Multimedia Tools and Applications (2022) 81:11845–11864
https://doi.org/10.1007/s11042-021-11888-8

1195: DEEP LEARNING FOR MULTIMEDIA SIGNAL PROCESSING AND APPLICATIONS

Action recognition and tracking via deep representation extraction and motion bases learning

Hao-Ting Li[1] · Yung-Pin Liu[1] · Yun-Kai Chang[1] · Chen-Kuo Chiang[1]

Received: 9 April 2021 / Revised: 7 November 2021 / Accepted: 23 December 2021 /
Published online: 21 January 2022

Abstract

Action recognition and positional tracking are critical issues in many applications in Virtual Reality (VR). In this paper, a novel feature representation method is proposed to recognize actions based on sensor signals. The feature extraction is achieved by jointly learning Convolutional Auto-Encoder (CAE) and the representation of motion bases via clustering, which is called the Sequence of Cluster Centroids (SoCC). Then, the learned features are used to train the action recognition classifier. We have collected new dataset of actions of limbs by sensor signals. In addition, a novel action tracking method is proposed for the VR environment. It extends the sensor signals from three Degrees of Freedom (DoF) of rotation to 6DoF of position plus rotation. Experimental results demonstrate that CAE-SoCC feature is effective for action recognition and accurate prediction of position displacement.

Keywords Action recognition · Motion bases · Action tracking · Deep learning

✉ Chen-Kuo Chiang
ckchiang@cs.ccu.edu.tw

Hao-Ting Li
lht107p@cs.ccu.edu.tw

Yung-Pin Liu
ping8tw@gmail.com

Yun-Kai Chang
tony091520933@gmail.com

[1] Department of Computer Science and Information Engineering, Advanced Institute of Manufacturingwith High-tech Innovations and Center for Innovative Research on Aging Society (CIRAS), National Chung Cheng University, No.168, Sec. 1, University Rd., Minhsiung, Chiayi 621301, Taiwan (R.O.C.)

1 Introduction

Virtual Reality (VR) has many applications, such as entertainment, art, education, and medical care. Most VR headsets have IMU sensors that can measure the translation and rotation of users. Mobile VR (MVR) is a special kind of VR. Unlike HTC Vive (Vive) or Oculus Rift, which require a computer and a set of VR equipment, users only need a smartphone and a head-mounted display (HMD) to use VR. Although MVR is not efficient as large VR, it is still popular due to its tiny, lightweight, and portable characteristics [24].

Action tracking is an essential issue for VR. It tracks the movement in real space by detecting the position of the HMD, controllers, or body parts. This allows the users to perform different actions or navigate the virtual environment. Action tracking in VR environment depends on the space locator. For example, Vive is equipped with two locators. However, MVR does not have a locator, like Vive or Oculus Rift. So, it needs other ways to track the positions.

Inertial tracking is one of the positional tracking methods where data is collected from accelerometers and gyroscopes. Accelerometers measure linear acceleration whereas gyroscopes measure angular velocity. Conventional inertial tracking uses a low-cost inertial measurement unit (IMU) based on Microelectromechanical systems (MEMS) to track the acceleration and orientation with high update rates. Generally, acceleration can be integrated to obtain the velocity and then it can be integrated again to obtain the displacement. However, the accuracy decreases as the error accumulation through time [5].

Action recognition plays an important role in VR applications as well. VR applications usually involve lots of action and accurate action recognition broadens the extent of applications. However, it is challenging to recognize actions with high similarity. Our goal is to use the sensor signals of wearable device for action recognition.

The first part of our paper aims for action recognition using sensor signals. To conquer the difficulty of recognizing similar actions, the motion bases method is proposed for discriminative feature representation. The motion bases is learned based on the feature learning from Convolutional Auto-Encoder (CAE) model in a clustering fashion. In addition, We propose a deep sequential model to predict the displacement of positions with acceleration and Euler angle to reduce error accumulation. In recent years, deep recurrent neural networks (RNN) have achieved remarkable success using sequential data. Body motions are intensely temporal dependent. It is appropriate to use the recurrent model to learn the temporal representations [8, 18, 32]. We adopt Long Short-Term Memory (LSTM) model to avoid vanishing gradient or exploding gradient problems during RNN training [11, 31]. It also has the ability to handle the long-term dependencies of sequential data [5, 11].

In addition to standard LSTM networks, we also use bidirectional LSTM (Bi-LSTM) networks, which are stacked two LSTM networks in forward and backward directions. Unlike standard LSTM networks, which can only consider the past information, Bi-LSTM networks can capture both past, and future information by two opposite temporal order in hidden layers [27, 33, 34]. Furthermore, except for the LSTM networks, we also use Temporal Convolutional Networks (TCN), capturing long-range patterns by temporal convolutions. Recent research shows that the convolutional architectures can achieve a good performance as recurrent models [1, 2, 7, 14].

The contribution of this paper two-fold. (1) A novel motion bases representation is proposed to extract discriminative features for action recognition. (2) Deep RNN models are exploited using sensor signals for action tracking in VR environment.

2 Related works

2.1 Human action recognition

The Human Action Recognition (HAR) task facilitates the computer to recognize human behavior and actions. It has become one of the essential tasks in the field of computer vision. Different HAR technology have been developed, such as sensor-based human action recognition [21], vision-based human action recognition [9]. As a typical deep learning model, Convolutional Neural Networks (CNN) [15] have been applied to many computer vision and image processing problems. The operation of 2D convolution is often used to extract features from two-dimensional data. Traditional 2D convolution operation usually captures locally spatial information; however, temporal information is also critical when processing the sequential data, such as videos. Therefore, Ji et al. [10] proposed a video-based deep learning method for action recognition. They added one dimension to the traditional 2D convolution operation and used the 3D convolution operation to capture the spatio-temporal information.

2.2 Action recognition in videos

In addition to the 3D convolution operation, Simonyan et al. [25] proposed a two-stream network structure for identifying actions by video-based data with temporal and spatial information. They used two independent CNN models, which process the spatial and temporal information of the video separately. Spatial information refers to the texture information within one single frame, such as objects, scenes. The network used for image classification can be used as a backbone model for spatial convolutional network, such as AlexNet [13], GoogLeNet [26]. Optical flows between frames are used as temporal information, which carry the energy of motions between frames. The outputs of two model which process spatial and temporal information are then fused to identify action types. Two schemes are usually adopted to fuse the outputs of two networks. One is averaging the outputs of softmax layers, and the other is to exploits the softmax scores as features and train another SVM classifier.

