KylinArm: An Arm Gesture Recognition System for Mobile Devices

Shikun Zhao[1], Jingxuan Hong[1,2], Zixuan Zhang[4], Xuqiang Wang[4], Jin Zhang[1,2], and Xiaoli Gong[1,2,3(✉)]

[1] College of Computer Science, Nankai University, Tianjin, China
gongxiaoli@nankai.edu.cn
[2] Tianjin Key Laboratory of Brain Science and Intelligent Rehabilitation, Tianjin, China
[3] State Key Laboratory of High-End Server and Storage Technology, Jinan, China
[4] State Grid Tianjin Information and Communication Company, Tianjin, China

Abstract. Gesture-based Human-Computer Interaction (HCI) has become a primary means of device control due to its naturalness and humanized characteristics, making it applicable for tasks such as drone control and gaming. Gesture recognition using an inertial measurement unit (IMU) has emerged as a major trend in this field. However, due to the intricate nature of the arm structure and the diversity of gestures, relying on a single IMU system for gesture recognition results in limited accuracy. Modern mobile devices, such as smartphones and smartwatches, are equipped with IMUs that allow for convenient data acquisition methods and offer computing resources for deep learning model inference. In this paper, we propose a real-time arm gesture recognition method, called *KylinArm*, which achieves high-precision gesture recognition by coordinating 2 IMUs. The *KylinArm* method is optimized for mobile devices and based on a dual-branch 1D-CNN classifier. It supports the classification of 12 arm gestures with an optimized strategy for mobile devices that have limited computation resources and power supply. Additionally, we adopt an optimization method based on CORrelation ALignment (CORAL) to address the decreasing accuracy that occurs when new users are introduced. Finally, we evaluate *KylinArm* and test it in real scenes, achieving a recognition accuracy of over 98%.

Keywords: Arm Gesture Recognition · Inertial Measurement Unit · Human-Computer Interaction · Mobile Devices · Deep Learning

1 Introduction

As artificial intelligence develops rapidly, Human Activity Recognition (HAR) [22,29] has become an important technology in many fields, including health

This work is supported in part by Natural Science Foundation of China (62172239), Key Technologies R&D Program of Guangdong Province, China (2021B0101310002), and Shandong Provincial Natural Science Foundation, China (ZR2022LZH009).

Z. Tari et al. (Eds.): ICA3PP 2023, LNCS 14488, pp. 92–111, 2024.
https://doi.org/10.1007/978-981-97-0801-7_6

care and human-computer interaction (HCI) [11]. There has been a growing interest in HCI as a research area that caters to the ever-changing needs of human progress with the integration of smart devices into daily lives. Within HCI, gesture recognition, a subfield of HAR, presents a flexible and practical method of transmitting information in complex and dynamic environments.

There are two primary types of gesture recognition: finger gestures and arm gestures. Finger gestures are more complex and may be affected by hand occupation or injury. Arm gestures can free up the hands, improve efficiency, and be used by more individuals to control machines.

Advancements in microelectromechanical system have led to the development of smaller and lighter sensors, making it easier to produce wearable devices based on inertial measurement units (IMU). Customized devices such as wristbands and rings can be used in gesture recognition tasks. However, smartwatches and mobile phones equipped with IMUs offer a cost-effective and convenient way to acquire data for gesture recognition, while also providing an inference environment. This simplifies the hardware composition and makes the HCI process much more natural and user-friendly. The success of projects like EatingTrak [33], MoRSE [14] and the work of Wei et al. [28] demonstrates the feasibility of arm gesture recognition based on IMU in smart mobile devices. However, due to the intricate nature of the arm structure and the diversity of gestures, relying on a single IMU system for gesture recognition results in limited accuracy.

To successfully recognize arm gestures and create a HCI process, obtaining data that comprehensively captures the characteristics of arm gestures is of utmost importance. In this paper, we propose *KylinArm*, a solution for recognizing arm gestures based on dual IMU data collection and optimized for mobile devices to address the aforementioned issues. By leveraging two IMUs located at the wrist and elbow, more data can be obtained to distinguish between arm gestures with similar wrist motion trends.

In *KylinArm*, a dual-branch convolutional neural network (CNN) model is designed that relies on one-dimensional convolution and accounts for data collected by two IMUs to classify 12 arm gestures for HCI with high precision. Additionally, *KylinArm* is optimized for the limited computing resources and energy supply of mobile devices by integrating a Wakeup-Detection-Classification module.

To enhance the generalization of arm gesture recognition, a real-time data alignment strategy based on CORrelation ALignment (CORAL) [25] is implemented. This is necessary as different users and the variance of sensor positions may change the features of data collected during gesture recognition.

The main contributions of this paper are as follows:

- We propose a new arm gesture recognition framework called *KylinArm*. This framework supports 12 arm gestures using 2 IMUs for data collection and a dual-branch convolutional neural network (CNN) model for classification.
- We optimize *KylinArm* for ubiquitous mobile devices with a Wakeup-Detection-Classification mechanism and real-time data alignment strategy, which ensures efficient utilization of limited computing resources and energy supply.

– We implement *KylinArm* on Android-based mobile devices and evaluate its effectiveness by testing with collected datasets and real-world scenarios. The results demonstrate the high precision and robustness of *KylinArm* in recognizing arm gestures.

The rest of the paper is organized as follows. Section 2 presents related works on IMU-based arm gesture recognition and algorithms for sensor-based gesture recognition. Section 3 provides details on the design and implementation of *KylinArm*. Section 4 evaluates the proposed arm gesture recognition method. Finally, Sect. 5 concludes the paper.

2 Related Work

2.1 IMU-Based Gesture Recognition

IMU typically includes three sensors: an accelerometer, a gyroscope, and a magnetometer, each of which measures data along the XYZ axes. The magnetometer is an optional component, and the IMU without it is commonly referred to as a 6-axis IMU. The accelerometer measures linear acceleration, while the gyroscope measures angular velocity.

IMUs are capable of accurately capturing the inertial data generated during the execution of gestures. Bianco et al. [2] used an IMU wristband to classify 6 letters, 6 numbers, and 12 simple forearm gestures. Cui et al. [6] used three 6-axis IMUs to recognize three arm activities. Kim et al. [15] identified handwritten digits with a handheld IMU.

