layer may represent the height of the first peak along the X-axis, such as AccX, while a neuron in the output layer could correspond to a class, such as "Gesture1". When the neural network is defined, all connections are initialized randomly, resulting in initial random predictions. During the training phase, the network generates predictions based on the raw data and subsequently adjusts the weights according to the results. Through numerous iterations, the neural network refines its ability to predict new data effectively. The total number of training cycles indicates the number of iterations completed. In the context of the proposed system, we executed 50 training cycles to ensure the development of a well-trained model. The decision to utilize 50 cycles was made heuristically, based on extensive repetitions within the parameter space, which revealed that results began to plateau beyond this point. The selected parameters are detailed below and presented in Table 3. The training cycle count was set at 50, with a learning rate of 0.0005. All parameters were determined through an iterative process aimed at enhancing accuracy while simultaneously mitigating the risk of overfitting. As shown in Table 4, the model achieved excellent accuracy, with the lowest recorded accuracy being 97.5. To enable the system to effectively handle previously unseen data, such as new gestures, the Anomaly Detection block provided by Edge Impulse was integrated into the proposed system. This block allows the neural network to recognize anomalous values and prevents their classification into established categories.

Table 3. Parameters and their values for the training process.

Number of training cycles	50
Learning rate	0.0005
Validation set size	20
Sampling frequency	62.5 (Hz)
Input layer	33 features
First dense layer	20 neurons
Second dense layer	10 neurons

Table 4. Accuracy results of all cases.

Gesture 1	**Gesture 2**	**Gesture 3**	**Gesture 4**	**Overall**	**F1 Score**	
Training	99,8%	99,7%	96,5%	94,9%	97,73%	97,64%
Classification	100%	100%	97,5%	98,9%	99,1%	98,92%

3.2.4. Machine Learning on Device

A significant aspect of this paper was the adaptation and transfer of the training model to the Arduino Nano 33 BLE board, facilitating on-device predictions. Furthermore, the code was adjusted to incorporate Bluetooth Low Energy technology, allowing the transmission of classification results to a mobile device (e.g. inform someone that he locked the door). With BLE activated, the algorithm's decisions could be relayed to a remote location, such as a mobile device, while the gesture prediction was performed on the device itself to effectively assess the performance of the edge computing implementation of the proposed solution.

3.3. Vision Based

Despite the success of CNNs, as described previously, challenges remain in achieving high accuracy and efficiency in gesture recognition. Traditional methods often struggle with variations in lighting conditions and camera angles. To address these issues, research focused on exploring keypoint detection as a means of simplifying the input data for gesture recognition systems [20]. Keypoint-based methods focus on identifying critical points on the hand and body, allowing for a more robust representation of gestures that is less sensitive to environmental changes.

MoveNet [21], a state-of-the-art model for pose estimation, emerged as a leading solution for keypoint extraction. It efficiently detects 17 keypoints in real-time, providing a reliable framework for understanding hand positions and movements without requiring additional markers. This advancement significantly enhances the robustness of gesture recognition systems by normalizing the input data.

The integration of MoveNet for keypoint extraction with CNNs for classification represents a significant leap forward in gesture recognition technology. By using MoveNet to extract keypoints, the system can focus on relevant features such as hand position, orientation, and movement dynamics while discarding irrelevant background information. This streamlined approach not only reduces computational complexity but also improves accuracy by providing the CNN with high-quality input data.

In this paper we present a combined solution, with an architecture that consists of two main components: the keypoint extraction stage using MoveNet and the classification stage employing a CNN. The output from MoveNet—a set of x and y coordinates representing keypoints along with additional features that are extracted serve as input to the CNN. The CNN then processes these coordinates through multiple convolutional layers designed to learn complex patterns associated with specific gestures.

This combined approach leverages the strengths of both models, such as:

- Efficiency: By using keypoints instead of raw images, the computational load on the CNN is significantly reduced, enabling faster processing times.
- Robustness: Keypoint-based representations are less sensitive to variations in lighting and background conditions, leading to more consistent gesture recognition performance.
- Real-Time Performance: MoveNet's ability to run at over 30 frames per second ensures that the system can provide immediate feedback, which is crucial for interactive applications.