2.3 Activity recognition and early detection

Ma et al. [19] proposed a activity detection and early detection method to improve the accuracy of the temporal model based on the observation of of earlier activities. In their method, a new loss function is designed to train the Long Short-Term Memory (LSTM) model. The output of the detector is the detection score. As the score is continuous and monotonously non-decreasing function through time, the detector can output a higher detection score for the correct category while suppressing the scores of other categories.

2.4 Sensor-based human activity recognition

Li et al. [17] divide the problem of sensor-based human activity recognition into five steps to solve: data acquisition, data preprocessing, data segmentation, feature extraction, and classification. They compare seven feature extraction methods, including Hand Crafted (HC), Multi-Layer Perceptron (MLP), CNN, LSTM, Hybrid model featuring CNN and LSTM layers, Auto-encoder (AE), and CodeBook (CB) approach. In all, the hybrid architecture involving CNN and LSTM demonstrates significant improvement on the OPPORTUNITY dataset [4]. Yang et al. [30] propose CNN-based architecture for multi-sensor human activity recognition problems. The architecture follows the classic LeNet-like architecture that comprises four convolutional layers and two fully connected layers. The input multi-sensor signals are passed through the convolutional layers that operate along with the temporal dimension. The multi-sensor features are unified with the first fully connected layer and classified with the last fully connected layer. It is the early non-hand-crafted method for human activity recognition and gets a significant breakthrough. Qian et al. [22] propose DDNN architecture consisting of spatial, temporal, and statistical modules to extract different representations. The multi-sensor signals are partitioned by the sliding window method first and passed through the three modules simultaneously. The spatial module comprises LSTMs that input along with the sensors' dimensions, whereas the temporal module is along with the temporal dimension. Besides, the statistical module is composed of an auto-encoder to capture statistical features, where the training procedure is accomplished with MMD loss. These features are concatenated and fed into fully connected layers to fuse and classify activities. Wang et al. [29] propose a hierarchical LSTM, stacking two LSTM layers to extract different levels of features and a fully connected layer to classify activities. Nafea et al. [20] propose a two-stream architecture containing CNN layers and BiLSTM layers. The CNN layers comprise convolutional layers with various kernel-size to capture different resolutions. The BiLSTM layers are used temporal information for activity recognition and are considered forward and backward. The outputs of the two-stream layers will be fused with fully connected layers and then produce a classification result.

3 Proposed method

3.1 Action recognition

Our method can be divided into two parts: (1) feature extraction and (2) action recognition. In the first part, we have proposed two features, the Convolutional Auto-Encoder (CAE) feature and Sequence of Cluster Centroids (SoCC) feature. These features represent the action patterns and used for classifier of action recognition. The CAE feature is learned by the deep auto-encoder model, whereas the SoCC feature is represented by action bases learned from CAE features. Two kinds of features are concatenated as feature representation and are used to train XGBoost model to identify different actions. Figure 1 depicts the proposed system framework.

Convolutional Neural Network (CNN) is a very popular method in deep learning field for image analysis. LeCun [16] proposed this method which is used in many visual recognition tasks. The advantage of CNN is that its network structure exploits convolutional layers to capture the local dependency of the image or signals, other than entirely using fully connected layers. In additions, it has less number of parameters compared to that of deep neural network with fully connected layer only. CNN model training on a large amount of labeled data has high accuracy in recognition tasks.

CNN can be also used for feature extraction. The Auto-Encoder (AE) is a network architecture that performs feature extraction and feature representation in an unsupervised learning manner. The architecture is composed of the encoder and decoder part. In the encoder strurcture, the number of neurons layer by layer is decreased to compress the high-dimensional data into a low-dimensional latent vector. Based on the low-dimensional representation, the decoder increases the number of neurons layer by layer to reconstruct the input samples.

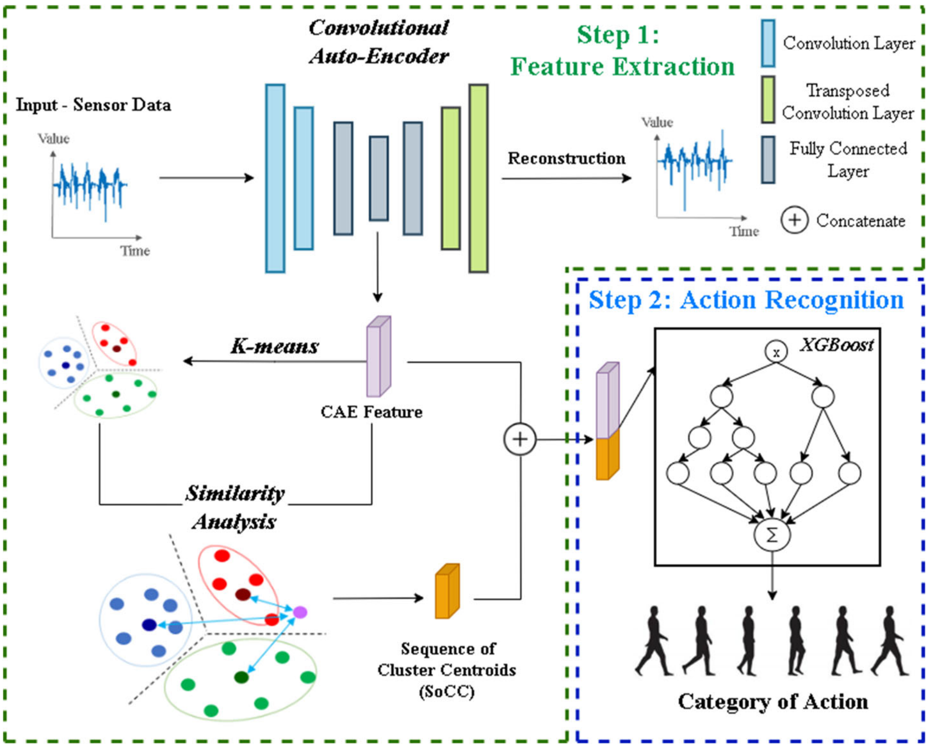

Fig. 1 The proposed system framework consists of two parts, feature extraction and action recognition. CAE feature and SoCC are learned from a deep auto-encoder model and a clustering model. These two features are then combined to train the action recognition classifier

3.1.1 Feature extraction

The advantages of CNN and AE models can be considered altogether for feature representation. CNN is good at capturing locally dependency of action signals, while AE exploits unlabeled data for training which is helpful when the number of training samples are limited. Therefore, we proposed to use the convolutional auto-encoder (CAE) as a feature extraction model for feature representation of action signals. As shown in Fig. 2, the CAE model is composed of an encoder and decoder. The encoder contains two 2D convolutional layers with a kernel of size 3×1 with the stride of 1. The decoder contains two 2D transposed convolutional layers with the same kernel size and stride. The bottleneck is three feed-forward fully-connected layers with neurons of 1024, 128, and 1024, respectively. The tanh function is used as the activation function between the convolution layer and the fully connected layer.