With the advancement of mobile devices, smartphones and smartwatches are often equipped with built-in IMUs and possess stronger computing capabilities, which can support deep learning model inference. Kurz et al. [17] utilized a self-assembled 6-axis IMU and a smartwatch to collect data and successfully recognize 12 different gestures on a smartphone. Kang et al. [12] used a smartwatch to recognize 7 gestures while walking. Guo et al. [7] used a smartwatch and a camera to recognize 12 hand gestures. Kasnesis et al. [14] recognized 5 arm gestures representing different emergency signals using the 6-axis IMU in a smartwatch.

2.2 Algorithms for Sensor-Based Gesture Recognition

The design of the classifier plays a crucial role in achieving high-accuracy gesture recognition. Classifiers based on time-series data collected from IMUs can be categorized into two groups: traditional machine learning and deep learning. Common machine learning methods include KNN [8], Random Forest [10] and SVM [4,23,27], while deep learning methods can be further divided into three types: CNN [1,5,14,26], RNN [2,7], and their combinations [21,31]. Ulysse et al. [5] proposed a CNN model based on continuous wavelet transform to classify 7 gestures. Bianco et al. [2] built an arm gesture recognition system based on GRU and LSTM. Yuan et al. [31] utilized the CNN+LSTM method to recognize sign language.

Moreover, the generalization of classifiers is also an important issue. Both [2] and [30] discovered that deep learning models can differentiate the same gesture done by different people. [4] and [14] noted that the accuracy of arm gesture recognition is affected when using the model with people whose motion data are not included in training, but they only described the phenomenon without proposing improvements. In [7], gesture recognition was performed on smartwatch. To alleviate the impact of new users on accuracy, data was re-collected and trained on the server. Although effective, this process can be cumbersome.

3 Design and Implementation of KylinArm

3.1 Framework Design of KylinArm

We develop an arm gesture recognition system, named *KylinArm*, that is designed to achieve high-precision recognition and perform real-time inference on mobile devices. The overall structure of *KylinArm* is illustrated in Fig. 1.

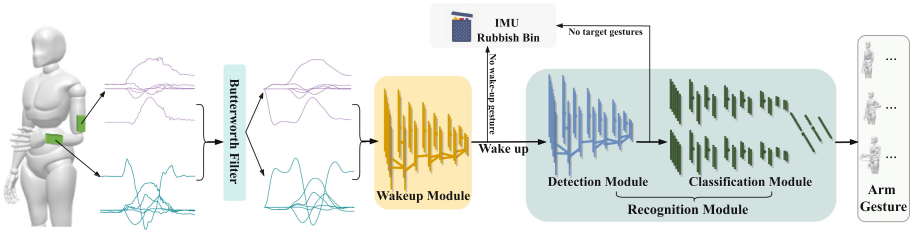


Fig. 1. Framework of KylinArm

To ensure accurate capture of data that represents the differ characteristics of arm gestures, we adopt a dual-IMU data acquisition mechanism. Specifically, the user wears two 6-axis IMUs, one on the wrist and the other on the upper arm near the elbow. The IMU data is collected at a sampling frequency of 30 Hz. The classifier in *KylinArm*, which is referred to as the Classification Module in Fig. 1. From the acquisition of signals to input into the classifier, the continuous IMU data is filtered using a low-pass Butterworth filter to remove noise caused by the environment and equipment. In addition, a sliding window with a size of 3 s and a step of 200 ms is used to segment the continuous data.

KylinArm is set to sleep mode by default to conserve computing resources and prevent unintended operations. Wakeup Module is responsible for determining when the recognition procedure should be activated. The preprocessed data windows are initially directed to Wakeup Module to determine if they constitute a wake-up gesture. However, if a wake-up gesture is detected, Wakeup Module stops working and the recognition procedure starts. Once the recognition procedure is woken up, windows are fed directly into Recognition Module. The

windows are then processed by Detection Module to prepare input for Classification Module, with only one window being input to Classification Module for each arm gesture. Finally, the recognized arm gesture type is output as a control command.

3.2 Gesture-Controlled Command Set

Gestures are fundamental to HCI, and the design of arm gestures should be tailored to the actual operational requirements of the devices. It is essential to consider the distinction between control gestures and daily arm gestures as well. We are inspired by Zhang et al.'s study [32] on the naturalness of upper limb movements and design twelve arm gestures that involve movement of the entire arm, which are illustrated in Fig. 2. The correspondence between these gestures and their respective potential control commands is also shown in the same figure.

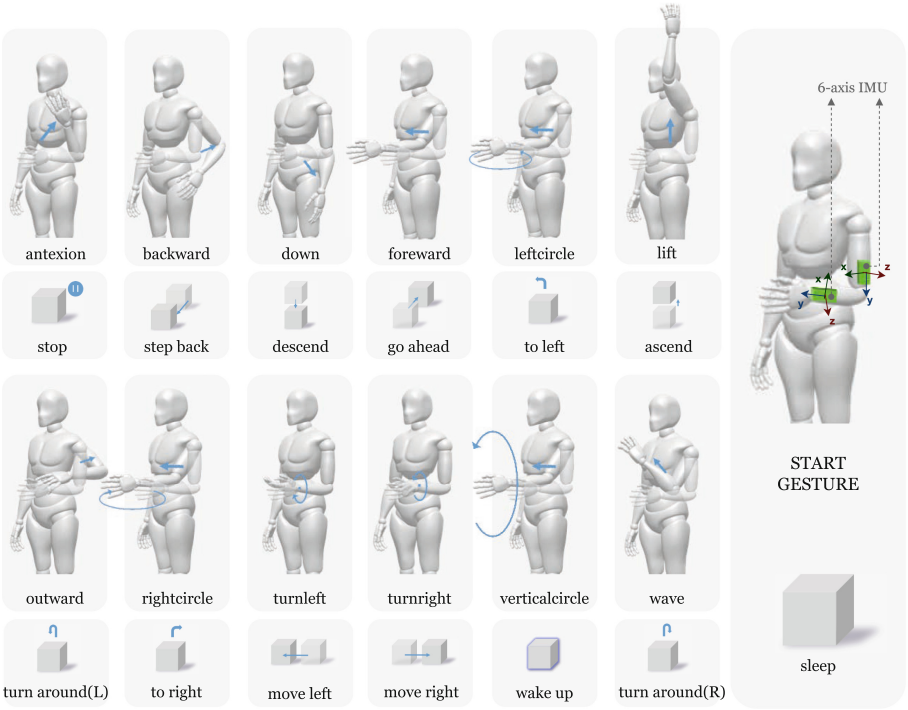


Fig. 2. Designed arm gestures and corresponding control commands

3.3 Classification Module: Dual-Branch 1D-CNN Classifier

For the 12-class arm gesture recognition task, we design a classification model named dual-branch 1D-CNN (*Dual-CNN*). The model is based on one-dimensional (1D) convolution and uses two convolutional branches to extract

features from the time-series data obtained from each IMU to enhance classification accuracy. The network structure is shown in Fig. 3 and more detail can be found in Table 1. This model is applied in Classification Module to classify arm gestures. During training, only windows containing the 12 arm gestures are used.