3.3.1. Feature Extraction

To implement the gesture recognition in smart home systems, particularly through computer vision techniques like MoveNet, several features were extracted from the detected keypoints. These features are the main source of information for the accurate interpretation of the user gestures and translating them into actionable commands. In this work the following features were extracted to form the input vectors of the system (either for training or inference):

- Distance Between Keypoints: Calculating the Euclidean distances between keypoints (e.g., between hands or from hands to shoulders) helps identifying specific gestures based on the spatial relationships of body parts. First, Euclidean distances are calculated between specific pairs of keypoints, such as between the hands or from hands to shoulders, helping to identify gestures based on the spatial relationship of body parts.

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

For three-dimensional space, if the keypoints are represented as $P_1(x_1, y_1, z_1)$ and $P_2(x_2, y_2, z_2)$, the formula extends to:

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

- Angle Between Joints: The angles formed at joints (e.g., elbow and shoulder angles) are indicative of certain gestures, such as waving or pointing Joint angles are calculated by examining specific body joints, such as at the fingers or arm and shoulder, which can help recognize gestures that involve particular arm movements, like waving or pointing.

  To calculate the angle θ at a joint formed by three keypoints $A(x_a, y_a)$, $B(x_b, y_b)$, and $C(x_c, y_c)$, we used the cosine rule. The lengths of the sides formed by these points are given by:

  - Length $AB = d(A, B)$ - Length $BC = d(B, C)$ - Length $AC = d(A, C)$

  Using the cosine rule:

$$\cos(\theta) = \frac{AB^2 + BC^2 - AC^2}{2 \cdot AB \cdot BC}$$

  Thus,

$$\theta = \cos^{-1}\left(\frac{AB^2 + BC^2 - AC^2}{2 \cdot AB \cdot BC}\right)$$

- Velocity of Movement: Tracking the speed at which keypoints move differentiate between slow gestures (like a gentle wave) and fast movements (like a quick swipe), enhancing gesture recognition accuracy. The velocity of each keypoint is then assessed by tracking positional changes across consecutive frames, allowing for differentiation between slow and rapid gestures, such as a gentle wave versus a fast swipe.

  The velocity v_i of a keypoint can be calculated by tracking its position over time. If the position of a keypoint at time t_i is $P_i(x_i, y_i)$, then the velocity between two consecutive frames can be expressed as:

$$v_i = \frac{P_{i+1} - P_i}{\Delta t}$$

  where $P_{i+1} = (x_{i+1}, y_{i+1})$, $P_i = (x_i, y_i)$, and Δt is the time interval between frames.
- Gesture Duration: The time taken to perform a gesture as a distinguished feature. For instance, holding a hand in a specific position for an extended period might indicate a different command than a quick motion. Gesture duration is also measured by tracking the consistency of a gesture over a sequence of frames, distinguishing longer, held gestures from quicker motions.

  Gesture duration is measured by counting the number of frames in which a gesture is consistently detected. If a gesture is detected in frames from t_{start} to t_{end}:

$$\text{Duration} = t_{\text{end}} - t_{\text{start}}$$

- Hand Orientation: The orientation of the hand, e.g., palm facing up or down The orientation of each hand is determined based on the relative positions of keypoints on the wrist, hand, and fingers, which helps recognize gestures with specific hand orientations, such as a palm facing upward or downward. For example, if we have keypoints for the wrist $W(x_w, y_w)$, index finger tip $I(x_i, y_i)$, and middle finger tip $M(x_m, y_m)$, we can calculate the orientation vector as:

$$V_{\text{hand}} = (x_i - x_w, y_i - y_w)$$

The angle of orientation relative to a reference axis (e.g., horizontal axis along x-axis) can be calculated using:

$$\text{Orientation Angle} = \tan^{-1}\left(\frac{y_i - y_w}{x_i - x_w}\right)$$

After extracting these features, they are compiled into feature vectors, with each vector representing the characteristics of a frame or sequence of frames.