Each action takes different lengths of time interval, resulting in different data dimensions. To train CAE, the input size is requires to be fixed. We first take body acceleration, gravity acceleration, gyroscope, and quaternion to be the input data. The total data dimension is 26. The action signal is divided into segments of 25 timestamps by stride-1 sliding windows Fig. 3. Therefore, the input data from one sliding window can be fixed to the size 25 ×

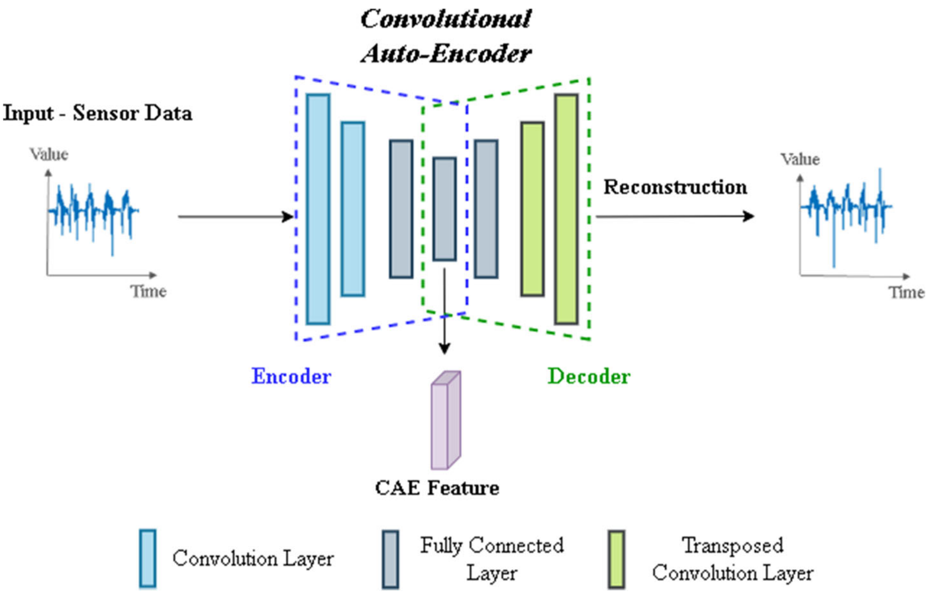

Fig. 2 CAE network structure

26. Then, the input data are used to train our CAE feature extraction model. During the collection of action signals, each action is performed and recorded separately. Therefore, the ground-truth labels can be assigned to the action signals once the action is finished. Then, the signals of each action with ground-truth labeling are used as input to CAE model for feature extraction by sliding windows.

The loss function to train the CAE model is defined as (1). N_t is the total number of timestamps in the input data. N_{attr} is the number of data dimension. $\hat{x}_{ij}$ is the output data, and x_{ij} is input data. The function is used to determine whether the output reconstructed by

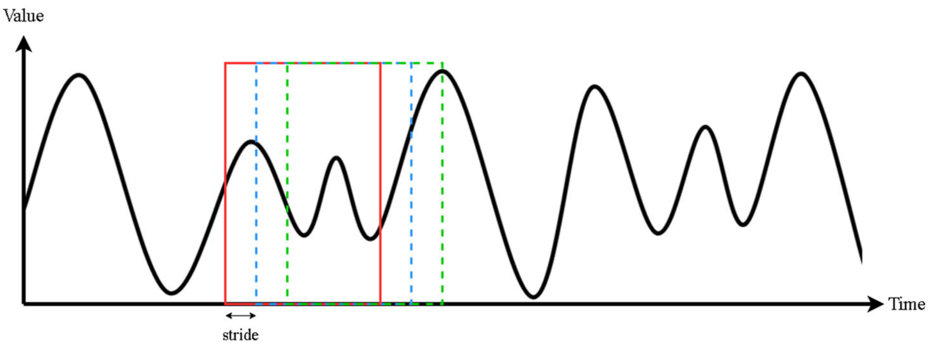

Fig. 3 The formation of input data to CAE model. The red box is the pre-defined time window. According to the number of strides, the time window moves to the blue box at the next step and then to the green box to segment the action signals into fixed-size input data to CAE model

the latent vector (CAE feature) is similar to the input. The closer the output to the input, the more representative features that the CAE model learns.

$$L_{reconst} = \frac{1}{N_t N_{attr}} \sum_{i=1}^{N_t} \sum_{j=1}^{N_{attr}} \left(\hat{x}_{ij} - x_{ij}\right)^2 \tag{1}$$

3.1.2 Feature representation of SoCC

In the action recognition problem, some actions are somewhat similar. Some different actions may have a common part. Take the actions of lower limbs of football for examples, *right foot raises to front* may be partially similar to *right foot kicks to front*. By the conventional metric learning, features learned for both actions could be similar. In view of this, we propose the idea of *deep motion bases* to distinguish actions when they are just slightly different or partially similar.

The motion signals are partitioned into segments via sliding windows. Each segment is fed to CAE to extract feature representation. By using the CAE model, the high-dimensional signal data can be represented by the latent vector in the low-dimensional space. The latent vectors in the latent space are firstly clustered by K-Means Clustering [12] to find cluster centroids. These vectors of cluster centroids are referred to as *motion bases* in our method. After obtaining these motion bases, the Euclidean distance between the CAE feature and each motion basis is calculated to analyze the similarity between the CAE feature and each centroid, as shown in Fig. 4. The lower value of the distance means the higher similarity to the centroid. The CAE features are sorted by the similarity to the sample data and choose the top-k to form the SoCC features; therefore, the centroids are numbered in sequence from