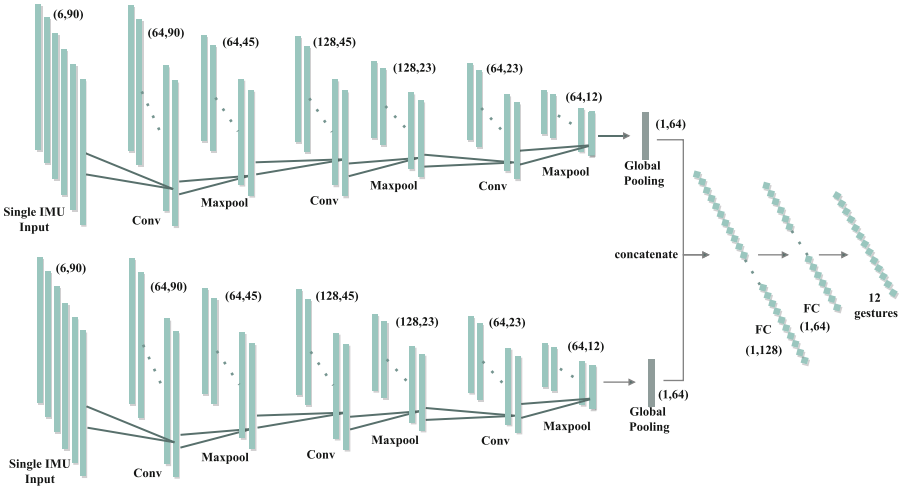


Fig. 3. Structure of Dual-CNN

Dual-CNN is designed to run on a mobile device and is composed of two convolutional branches, each of which includes three 1D convolutional layers. The input data of each branch has a shape of (6,90). After convolution, a MaxPooling [20] layer, a Batch Normalization (BN) [9] layer and a Dropout [24] layer are applied. The MaxPooling and Dropout layers serve to prevent model overfitting, while the BN layer accelerates the model's convergence speed. Global Average Pooling (GAP) [19] is then applied to reduce the number of parameters in *Dual-CNN*. The concatenated features are fed into two fully connected (FC) layers, which map extracted features to classification labels as the final output.

3.4 Arm Gesture Recognition

To adapt limited computing resources of mobile devices, we propose a Wakeup-Detection-Classification process. This process utilizes *Dual-CNN* discussed in Sect. 3.3 for arm gesture classification, as well as two much lighter models trained separately for the Wakeup Module and Detection Module.

Wakeup Module. utilizes a binary classification model, called CNN_{wake}, which is lightweight and efficient. CNN_{wake} consists of three 1D convolutional

Table 1. Details of Dual-CNN

Layer	Layer Type	Input Size	Output Size	Kernel Size	Activation
Layer 1-1	1DConvolution	(6,90)	(64,90)	8	ReLU
	Batch Normalization				
	Dropout				
Layer 1-2	1DMaxpooling	(64,90)	(64,45)	3	
Layer 1-3	1DConvolution	(64,45)	(128,45)	5	ReLU
	Batch Normalization				
	Dropout				
Layer 1-4	1DMaxpooling	(128,45)	(128,23)	3	
Layer 1-5	1DConvolution	(128,23)	(64,23)	3	ReLU
	Batch Normalization				
	Dropout				
Layer 1-6	1DMaxpooling	(64,23)	(64,12)	3	
	Global Average Pooling	(64,12)	(1,64)		
Layer 2-1	1DConvolution	(6,90)	(64,90)	8	ReLU
	Batch Normalization				
	Dropout				
Layer 2-2	1DMaxpooling	(64,90)	(64,45)	3	
Layer 2-3	1DConvolution	(64,45)	(128,45)	5	ReLU
	Batch Normalization				
	Dropout				
Layer 2-4	1DMaxpooling	(128,45)	(128,23)	3	
Layer 2-5	1DConvolution	(128,23)	(64,23)	3	ReLU
	Batch Normalization				
	Dropout				
Layer 2-6	1DMaxpooling	(64,23)	(64,12)	3	
	Global Average Pooling	(64,12)	(1,64)		
Layer 3	Fully-Connected	(1,128)	(1,64)		
	Dropout				
Output Layer	Fully-Connected	(1,64)	12		Softmax

layers and two fully connected layers. Each convolutional layer is followed by a Maxpooling layer, a BN layer, and a Dropout layer. With only 4994 parameters, CNN_{wake} is better suited for frequent use on mobile devices compared to *Dual-CNN*.

The gesture *verticalcircle* (shown in Fig. 2) is chosen as the wake-up gesture. This particular gesture is relatively specific and less commonly used in daily life, which makes it a suitable choice for ensuring the reliability of Wakeup Module. CNN_{wake} takes IMU data as input by the sliding window with a size of 3 s and a step of 200 ms. To improve the accuracy of capturing the wake-up gesture, a window will be classified as positive if it contained data of a gesture for more than 80%, and negative otherwise. During the *verticalcircle* gesture, CNN_{wake} outputs consecutive 1 s. If 0 appears, it indicates that the gesture has not occurred or has been completed. Therefore, we define the occurrence of *verticlecircle* as CNN_{wake} outputting 3 or more consecutive 1.