3.3.2. Deep Learning Approach

For the training of the extracted features, we evaluated several techniques based on the existing research literature. These techniques leverage machine learning and deep learning methodologies to effectively classify gestures based on the extracted features.

Convolutional Neural Networks (CNNs) are highly effective for processing spatial data, making them suitable for analyzing images or video frames where gestures can be represented visually. They automatically learn features from the input data, which is beneficial for recognizing complex gestures. CNNs excel in extracting hierarchical features from images, making them highly effective for tasks such as image classification, object detection, and segmentation.

One effective architecture is the SqueezeNet model, which is known for its lightweight design while maintaining high accuracy. This architecture consists of 18 layers, including convolutional layers, pooling layers, and eight fire modules, which are essential for feature extraction from hand gesture images.

1. Input Layer: The input layer receives pre-processed images of hand gestures. These images are resized to a consistent dimension (224 $\times$ 224 pixels) to ensure uniformity across the dataset.
2. Convolutional Layers: The initial layers consist of convolutional filters that extract low-level features such as edges and textures from the input images. This is followed by activation functions (ReLU) to introduce non-linearity into the model.
3. Fire Modules: SqueezeNet employs fire modules that consist of a squeeze layer (1x1 convolutions) followed by an expand layer (1 $\times$ 1 and 3 $\times$ 3 convolutions). This structure allows the network to learn the set of features while keeping the model size small.
4. Pooling Layers: Max pooling layers are interspersed throughout the network to downsample feature maps, reducing dimensionality and computational load while retaining important spatial information.
5. Dropout Layers: To prevent overfitting during training, dropout layers are added and randomly set a fraction of input units to zero at each update during training time.
6. Fully Connected Layers: After the convolutional and pooling layers, the output is flattened and passed through one fully connected layer that perform high-level reasoning about the features extracted by the previous layers.
7. Output Layer: The final layer uses a softmax activation function to classify the gestures into predefined categories (e.g., "lights on," "lights off," "find mobile," "activate locks").

3.3.3. Training Pipeline

The training process for the combined MoveNet and CNN solution involved several key steps to ensure effective gesture recognition. Initially, a diverse dataset was collected, comprising various hand gestures performed by multiple participants as described earlier with IMU approaches. This dataset is essential for training the model to recognize and classify different gestures accurately.

For this approach, participants were instructed to perform a set of predefined gestures multiple times. The gestures included the actions "lights on," "lights off," "find mobile," and "activate locks." Each gesture was captured using a camera system that employs MoveNet for real-time keypoint extraction. The extracted keypoints, which represent critical parts of the arm and hand and their spatial relationships, serve as the input features for the CNN.

The CNN architecture used in this solution consisted of multiple convolutional layers followed by pooling layers, fully connected layers, and an output layer. The convolutional layers extracted hierarchical features from the input keypoints, while pooling layers helped reduce dimensionality and enhance computational efficiency. The final output layer uses a softmax activation function to classify the gestures into their respective categories.

The training process involved feeding the preprocessed data into the CNN and optimizing the model parameters using backpropagation. A suitable loss function, such as sparse categorical cross-entropy, was employed to measure the difference between predicted and actual gesture classes. The model's performance was evaluated using metrics such as accuracy, precision, recall, and F1-score during training.

To prevent overfitting and enhance generalization, techniques such the dropout regularization and data augmentation techniques were applied. Data augmentation include transformations like rotation or scaling of keypoint coordinates to simulate variations in gestures that might occur in real-world scenarios.

After training, the model was validated using a separate test set that was not included in the training dataset. This evaluation helped assess how well the model generalizes to unseen data. The performance metrics obtained during this phase are critical for determining the effectiveness of the combined MoveNet and CNN approach in recognizing hand gestures accurately.