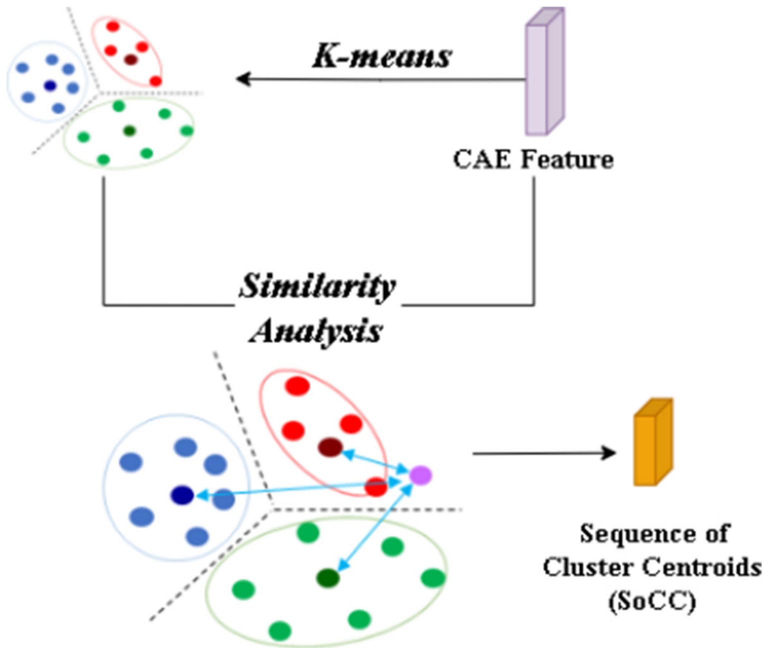

Fig. 4 The process to generate the SoCC feature

high similarity to low. This sequence is used as the feature representation *Sequence of Cluster Centroids* (*SoCC*) via motion bases learned from the action signals. Instead of using all clusters, only top-k clusters are chosen to form the SoCC features. This is because the distances of clusters in the bottom of the list after sorting are all very large. This hardly provides discriminative information in SoCC.

By using the feature representation SoCC, even though the first half of action 'right foot raises to front' may be partially similar to 'right foot kicks to front', the motion bases of the latter half of both actions will be different. This improves the discrimination of auto-encoder and k-means-based feature learning in an unsupervised manner.

3.1.3 Recognition models

In the second stage, we concatenate the CAE feature and SoCC to generate the CAE-SoCC feature for action recognition. We propose two different recognition methods. The first one is to use deep recurrent neural model trained by a novel loss function. The other is to exploit the popular machine learning method, XGBoost, which is widely used in Kaggle competitions.

LSTM (Long Short-Term Memory) is a deep network model that is used to process sequential data. It is different from the traditional RNN design. LSTM adds the gates to filter the past states, instead of accepting all past states, to process short-term time-series data effectively.

In the training of LSTM, we design the following loss function. The goal is to make the trained LSTM with higher probability to accurately recognize actions as more data samples are observed. In our method, LSTM outputs the detection scores corresponding to the total number of action categories and give the correct category a higher detection score than all other incorrect categories. The new function (2) is composed of classification loss (3) and ranking loss (4), p_t^y is the detection score of the ground-truth label y of the t-th time window. For classification loss, the detection score of the ground-truth class y of the t-th time window should be as large as possible. For ranking loss, we expect the detection score of the ground-truth class y of time t should be larger than the maximal detection score of the ground-truth class y of time 0 to t-1. In that case, as the timestamp increases, the detection score of the correct action class can be monotonically increasing through time. This helps determine the correct action class in the last sliding window.

$$L^t = L_c^t + \alpha L_r^t \tag{2}$$

$$L_c^t = -\log p_t^y \tag{3}$$

$$L_r^t = \max\left(0, \max_{t'=[0,t-1]} p_{t'}^y - p_t^y\right) \tag{4}$$

In the second model, XGBoost [6] is exploited. The XGBoost model uses an extreme gradient boosting algorithm to enhance the Gradient Boosted Decision Tree (GBDT) model. XGBoost is an optimized implementation for the Gradient Boosting method. It uses tree structure to fit the data and is optimized with Newton–Raphson method. XGBoost is implemented in parallel computing, and it provides hugely significant efficiency. Basically, XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. XGBoost and gradient boosting machines are both ensemble tree methods that apply the principle of boosting weak learners using the gradient descent architecture. However, XGBoost improves upon the gradient boosting framework through systems optimization and algorithmic enhancements. In addition, XGBoost minimizes a regularised (L1 and L2) objective function that incorporates a function of convex loss and a model complexity penalty term. In order to make the final prediction, the training continues iteratively, inserting new trees that predict the residuals or errors of previous trees that are then combined with previous trees. It is called gradient boosting since it uses a gradient descent algorithm to minimize the loss. Therefore, it is used broadly in Kaggle competitions.

3.2 Action tracking

For action tracking, we predict the displacement in each timestamp and then add it to its previous location, instead of predicting each position directly. To obtain displacement at each time t, we trained three different models, including LSTM, Bi-LSTM, and TCN, using acceleration and Euler angle information as input. Bidirectional LSTM (Bi-LSTM) networks are structured with two LSTM networks in forward and backward directions. Unlike standard LSTM networks, which only considers the past information, Bi-LSTM networks captures both past and future information by two opposite temporal order in hidden layers [27, 33, 34]. Except for the LSTM networks, Temporal Convolutional Networks (TCN) model is recently proposed and have shown good performance of capturing long-range patterns by temporal convolutions. Recent researches show that TCN achieves as good performance as recurrent models [1, 2, 7, 14]. We explore the capabilities of three models and compare the action tracking results in the next section.

4 Experimental results

4.1 Dataset

For action recognition and action tracking, we collected different datasets. The details of data collection are provided as follows.

Action Recognition Dataset. We use a self-collected dataset – Lower Limb Sensor Signals Dataset. There are 44 types of lower limb movements performed by 24 people at different time. One sensor, the MoTi wristband, is equipped at the ankle of the left and right feet, respectively. The collected sensor signals include acceleration, gyroscope, and quaternion. The details of action classes is provided in Table 1. Pre-processing by the low-frequency filter, the original acceleration data is divided into body acceleration and gravitational acceleration data [28]. The dataset is split into the training set and the test set. The training set comprises all the actions performed by 20 of the 24 people, and the remaining four people are used for the test set. Our model tries to classify the pattern of action based on the IMU quantity varies over time. Take foot raise and foot kick for examples, the changes of IMU quantity should have unique pattern for different actions.