Detection Module. is responsible for capturing windows that contain the complete gesture as much as possible and ensuring that the gesture classifier only runs when necessary. This helps to reduce the computational cost of the Recognition Module and improve the accuracy of arm gesture recognition. Detection Module uses a binary classification model on the IMU data processed by sliding window mechanism, determining whether it contains data of one of the 12 arm

Algorithm 1. Main Window Selection Mechanism

Input: The results of Detection(0 or 1), Classifier Dual-CNN
Output: The inference result of Recognition

```
if (0 appear and len(pool)≠0) or len(pool)≥7 then
    if len(pool) is odd then
        center_idx = ⌊len(pool) / 2⌋
    else
        center_idx = len(pool) / 2 - 1
    end if
    standard_window = pool[center_idx]          ▷ Select the main window
    result = Dual-CNN(standard_window)          ▷ Classify the selected window
    clear pool
else if 1 appear then
    append current window to pool               ▷ Accumulate windows
end if
```

gestures. It is similar to the role of CNN_{wake}. Therefore, Detection Module utilizes the same model structure as CNN_{wake}, which called CNN_{detect}. When the user performs one of the 12 gestures, CNN_{detect} first outputs 0, then outputs a continuous segment of 1, and finally returns to outputting 0. The Main Window Selection (MWS) mechanism is employed to provide the window that contains as much gesture data as possible for Classification Module.

As illustrated in Algorithm 1, when an arm gesture is performed, CNN_{detect} outputs consecutive 1 s and the corresponding windows are added to the window pool with a maximum size of 7. Window accumulating process continues until either a zero appears or the pool reaches its maximum size. Subsequently, a method similar to finding the median is applied to select the main window, which is then passed to Classification Module. The pool is cleared and MWS mechanism awaits the next sequence of 1 s.

3.5 CORAL-Based Generalization Optimization

Arm gesture recognition is a user-sensitive task. There are distributional differences even in the performance of the same gesture by different users. Arm gesture classifier may not perform optimally on data from new users that have not been previously trained on. In the case of data with different distributions, Domain Adaption [16] typically involves mapping data from both the source and target domains to a common distribution space through feature changes. And then the migrated data is used to train the model improving its performance on the target domain data. However, implementing this process solely on mobile devices is often infeasible. Consequently, we propose a new real-time generalization strategy based on CORAL [25], named as $CORAL_{REVERSE}$, which shifts the focus from a model-to-data fit to a data-to-model fit approach.

$CORAL_{REVERSE}$ migrates the new data (target domain) to the original data (source domain) in real time. As outlined in Algorithm 2, the new users' data is represented by D_t, while D_s represents the original data used to train

the model. The process of data alignment involves two steps: first, the new data D_t is whitened, and then, the whitened data is re-colored using the statistical characteristics of original data D_s.

Algorithm 2. $CORAL_{REVERSE}$

Input: Source Data D_s, New Data D_t
Output: Transferred New Data $D_t{}^*$
1: $C_s = cov(D_s) + eye(size(D_s, 2))$
2: $C_t = cov(D_t) + eye(size(D_t, 2))$
3: $D_t = D_t * C_t^{-\frac{1}{2}}$ ▷ Whitening New Data D_t
4: $D_t^* = D_t * C_s^{\frac{1}{2}}$ ▷ Re-coloring New Data D_t

IMU data is continuously input to the inference system. Each new data needs to choose a source domain to perform whitening and re-coloring, which produce a lot of repetitive work. Since $CORAL_{REVERSE}$ implements data migration based on two matrix multiplication operations, the migration matrix $coral_t$, which represents the data gap between the target and source domains, is calculated in advance.

$$coral_t = C_t^{-\frac{1}{2}} \cdot C_s^{\frac{1}{2}} \tag{1}$$

where C_t is the feature matrix of gesture data for new users, and C_s is the feature matrix of original data. During the real-time inference process, $coral_t$ is dot-producted with the continuously IMU windows to align the new data with the original data in real-time. It improves the overall recognition accuracy of Recognition Module.

To effectively employ transfer learning, it is crucial to identify a source domain that is similar to the target domain. Therefore, for new users with different characteristics, we utilize different data transfer strategies for data migration that incorporated the migration matrix calculation method. The method of using $CORAL_{REVERSE}$ in $KylinArm$ is shown in Algorithm 3.

The $coral_t$ matrix is calculated prior to performing real-time data alignment. Therefore, a gesture sample set containing 12 gestures, denoted as $D_t^{'}$, is captured in advance to implement Algorithm 3. $D_t^{'}$ contains one item for each gesture. Then, the source domain data is selected based on the distribution of the pre-collected gestures set $D_t^{'}$. The classification accuracy of $D_t^{'}$ is used as the criterion for selecting the data transfer strategy. These two data transfer strategies are referred to as $Select\_Person$ and $Select\_Overall$ in Algorithm 3. If the accuracy exceeds threshold A, the former strategy is chosen; otherwise, the latter is chosen.

D_O contains all gesture data for all individuals which is the training dataset of the gesture classifier. In $Select\_Overall$ strategy, the global source domain data D_s is fixed and a K-means [18] method is used to select representative data from the training dataset for Classification Module. D_s contains the original data of each class of gestures for each person. In $Select\_Person$ strategy, D_s

Algorithm 3. New Data Transfer Process

Require: Overall Original Data D_O, Sample Target Data $D_t^{'}$, Classifier *Recognition*
Input: Real-time IMU Windows D_{IMU}
Output: Transferred Real-time IMU Windows D_{IMU}^{*}

$label_{predict} = Recognition(D_t^{'})$
$accuracy = accuracy\_score(lable_{sample}, label_{predict})$
if $accuracy > A$ **then**
    Source Data $D_s = Select\_Person(D_O)$
else
    Source Data $D_s = Select\_Overall(D_O)$
end if
for act_{t_i} in $D_t^{'}$ **do**
    Select action data act_{s_i} with the same label as act_{t_i} from D_s
    $coral_{t_i} = CORAL_{REVERSE}(act_{t_i}, act_{s_i})$
end for
$CORAL_t = Average\{coral_{t_0}, coral_{t_1}, \ldots\}, coral_{t_{11}}$
$D_{IMU}^{*} = D_{IMU} \cdot CORAL_t$

includes data from the person who is most similar to $D_t^{'}$. We adapt a classification accuracy-based method to assess the similarity. Specifically, original data samples from each person are used as training sets in turn to train corresponding classifiers. $D_t^{'}$ of new user is then put into every classifier to obtain the accuracy. The data from the person whose classifier achieves the highest accuracy is selected as the source domain D_s.