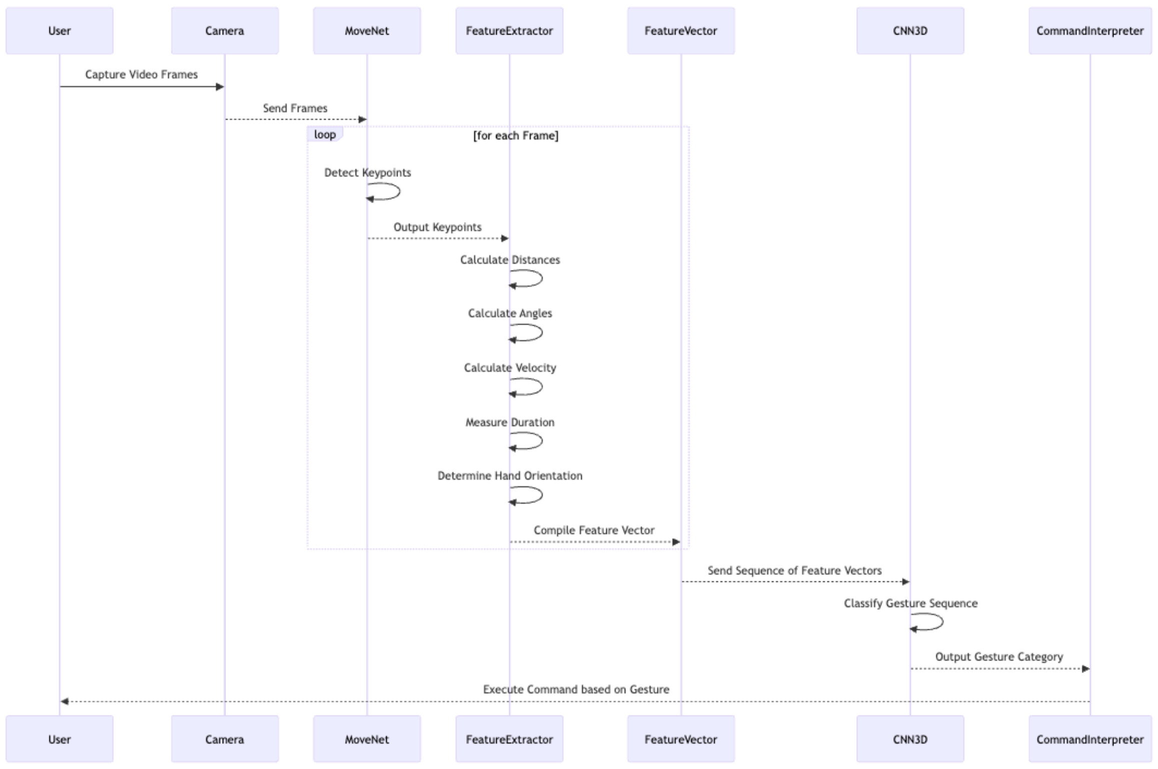

Figure 3. Sequence diagram for the process of the vision based gesture detection.

3.3.4. Inference Pipeline

The gesture recognition architecture utilizing MoveNet and a CNN that has been adopted and implemented involves a structured pipeline where video frames are initially captured as input. These frames are preprocessed to ensure compatibility with MoveNet, which is responsible for detecting body keypoints. Once detected, these keypoints provide essential positional information, such as coordinates of various body parts.

The next stage involves extracting key features from the detected keypoints. These feature vectors are then processed by the CNN, which classifies gestures. The CNN structure consists of convolutional layers that capture patterns within the sequences, pooling layers that reduce dimensionality while preserving critical gesture information, and fully connected layers that output gesture classifications.

Once classified, the recognized gesture is interpreted as a command, which can be mapped to specific actions, such as controlling devices or triggering applications. For real-time performance, this architecture is optimized to ensure efficient processing, leveraging GPU or TPU resources as needed to maintain low latency and smooth recognition. The result is a comprehensive pipeline for gesture recognition that moves seamlessly from video input to gesture-based command execution.

The results indicate strong performance across all gestures. The gesture "Lights On" achieved an accuracy of 96.5%, with a precision of 97.0%, indicating effective identification with minimal false positives. The "Lights Off" gesture recorded an accuracy of 95.2% and a precision of 95.6%, suggesting room for improvement in recall at 94.7%. The "Find Mobile" gesture had an accuracy of 94.8% and a precision of 95.3%, with a recall rate of 93.9%, highlighting potential enhancements in detection. The "Activate Locks" gesture showed the highest performance metrics, with an accuracy of 97.0% and a precision of 97.5%. Overall, these results demonstrate that the combined MoveNet/CNN solution effectively recognizes gestures in smart home applications, paving the way for intuitive user interactions.