Action Tracking Dataset. The IMU sensor is installed in the wristband while the user is holding the Vive controller to simultaneously collect data in synchronized time stamps, including the coordinates of Vive controller and the IMU signals. The IMU sensor and Vive controller are equipped by the same hand so that the coordinates and sensor data can be recorded simultaneously. All data are labeled with SI units. The IMU sensor is connected to a mobile APP through Bluetooth, whereas the Vive controller is connected to a computer under Unity environment. The mobile APP and Unity are triggered at the same time to record the accelerometers and gyroscopes from IMU and x-, y-, and z-coordinate of Vive controller in the VR environment. The displacement of x-, y-, and z-coordinate of VR world coordinate can be calculated and saved. Figure 5 illustrates

Table 1 Lower limb sensor signals dataset

Action	#-of samples	Action	#-of samples
right foot moves to front	200	right foot kicks to back forward	200
right foot moves to back	200	right foot kicks to back backward	200
left foot moves to front	200	left foot kicks to back forward	200
left foot moves to back	200	left foot kicks to back backward	200
right foot moves to left	200	right foot kicks to left forward	242
right foot moves to right	200	right foot kicks to left backward	242
left foot moves to left	200	left foot kicks to right forward	280
left foot moves to right	200	left foot kicks to right backward	280
right foot treads forward	200	right foot kicks to right forward	200
right foot treads backward	200	right foot kicks to right backward	200
left foot treads forward	201	left foot kicks to left forward	202
left foot treads backward	201	left foot kicks to left backward	202
right foot raises to front forward	200	feet jump and open	191
right foot raises to front backward	200	feet jump and close	200
left foot raises to front forward	200	rotate ankle along right tiptoe	211
left foot raises to front backward	200	rotate ankle along left tiptoe	200
right foot raises to right forward	197	rotate ankle along right heel	200
right foot raises to right backward	197	rotate ankle along left heel	200
left foot raises to left forward	200	draw a circle with right tiptoe	200
left foot raises to left backward	200	draw a circle with left tiptoe	200
right foot kicks to front forward	240	walk forward	20
right foot kicks to front backward	240	walk backward	21
left foot kicks to front forward	280	run forward	20
left foot kicks to front backward	280		

the process of data collection. We invited 8 subjects to collect the movement data. In the beginning, subjects need to hold the Vive controller and IMU in hands. The frame rate was set to 50 for both of Vive controller and IMU. During data collection, each subject starts to move freely in ten minutes. There are about 80 minutes of movement data in the end.

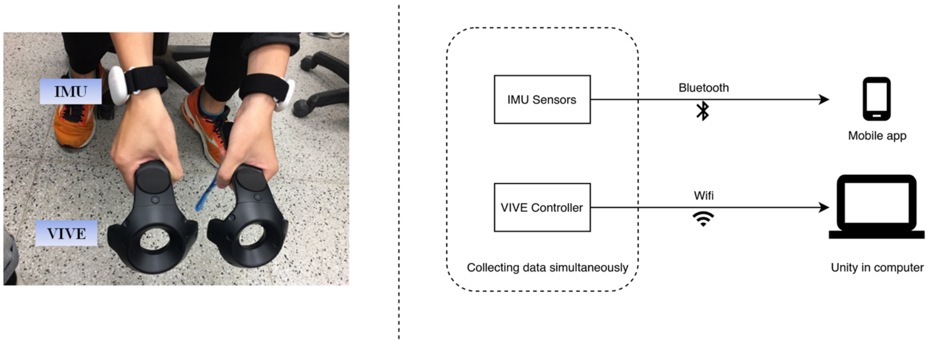

Fig. 5 Data collection with the HTC Vive and IMU sensors. The collection environment is built by Unity3D

The IMU sensors intrinsically comprise an accelerometer, a gravity sensor, and a gyroscope to measure accelerator, Euler angle, quaternion, and timestamp. We developed a mobile APP that connects to the IMU sensors through Bluetooth. The APP provides UI to label data and is able to start and stop receiving sensor signals. The accelerator consists of body accelerator and gravity. To extract the body accelerator, we use a low-pass filter with 0.25 Hz for passband frequency, 2 Hz for stopband frequency, 0.001 dB for passband ripple, 100 dB for stopband attenuation, and 32 Hz for sampling frequency.

The IMU signals contain the linear acceleration, the Eular angle, the quaternion, the velocity, and the corresponding labels. The frame rate is set to 50 Hz. The collected signals may delay at some time, resulting in the time gap of two samples decreasing to 0.01 seconds or increasing over 0.05 seconds. Thus, thresholds are set. If the signal with time gap to a previous sample fewer than 0.01 seconds is removed. The signal is interpolated if the time gap is greater than 0.05 seconds. To collect the coordinates of the Vive controller, we activate the Vive controller and the IMU sensor simultaneously when we start collecting the data. We convert the coordinates of the Vive controller into displacement for each timestamp.

4.2 Action recognition model

The accelerometer, gyroscope, and quaternion in the dataset are essential attributes, which are used to train the CAE model. Firstly, the stride-1 window is exploited to separate the signals of the whole action in the training dataset into partial time-series data in a sliding manner along the time axis (Fig. 3). This is used as the input data to train the CAE model. The CAE model comprises convolution layers, fully connected layers, and transposed-convolution layers. The encoder of the trained CAE model is used to extract features. Then, the CAE features are clustered by the K-means Clustering method to learn the motion bases. The Fig. 6 depicts the formation of SoCC feature.

The CAE feature and SoCC are concatenated into a new feature, CAE-SoCC. The dimension of CAE feature is 128 and that of SoCC is set to 20. Therefore, the dimension of CAE-SoCC feature is 148. In the action recognition model, the LSTM model is trained to predict the action class. The number of decision trees in the GBDT model is set to 1000. The maximum depth of the tree is set to 7, and the sum of the sample weights in the child node is 2.