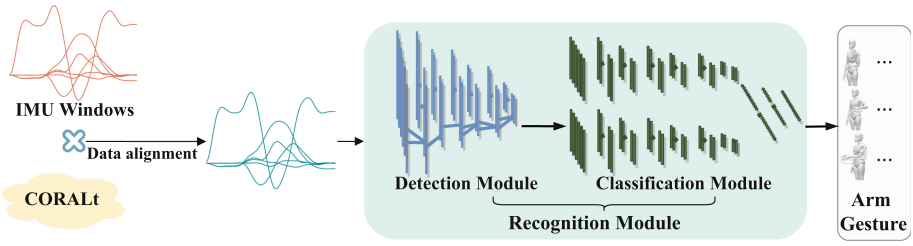


Fig. 4. Usage of the transformation migration matrix $CORAL_t$

After selecting the source domain data, $CORAL_{REVERSE}$ is used to calculate migration matrix $CORAL_t$. In the process of calculating $CORAL_t$, the source and target domain data with the same label are calculated according to Algorithm 1 to obtain $coral_{t_i}$. $CORAL_t$ is computed as the average of multiple $coral_{t_i}$ matrices. Once the feature linear transformation migration matrix $CORAL_t$ is obtained, $CORAL_{REVERSE}$ can be used to perform whitening and re-coloring operations via dot product between $CORAL_t$ and D_{IMU}, resulting in the aligned data D_{IMU}^{*}. And then D_{IMU}^{*} is inputted into Recognition Module for gesture recognition, as shown in Fig. 4.

4 Evaluation

To fully validate the effectiveness of *KylinArm*, we evaluate the performance of *Dual-CNN* and each module in *KylinArm* using our datasets. At last, we implement it on smartphones and test the system in a real scene.

4.1 DataSet Description

We used two IMUs placed on the wrist and upper arm to collect data from 15 volunteers (9 males and 6 females) with ages ranging from 22 to 27 years, heights ranging from 160–185 cm, and weights ranging from 50–80 kg. Data was collected at a frequency of 30 Hz, with each gesture performed for 3 s. Both Dataset for *Dual-CNN* and Dataset for *KylinArm* are derived from the raw IMU data collected, and they are processed separately and labeled in different ways to accommodate different task needs. The former dataset is only used to train and test the designed classification model. The latter dataset simulates the data generated in real-world scenarios and contains windows of continuous time, which is used to train and test each module in the system inference phase.

Dataset for *Dual-CNN* only contains the data of 12 gestures. Each of the 15 volunteers repeated each action 10 times according to the rule of performing for 3 s and resting for 3 s. The final dataset contains 1800 items, where each item has the shape of (12,90).

Dataset for *KylinArm* includes data collected from 15 volunteers, each of whom performed each gesture 10 times within a 10-minute time frame. The dataset contains a total of 1800 gestures and spans a duration of 3 h. To segment the continuous IMU signals, we employed a sliding window approach with a size of 3 s and a step of 200 ms. A window was labeled as 1 if it overlapped with gesture data for more than 80% of its duration, and as 0 otherwise. Therefore, Dataset for *KylinArm* contains 46230 windows data, of which there are 7605 positive windows and 38625 negative windows.

4.2 Performance of Dual-Branch 1D-CNN Model

5-fold cross-validation is performed on five baseline models and *Dual-CNN* using Dataset for *Dual-CNN* to evaluate classification accuracy. Adam optimizer and cross-entropy loss function are used, batch size is 8, and 50 epochs are trained. The single-branch feature extraction variant of *Dual-CNN* is 1D-CNN, which takes the data from two IMUs and concatenates them as input. CNN-LSTM [34] and EMGHandNet [13] perform well in HAR tasks.

To provide a comprehensive assessment of the effectiveness of each model, we utilized four commonly-used metrics: Accuracy, Precision, Recall, and F1 score. The results of the evaluation, which include the *Dual-CNN* model and five baseline models, are presented in Table 2. The *Dual-CNN* model outperforms all other models across all metrics. These results highlight the significant difference in classification ability between machine learning and deep learning methods.

Table 2. The global 5-fold cross-validation results for the different models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
SVM	91.61 ± 2.04	92.58 ± 2.44	91.61 ± 2.04	91.61 ± 2.03
Decision Tree	91.94 ± 1.65	92.19 ± 1.61	91.94 ± 1.65	91.94 ± 1.67
1D-CNN	99.56 ± 0.33	99.55 ± 0.35	99.53 ± 0.31	99.53 ± 0.34
CNN-LSTM [34]	99.67 ± 0.22	99.67 ± 0.20	99.65 ± 0.22	99.65 ± 0.21
EMGHandNet [13]	99.50 ± 0.21	99.52 ± 0.20	99.50 ± 0.21	99.50 ± 0.21
Dual-CNN	$\mathbf{99.78 \pm 0.27}$	$\mathbf{99.78 \pm 0.26}$	$\mathbf{99.78 \pm 0.27}$	$\mathbf{99.78 \pm 0.27}$

While the accuracy of all tested deep learning models exceeded 99%, our proposed *Dual-CNN* model stands out as the most efficient solution. It has the least number of parameters and fewer FLOPs as shown in Table 3. We then evaluate the inference time cost of *Dual-CNN* on the Honor 30 smartphone, which runs on HarmonyOS 2.0, has a memory size is 8.0 GB and is powered by Kirin 985 processor. *Dual-CNN* takes an average of less than 5ms to perform an inference on Honor 30. The efficiency of the *Dual-CNN* model makes it more mobile device-friendly and enables real-time gesture recognition with greater accuracy.

Table 3. FLOPs and parameters of four deep learning model

Model	FLOPs	Parameters
1D-CNN	3087616	172428
CNN-LSTM [34]	10816928	504748
EMGHandNet [13]	3572199680	2634572
Dual-CNN	**5482752**	**147788**

To evaluate the generalization, we performed 15-fold cross-validation by selecting one individual's data as the test set and the data of the other 14 individuals as the training set. The model settings, training settings are the same as 5-fold cross-validation. The results are shown in Table 4 and Fig. 5. *Dual-CNN* exhibits superior generalization performance when facing untrained new data. Compared to the results of the 5-fold cross-validation, *Dual-CNN* has the least decrease in accuracy when classifying unfamiliar data. While the results of 1D-CNN is comparable to that of *Dual-CNN*, merging feature extraction appears to weaken it's generalization ability across individuals.