Table 5. Performance Metrics for the vision based approach.

Gesture	Accuracy	Precision	Recall	F1-Score
Lights On	96.5%	97.0%	95.8%	96.4%
Lights Off	95.2%	95.6%	94.7%	95.1%
Find Mobile	94.8%	95.3%	93.9%	94.6%
Activate Locks	97.0%	97.5%	96.5%	96.9%

4. Discussion and Future Work Challenges

The evaluation of various gesture recognition approaches presented in this paper highlights the strengths and limitations of each method in the context of enhancing human-computer interaction, particularly for elderly individuals and those with disabilities. The results from the combined MoveNet and CNN solution demonstrate a notable performance in accuracy and reliability. The high performance metrics achieved across different gestures indicate that this hybrid approach effectively captures the nuances of hand movements, enabling intuitive control over smart home devices.

However, challenges remain across all methods discussed. Variability in user gestures, influenced by individual physical differences and environmental conditions, can lead to inconsistencies in recognition accuracy. For instance, while wearable devices provide precise data, they may not be user-friendly for all demographics, particularly elderly users who may struggle with complex setups. Similarly, camera-based systems may face difficulties with occlusions or varying lighting conditions, impacting their effectiveness in real-world applications.

Moreover, while deep learning techniques like CNNs have shown promise in recognizing gestures through large datasets, they require substantial computational resources and extensive training data to generalize effectively across diverse user populations. This necessity raises concerns about accessibility and the practicality of deploying such systems in everyday settings.

Future research should focus on addressing these challenges by exploring adaptive learning algorithms that can personalize gesture recognition systems based on individual user profiles. This could enhance accuracy by allowing the system to learn and adapt to specific gestures over time. Additionally, integrating multimodal sensing technologies—such as combining visual data from cameras with inertial measurement units (IMUs)—could improve robustness against occlusions and environmental variability.

Expanding the training datasets to include a broader range of gestures performed by diverse user groups will also be crucial for improving model generalization. This inclusivity will help ensure that gesture recognition systems are effective for all users, regardless of their physical capabilities or cultural differences in gesture use.

5. Conclusions

In conclusion, this document presents a comprehensive evaluation of various gesture recognition techniques, including the combined MoveNet and CNN solution, wearable IMU-based systems, and machine learning approaches. The results indicate that the hybrid MoveNet/CNN approach significantly enhances gesture recognition accuracy and reliability, achieving impressive performance metrics across multiple gestures. This method demonstrates the potential to facilitate intuitive interactions in smart home environments, particularly benefiting elderly individuals and those with disabilities.

Overall, the insights gained from evaluating these various approaches provide a solid foundation for ongoing research and development in gesture recognition technology. By continuing to refine these systems, we can create more effective and inclusive solutions that empower users to interact seamlessly with their environments, ultimately enhancing their quality of life and independence.

Acknowledgments: This work has been funded in the context of InclusiveSpaces, a Horizon Europe project supported by the European Commission under Grant Agreement No. 101147881.