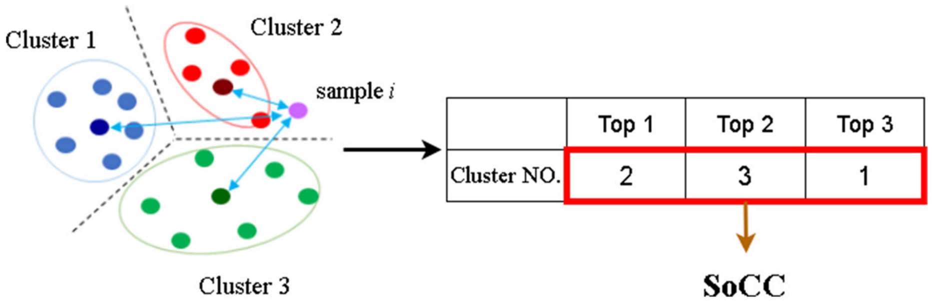

Fig. 6 The example of SoCC feature. Sample i is the CAE feature which is the latent vector of i-th sliding window data. The dark red, dark green, dark blue dots are cluster centroids. The sample i is closest to cluster 2 and far away to cluster 1. So, the SoCC of sample i is formed as [2, 3, 1]

4.3 Silhouette analysis

Silhouette analysis [23] is used to evaluate the effectiveness of clustering. It is measured based on the two factors: (1) the degree of cohesion within the cluster, and (2) the degree of separation between different clusters. Silhouette coefficient $s(\cdot)$ quantifies the performance of clustering. To compute the silhouette coefficient for a single sample i, the equation is defined as follows:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{5}$$

$$= \begin{cases} 1 - \frac{a(i)}{b(i)} & , a(i) < b(i) \\ 0 & , a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1 & , a(i) > b(i) \end{cases} \tag{6}$$

where $a(i)$ is the average distance between sample i and other samples in the same cluster, showing the degree of cohesion within the cluster. The smaller the value of $a(i)$, the higher the degree of cohesion. $b(i)$ is the average distance between sample i and all samples of other clusters, which represents the degree of separation between clusters. The higher the separation degree, the larger value the $b(i)$ is.

The value of the Silhouette coefficient is between -1 and 1. When silhouette coefficient is 0, it means that the sample is very close to the adjacent cluster; if it is 1, it means that the sample is far away from the adjacent cluster. When it is -1, the sample is assigned to the wrong cluster. When $a(i)$ ¿ $b(i)$, $b(i)/a(i)$ – 1 will be negative. If most samples have negative silhouette coefficients, the cluster is loose. By (5), Silhouette coefficient can be calculated for one sample. The Silhouette coefficient can be plotted one by one for each sample and for each cluster. The Silhouette coefficients for samples within the same cluster are depicted using the same color.

4.4 Clustering results

K-means clustering method is adopted in our experiments. To compare the clustering results under different cluster numbers, the number of clusters is set to 25, 30, 35, 40, and 44, respectively. The average silhouette coefficient is calculated for different experiments. It can be seen from Figs. 7, 8, 9, 10 and 11 and Table 2 that when the number of initial k is 30, the average silhouette coefficient shows the highest value 0.093432. Based on this result, the number of clusters is set to 30 for the rest experiments. The visualization of clustering using t-SNE is presented in Fig. 12

4.5 Comparison of feature learning

In this experiment, the CAE feature, SoCC, and CAE-SoCC features are compared using the same classifier. The SoCC feature is set as follows. Under the condition that the number of clusters is set to 30, top 20 cluster numbers closest to the sample data is used to form the SoCC feature. In other words, the dimension of SOCC feature is fixed to 20. XGBoost is trained using different features in this experiment. The experimental results in Table 3 show that the CAE-SoCC feature has a higher classification accuracy than SOCC and CAE features on Lower Limb Sensor Dataset.

Then, We choose two different classifiers in the experiments. LSTM is the prediction model for time series data. The ranking loss is included to train LSTM. The other classifier

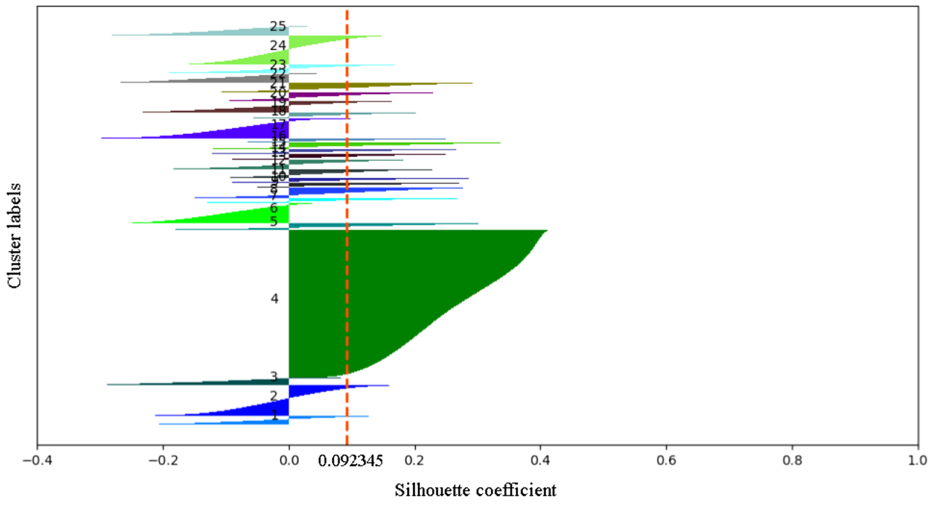

Fig. 7 The average silhouette coefficient for 25 clusters is 0.092345

is a non-sequential model, XGBoost. It is widely used in Kaggle competitions and shows great performance of classification and regression problems.

In this experiment, we use the CAE-SoCC feature to train different classifiers. Because one-time window data is regarded as one input data of CAE, a single action may have many outputs from the classifier. For XGBoost and LSTM trained without ranking loss, the voting mechanism is applied to fuse these predictions into one final prediction. The LSTM classifier trained with ranking loss takes consecutive window data as input. The prediction of the last time window is used as the final decision of action recognition. It can be observed from the Table 4 that the XGBoost classifier performs better than other classifiers.