Table 4. Results of the 15-fold cross-validation for different models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
SVM	80.83	84.27	80.83	77.88
Decision Tree	75.00	74.79	75.00	71.65
1D CNN	94.61	95.58	94.61	93.68
CNN-LSTM [34]	93.50	94.01	93.50	92.49
EMGHandNet [13]	93.94	94.65	93.94	93.66
Dual-CNN	**96.00**	**96.91**	**96.00**	**95.33**

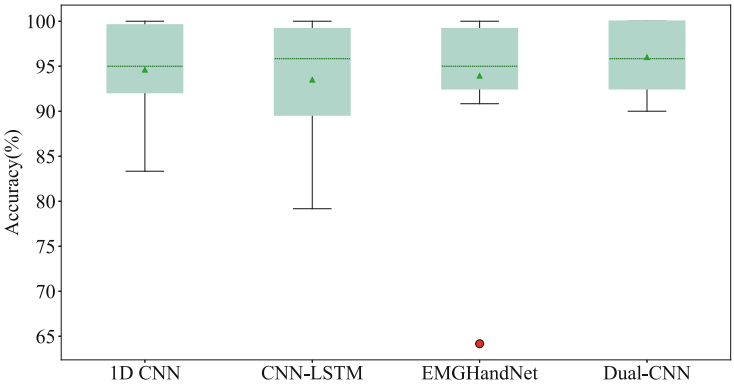


Fig. 5. Accuracy comparison of four deep learning models in 15-fold cross-validation (triangles represent means and dotted lines represent medians)

4.3 Performance of Inference Modules

Wakeup Module. The CNN_{wake} is trained based on Dataset for *KylinArm*. However, due to the fact that the positive examples of Wakeup Module are only related to windows of *verticalcircle*, the number of positive examples is greatly reduced. This results in severe imbalances between positive and negative labels, which significantly impacts the model's discrimination ability. To address this issue, we use SMOTE [3] to increase the number of positive examples. Wakeup Module achieves an accuracy of 100% for *verticalcircle* in a 5-fold cross-validation experiment conducted on Dataset for *KylinArm*. When using all data from a person as the test set and using the remaining data as the training set, the recognition accuracy of the module is 98.46%. The complexity of *verticalcircle* makes it relatively easy to distinguish. Additionally, SMOTE can increase the diversity of data, which has a positive effect on compensating for differences between individuals' gestures.

Recognition Module. The CNN_{detect} is trained using Dataset for *KylinArm*. To fully validate the effectiveness and generalization of Recognition Module, we designed the same 5-fold and 15-fold cross-validation experiments as in Sect. 4.2.

As shown in Table 5, continuous windows containing 360 gestures are inputed into Recognition Module for test. Detection Module achieves an accuracy of 98.67%, while Classification Module achieves an accuracy of 99.72%. As a result, Recognition Module exhibited an overall accuracy of 98.39% for recognizing gestures. Table 6 shows the results of 15-fold cross-validation. We can find that the accuracy of Detection Module is 96.72%, which is 1.95% lower than the 5-fold cross-validation result. The accuracy of Classification Module remains almost unchanged at 99.54%. The overall accuracy of Recognition Module decreases by 2.11% to 96.28%. While Detection Module alleviates the performance degradation caused by the sensitivity of gesture classifier to unfamiliar data, data not included in the training dataset still caused Detection Module to miss a certain number of windows, limiting the high-precision classification ability of Recognition Module.

Table 5. Accuracy (%) of global 5-fold cross-validation

Fold #	*Detect.*	*Classif.*	*Recogn.*
0	100	99.44	99.44
1	99.44	99.16	98.61
2	97.78	100	97.78
3	96.94	100	96.94
4	99.17	100	99.17
avg.	98.67	99.72	98.39

Table 6. Accuracy (%) of 15-fold cross-validation

Person	*Detect.*	*Classif.*	*Recogn.*
P_0	96.67	100	96.67
P_1	100	100	100
P_2	100	100	100
P_3	99.17	99.16	98.33
P_4	100	100	100
P_5	94.17	100	94.17
P_6	100	99.17	99.17
P_7	100	100	100
P_8	100	100	100
P_9	94.17	99.12	93.33
P_{10}	79.17	98.95	78.33
P_{11}	98.33	100	98.33
P_{12}	89.17	100	89.17
P_{13}	100	100	100
P_{14}	100	96.67	96.67
avg.	96.72	99.54	96.28

4.4 Performance of $CORAL_{REVERSE}$

The experiment primarily focuses on verifying the effect of adding $CORAL_{REVERSE}$ to Recognition Module on the recognition of gestures. As described in Sect. 3.5, we firstly pre-collect 12 gestures as samples, which are then verified using the Classification model for initial distribution verification. If the accuracy is close to 100%, samples are aligned with 12 gestures of one individual. Otherwise, they are aligned with the global source gesture library. The global library contains data for each arm gesture performed by each volunteer, resulting in a total dataset size of 12×15.

By adding $CORAL_{REVERSE}$ to Recognition Module, the rate of gestures captured by Detection Module increased from 96.72% to 97.89%, while the overall gesture recognition accuracy increased from 96.28% to 97.56%. The detailed results of each test group are presented in Fig. 6. The results indicate that 7 groups exhibited varying degrees of improvement, while 6 groups remained unchanged. Among them, P9 and P10 have an accuracy increase of 6.67% and 7.50%, respectively. However, P6 and P8 experienced a decrease in the number of recognized gestures by 2. The reduction in accuracy observed in the P6 and P8 groups can be attributed to the transformation of accurately classifiable gesture data during data alignment based on $CORAL_{REVERSE}$. This process has the potential to convert the data into other similar gestures when the arm gesture is not standard enough.