References

1. Muneeb, M.; Rustam, H.; Jalal, A. Automate appliances via gestures recognition for elderly living assistance. 2023 4th International Conference on Advancements in Computational Sciences (ICACS). IEEE, 2023, pp. 1–6.
2. Stephan, J.J.; Khudayer, S. Gesture Recognition for Human-Computer Interaction (HCI). *Int. J. Adv. Comp. Techn.* **2010**, *2*, 30–35.
3. Nooruddin, N.; Dembani, R.; Maitlo, N. HGR: Hand-gesture-recognition based text input method for AR/VR wearable devices. 2020 IEEE international conference on systems, man, and cybernetics (SMC). IEEE, 2020, pp. 744–751.
4. Spournias, A.; Faliagka, E.; Skandamis, T.; Antonopoulos, C.; Voros, N.S.; Keramidas, G. Gestures detection and device control in AAL environments using machine learning and BLEs. 2023 12th Mediterranean Conference on Embedded Computing (MECO). IEEE, 2023, pp. 1–5.
5. Faliagka, E.; Skarmintzos, V.; Panagiotou, C.; Syrimpeis, V.; Antonopoulos, C.P.; Voros, N. Leveraging Edge Computing ML Model Implementation and IoT Paradigm towards Reliable Postoperative Rehabilitation Monitoring. *Electronics* **2023**, *12*, 3375.
6. Zhang, Y.; Dong, S.; Zhu, C.; Balle, M.; Zhang, B.; Ran, L. Hand gesture recognition for smart devices by classifying deterministic Doppler signals. *IEEE Transactions on Microwave Theory and Techniques* **2020**, *69*, 365–377.
7. Rashid, A.; Hasan, O. Wearable technologies for hand joints monitoring for rehabilitation: A survey. *Microelectronics Journal* **2019**, *88*, 173–183.
8. Bui, T.D.; Nguyen, L.T. Recognizing postures in Vietnamese sign language with MEMS accelerometers. *IEEE sensors journal* **2007**, *7*, 707–712.
9. Kim, J.H.; Thang, N.D.; Kim, T.S. 3-D hand motion tracking and gesture recognition using a data glove. 2009 IEEE international symposium on industrial electronics. IEEE, 2009, pp. 1013–1018.

10. Kim, S.Y.; Han, H.G.; Kim, J.W.; Lee, S.; Kim, T.W. A hand gesture recognition sensor using reflected impulses. *IEEE Sensors Journal* **2017**, *17*, 2975–2976.
11. Ojeda-Castelo, J.J.; Capobianco-Uriarte, M.d.L.M.; Piedra-Fernandez, J.A.; Ayala, R. A Survey on Intelligent Gesture Recognition Techniques. *IEEE Access* **2022**, *10*, 87135–87156. doi:10.1109/ACCESS.2022.3199358.
12. Oudah, M.; Al-Naji, A.; Chahl, J. Hand gesture recognition based on computer vision: a review of techniques. *journal of Imaging* **2020**, *6*, 73.
13. Köpüklü, O.; Gunduz, A.; Kose, N.; Rigoll, G. Real-time hand gesture detection and classification using convolutional neural networks. 2019 14th IEEE international conference on automatic face & gesture recognition (FG 2019). IEEE, 2019, pp. 1–8.
14. Neethu, P.; Suguna, R.; Sathish, D. An efficient method for human hand gesture detection and recognition using deep learning convolutional neural networks. *Soft Computing* **2020**, *24*, 15239–15248.
15. Chung, H.Y.; Chung, Y.L.; Tsai, W.F. An efficient hand gesture recognition system based on deep CNN. 2019 IEEE International Conference on Industrial Technology (ICIT). IEEE, 2019, pp. 853–858.
16. Sen, A.; Mishra, T.K.; Dash, R. A novel hand gesture detection and recognition system based on ensemble-based convolutional neural network. *Multimedia Tools and Applications* **2022**, *81*, 40043–40066.
17. Park, G.; Chandrasegar, V.K.; Koh, J. Accuracy enhancement of hand gesture recognition using CNN. *IEEE Access* **2023**, *11*, 26496–26501.
18. He, Z.; Jin, L.; Zhen, L.; Huang, J. Gesture recognition based on 3D accelerometer for cell phones interaction. APCCAS 2008-2008 IEEE Asia Pacific Conference on Circuits and Systems. IEEE, 2008, pp. 217–220.
19. Erdaş, Ç.B.; Atasoy, I.; Açıcı, K.; Oğul, H. Integrating features for accelerometer-based activity recognition. *Procedia Computer Science* **2016**, *98*, 522–527.
20. Avola, D.; Cinque, L.; Fagioli, A.; Foresti, G.L.; Fragomeni, A.; Pannone, D. 3D hand pose and shape estimation from RGB images for keypoint-based hand gesture recognition. *Pattern Recognition* **2022**, *129*, 108762.
21. Votel, R.; Li, N. Next-Generation Pose Detection with MoveNet and TensorFlow.js. *TensorFlow Blog* **2021**.