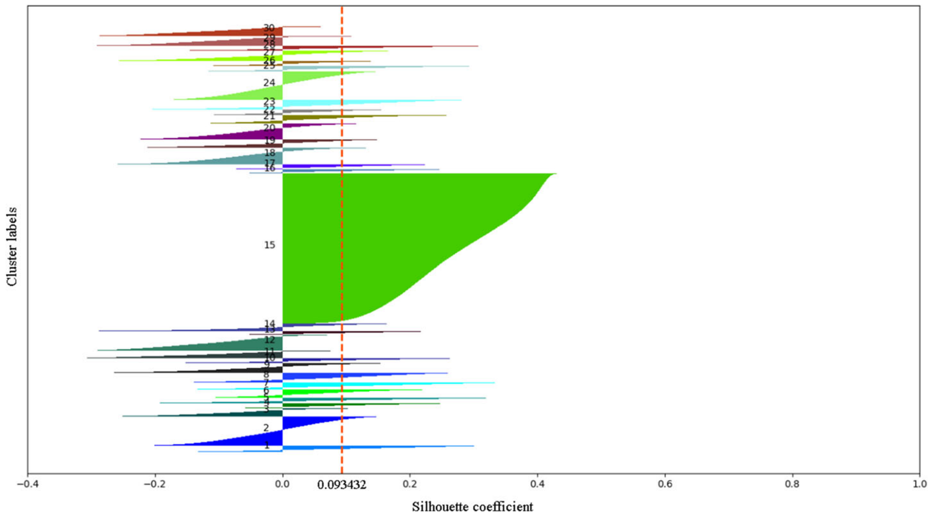

Fig. 8 The average silhouette coefficient for 30 clusters is 0.093432

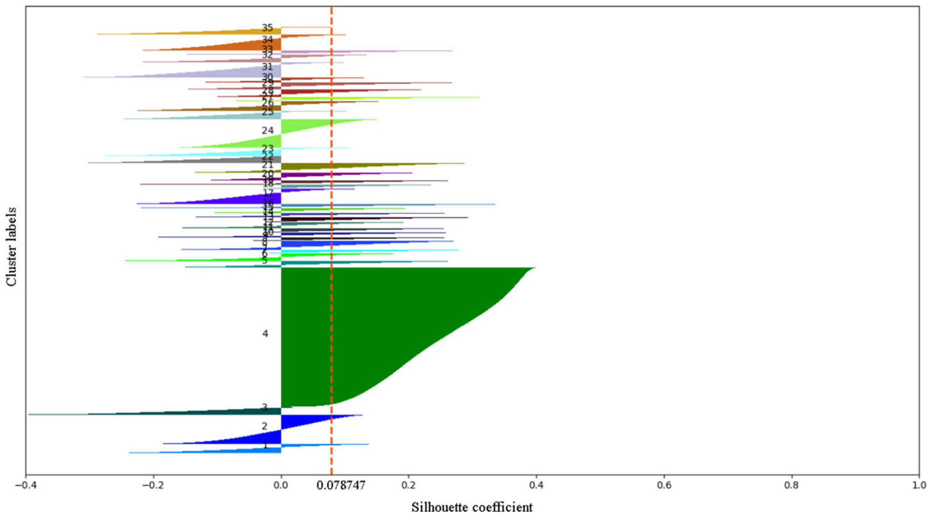

Fig. 9 The average silhouette coefficient for 35 clusters is 0.078747

Compared to other deep learning based methods methodologically, Yang et al. [30] proposed CNN-based architecture for multi-sensor human activity recognition problems, where the CNN is composed of four convolutional layers and two fully connected layers. Qian et al. [22] proposed DDNN architecture which comprised of spatial, temporal and statistical modules to extract different representations. Similarly, the statistical module contains an auto-encoder to extract statistical features, whereas we use the auto-encoder for unsupervised learning and clustering. Nafea et al. [20] proposed the two-stream architecture containing CNN layers and BiLSTM layers. Wang et al. [29] proposed a hierarchical LSTM, stacking two LSTM layers to extract different levels of features. These methods are simple

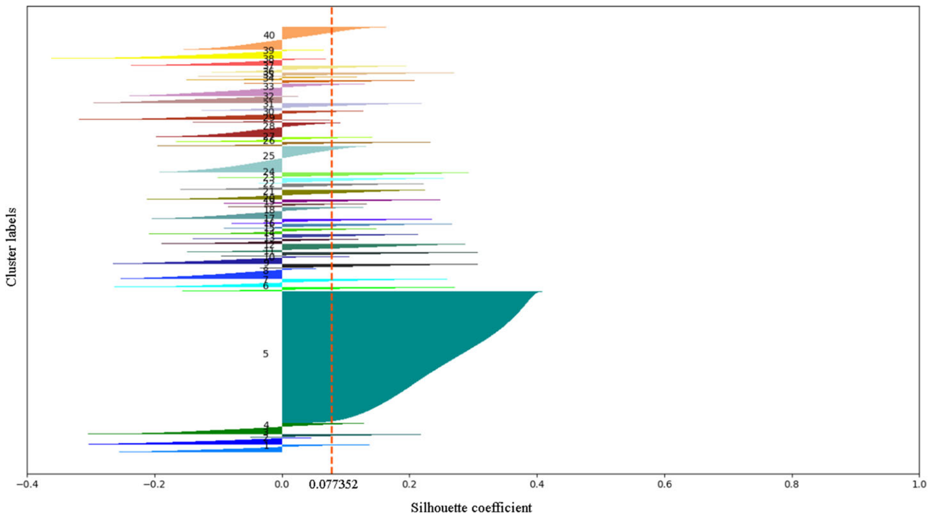

Fig. 10 The average silhouette coefficient for 40 clusters is 0.077352

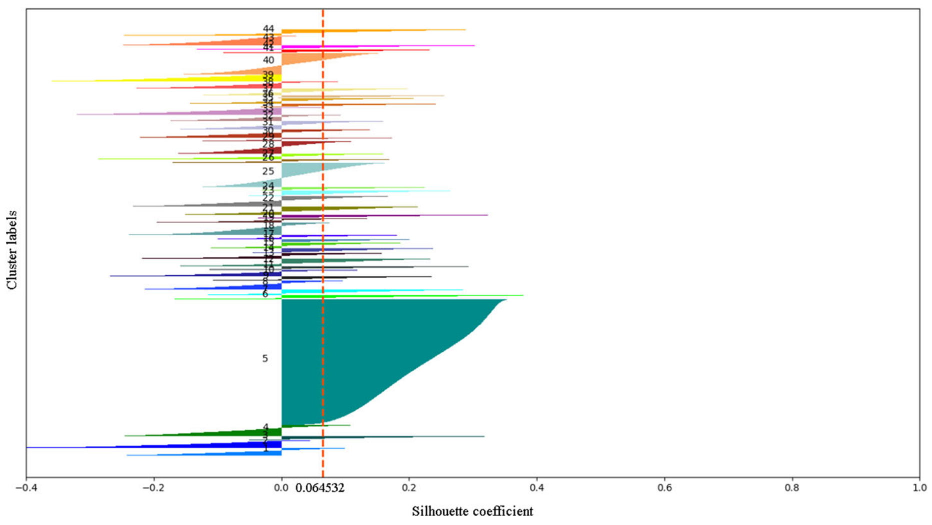

Fig. 11 The average silhouette coefficient for 44 clusters is 0.064532

and effective; however, most deep learning based models have the assumption that the action signals are required to be well-aligned. Since our method processes the signals by sliding window and encourages the detection scores to be monotonically increasing, the start time and end time can be more flexible in our method. Our method is potentially more suitable for continuous complex motion recognition.