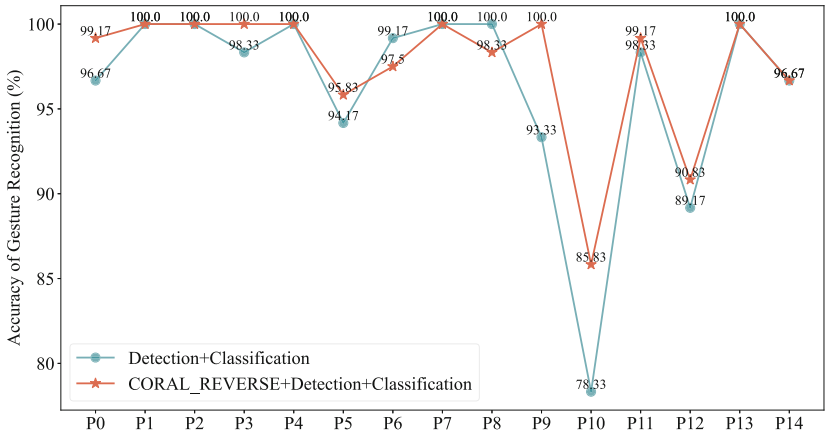


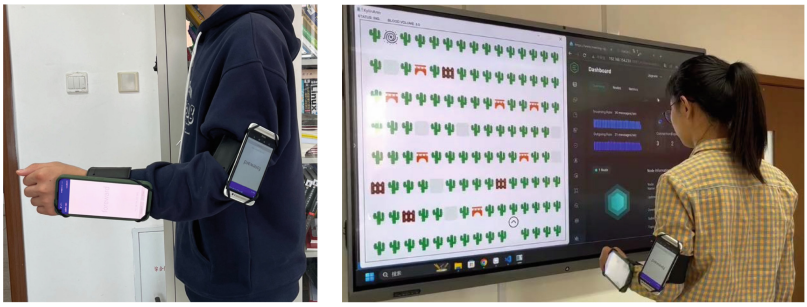
Fig. 6. Comparison of recognition accuracy before and after generalization optimization

4.5 Real-World Evaluation of KylinArm

To assess the real-world performance of *KylinArm*, we deploy it on two Android mobile phones. Figure 7 shows the arm gesture recognition system, which consists of two IMUs provided by respective mobile phones. The system utilizes

MQTT for communication with the broker running on a laptop. The mobile phone located on the upper arm is responsible for collecting IMU data, while that located on the wrist handles preprocessing of IMU data, as well as the recognition of arm gestures. The recognition result is then transmitted to the target device. The mobile application running on the mobile phones plays a crucial role in the system. It is responsible for preprocessing the IMU data and running *KylinArm* for arm gesture recognition. The main GUI of mobile application is presented in Fig. 8(a) whereas Fig. 8(b) shows the GUI where new users can perform gesture calibration to achieve data alignment before using the system.

First of all, five volunteers are recruited to test the arm gesture recognition system. After familiarizing themselves with the actions, each volunteer performed 12 actions according to their own ideas, ensuring that each gesture was completed 10 times. Table 7 presents the statistical results, which indicate that the arm gesture recognition accuracy of *KylinArm* is 98.5%.

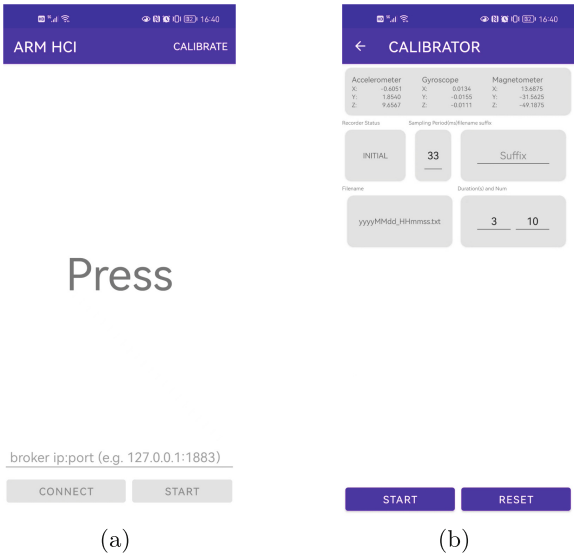


(a) (b)

Fig. 7. System test (As an illustration, we implement KylinArm on two smartphones. However, the use of two smartphones is not necessarily required, one of them can be replaced by a smartwatch or other IMU-equipped device.): (a) IMUs Position (b) Interact with the game through arm gestures

Table 7. Accuracy of arm gesture recognition system

User	Num of correct	Accuracy (%)
1	120	100
2	118	98.33
3	118	98.33
4	116	96.67
5	119	99.17
avg		98.50



(a) (b)

Fig. 8. Arm gesture recognition system app running KylinArm on Android smartphones: (a) Main GUI (b) Pre-collected for data alignment

We also developed a PC interactive game for further evaluation. All components are under the same local area network. The game character's behavior corresponds to 11 arm gestures, and the gesture *verticalcircle* used to wake up the recognition procedure is also used to control the start and end of the game. As shown in Fig. 7(b), volunteers are able to control the game character to reach the end point without any prompts. A video which demonstrating the operation process of each arm gesture during the game, as well as the game instructions, can be found in the GitHub repository[1]. KylinArm running on smartphones continued to provide normal services throughout this process, fully demonstrating its stability and usability.

5 Summary

We propose a new framework called *KylinArm*, which is designed to recognize arm gestures and is optimized for mobile devices. This framework employs two IMUs for data collection and a dual-branch CNN model called *Dual-CNN* for classification, which enables the high-accuracy classification of 12 arm gestures. In order to achieve real-time arm gesture recognition and conserve resources, we have optimized *KylinArm* for ubiquitous mobile devices by implementing a Wakeup-Detection-Classification mechanism and a real-time data alignment strategy. As a result, the *KylinArm* system can operate effectively on mobile devices that have constrained computing resources and energy supply. Finally,

[1] https://github.com/NKU-EmbeddedSystem/KylinArm.

we have implemented *KylinArm* on Android-based mobile devices and evaluated its effectiveness by testing it with collected datasets and real-world scenarios. The results demonstrate the high precision and robustness of *KylinArm* in recognizing arm gestures.

References

1. Bhattacharya, D., Sharma, D., Kim, W., Ijaz, M.F., Singh, P.K.: Ensem-HAR: an ensemble deep learning model for smartphone sensor-based human activity recognition for measurement of elderly health monitoring. Biosensors **12**(6), 393 (2022)
2. Bianco, S., Napoletano, P., Raimondi, A., Rima, M.: U-wear: User recognition on wearable devices through arm gesture. IEEE Transactions on Human-Machine Systems **52**(4), 713–724 (2022)
3. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
4. Colli Alfaro, J.G., Trejos, A.L.: User-independent hand gesture recognition classification models using sensor fusion. Sensors **22**(4), 1321 (2022)
5. Côté-Allard, U., et al.: Deep learning for electromyographic hand gesture signal classification using transfer learning. IEEE Trans. Neural Syst. Rehabil. Eng. **27**(4), 760–771 (2019)
6. Cui, J.W., Li, Z.G., Du, H., Yan, B.Y., Lu, P.D.: Recognition of upper limb action intention based on IMU. Sensors **22**(5), 1954 (2022)
7. Guo, K., Zhou, H., Tian, Y., Zhou, W., Ji, Y., Li, X.Y.: Mudra: a multi-modal smartwatch interactive system with hand gesture recognition and user identification. In: IEEE INFOCOM 2022-IEEE Conference on Computer Communications, pp. 100–109. IEEE (2022)
8. Hellara, H., et al.: Classification of dynamic hand gestures using multi sensors combinations. In: 2022 IEEE 9th International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), pp. 1–5. IEEE (2022)
9. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456. PMLR (2015)
10. Ji, L., Liu, J., Shimamoto, S.: Recognition of Japanese sign language by sensor-based data glove employing machine learning. In: 2022 IEEE 4th Global Conference on Life Sciences and Technologies (LifeTech), pp. 256–258. IEEE (2022)
11. Jindal, S., Sachdeva, M., Kushwaha, A.K.S.: Deep learning for video based human activity recognition: review and recent developments. In: Bansal, R.C., Zemmari, A., Sharma, K.G., Gajrani, J. (eds.) Proceedings of International Conference on Computational Intelligence and Emerging Power System. AIS, pp. 71–83. Springer, Singapore (2022). https://doi.org/10.1007/978-981-16-4103-9_7
12. Kang, P., Li, J., Fan, B., Jiang, S., Shull, P.B.: Wrist-worn hand gesture recognition while walking via transfer learning. IEEE J. Biomed. Health Inform. **26**(3), 952–961 (2021)
13. Karnam, N.K., Dubey, S.R., Turlapaty, A.C., Gokaraju, B.: EMGHandNet: a hybrid CNN and Bi-LSTM architecture for hand activity classification using surface EMG signals. Biocybern. Biomed. Eng. **42**(1), 325–340 (2022)
14. Kasnesis, P., Chatzigeorgiou, C., Kogias, D.G., Patrikakis, C.Z., Georgiou, H.V., Tzeletopoulou, A.: MoRSE: deep learning-based arm gesture recognition for search and rescue operations. arXiv preprint arXiv:2210.08307 (2022)

15. Kim, M., Cho, J., Lee, S., Jung, Y.: IMU sensor-based hand gesture recognition for human-machine interfaces. Sensors **19**(18), 3827 (2019)
16. Kouw, W.M., Loog, M.: An introduction to domain adaptation and transfer learning. arXiv preprint arXiv:1812.11806 (2018)
17. Kurz, M., Gstoettner, R., Sonnleitner, E.: Smart rings vs. Smartwatches: utilizing motion sensors for gesture recognition. Appl. Sci. **11**(5), 2015 (2021)
18. Likas, A., Vlassis, N., Verbeek, J.J.: The global k-means clustering algorithm. Pattern Recogn. **36**(2), 451–461 (2003)
19. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
20. Nagi, J., et al.: Max-pooling convolutional neural networks for vision-based hand gesture recognition. In: 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), pp. 342–347. IEEE (2011)
21. Nan, Y., Lovell, N.H., Redmond, S.J., Wang, K., Delbaere, K., van Schooten, K.S.: Deep learning for activity recognition in older people using a pocket-worn smartphone. Sensors **20**(24), 7195 (2020)
22. Punithavathi, D., Janakiraman, R., Santhoshkumar, S., Srikanth, R.: Human activity recognition using deep learning techniques: a review. J. Ambient. Intell. Humaniz. Comput. **12**(6), 5669–5695 (2021)
23. Shahzad, W., Ayaz, Y., Khan, M.J., Naseer, N., Khan, M.: Enhanced performance for multi-forearm movement decoding using hybrid IMU-SEMG interface. Front. Neurorobot. **13**, 43 (2019)
24. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
25. Sun, B., Feng, J., Saenko, K.: Return of frustratingly easy domain adaptation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016)
26. Tam, S., Boukadoum, M., Campeau-Lecours, A., Gosselin, B.: A fully embedded adaptive real-time hand gesture classifier leveraging HD-SEMG and deep learning. IEEE Trans. Biomed. Circuits Syst. **14**(2), 232–243 (2019)
27. Wahid, M.F., Tafreshi, R., Al-Sowaidi, M., Langari, R.: Subject-independent hand gesture recognition using normalization and machine learning algorithms. J. Comput. Sci. **27**, 69–76 (2018)
28. Wei, W., Kurita, K., Kuang, J., Gao, A.: Real-time 3D arm motion tracking using the 6-axis IMU sensor of a smartwatch. In: 2021 IEEE 17th International Conference on Wearable and Implantable Body Sensor Networks (BSN), pp. 1–4. IEEE (2021)
29. Wu, H., Zhang, C., Zhang, W., Wang, J.: Monocular 3D human pose estimation by predicting the 2D pose and depth map simultaneously. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4500–4509 (2019)
30. Yu, Y., Chen, X., Cao, S., Zhang, X., Chen, X.: Exploration of Chinese sign language recognition using wearable sensors based on deep belief net. IEEE J. Biomed. Health Inform. **24**(5), 1310–1320 (2019)
31. Yuan, G., Liu, X., Yan, Q., Qiao, S., Wang, Z., Yuan, L.: Hand gesture recognition using deep feature fusion network based on wearable sensors. IEEE Sens. J. **21**(1), 539–547 (2020)
32. Zhang, D., et al.: Fine-grained and real-time gesture recognition by using IMU sensors. IEEE Trans. Mob. Comput. **22**(4), 2177–2189 (2023). https://doi.org/10.1109/TMC.2021.3120475

33. Zhang, R., et al.: EatingTrak: detecting fine-grained eating moments in the wild using a wrist-mounted IMU. Proc. ACM Human-Comput. Interact. **6**(MHCI), 1–22 (2022)
34. Zhang, X.: Application of human motion recognition utilizing deep learning and smart wearable device in sports. Int. J. Syst. Assur. Eng. Manage. **12**(4), 835–843 (2021)