4.6 Evaluation of action tracking

In this experiments, we used Mean Squared Error (MSE) to measure the performance of our method. The formula of MSE is defined as follows.

$$MSE = \sum_{i=1}^{n} (y_i - y_i^p)^2 \tag{7}$$

Different classifiers, LSTM, Bi-LSTM, and TCN, are adopted for comparison. Table 5 shows the comparison results of MSE loss. The LSTM model performs best among all three models. The possible reason could be the short-term correlation of human actions. Bi-LSTM and TCN have more complicated model structure and are suitable for long-term actions. However, as the movement keep going, the accumulated positional errors may decrease the tracking performance dramatically by these methods.

Table 2 The average silhouette coefficient over the number of clusters

K	Average silhouette coefficient
25	0.092345
30	0.093432
35	0.078747
40	0.077352
44	0.064532

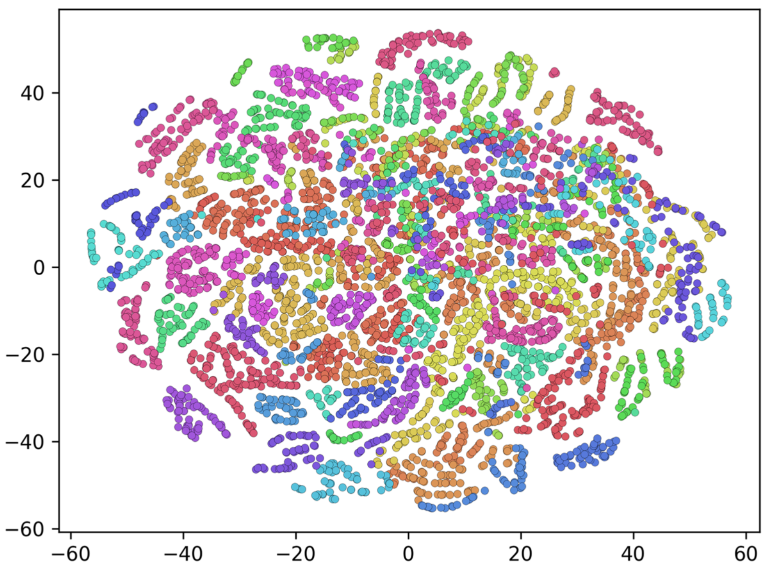

Fig. 12 The visualization result using t-SNE

Table 3 The classification results of features at different stages

Feature type	Accuracy
SoCC	44.52
CAE feature	70.32
CAE-SoCC feature	76.96

Table 4 The classification results of different classifiers using CAE-SoCC feature

Classifier type	Accuracy
LSTM without ranking loss	57.26
LSTM with ranking loss	57.83
XGBoost	76.96

Table 5 MSE results in the test set for different models

Model name	MSE Loss
LSTM	0.000033
Bi-LSTM	0.000045
TCN	0.000052

Table 6 MSE results in the test set for the different number of LSTM layers

#cells	MSE Loss
1	0.0000400
2	0.0000240
3	0.0000103
4	0.0000240

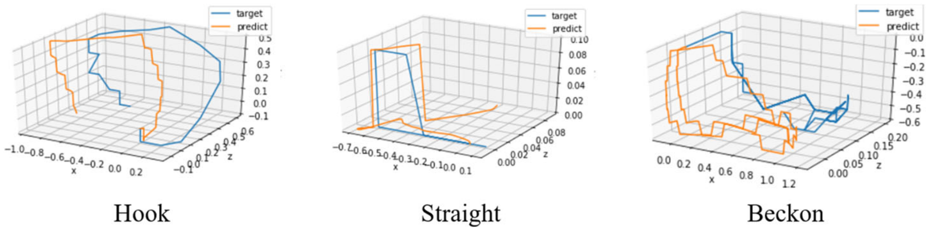

Fig. 13 Prediction result by LSTM. The picture shows the prediction result of hook punch, straight punch, and beckon. The blue line is the target, and the orange line is the prediction result

According to the result above, we further investigate different cell number of LSTM to compare the performance. Table 6 presents the results of MSE loss with different cells in LSTM. It shows three cells of LSTM gives the lowest MSE. Figure 13 expresses the 3D visualization motion tracking for actions, such as hook punch, straight punch, and beckon. We can see that the motion tracking predicted by the model (the orange line) is very similar to the target (the blue line).

4.7 Comparison to acceleration integration

In addition to use deep model to predict displacement, a conventional method [3] is to integrate the acceleration to obtain the velocity by (8).

$$v_t = v_{t-1} + a_t \cdot t \tag{8}$$

where v_t is the velocity at time t, a_t is the acceleration at time t, and t is the time interval between two samples. The integration result of velocity is in Fig. 14. We can see the velocity of x-axis has increasing deviations as time changes. The velocity of y-axis and z-axis also have slight deviations, respectively. Next, we used (9) to integrate the velocity over time to calculate the displacement.

$$x_t = v_{t-1} \cdot t + \frac{1}{2} \cdot a_t \cdot t^2 \tag{9}$$

where x_t is the displacement at time t. The displacement of the target ground-truth, prediction of deep model, and integration results are compared in Fig. 15. The green line is the integration result. The blue and orange lines are the target ground-truth and prediction results from deep model, respectively. We can see that the integration results are much

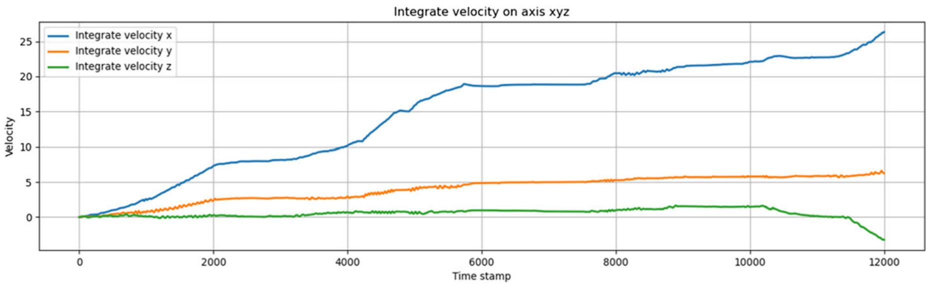

Fig. 14 x, y, z axis of the velocity integrated by the acceleration. The blue line is the x-axis; the orange line is the y-axis; the green line is the z-axis

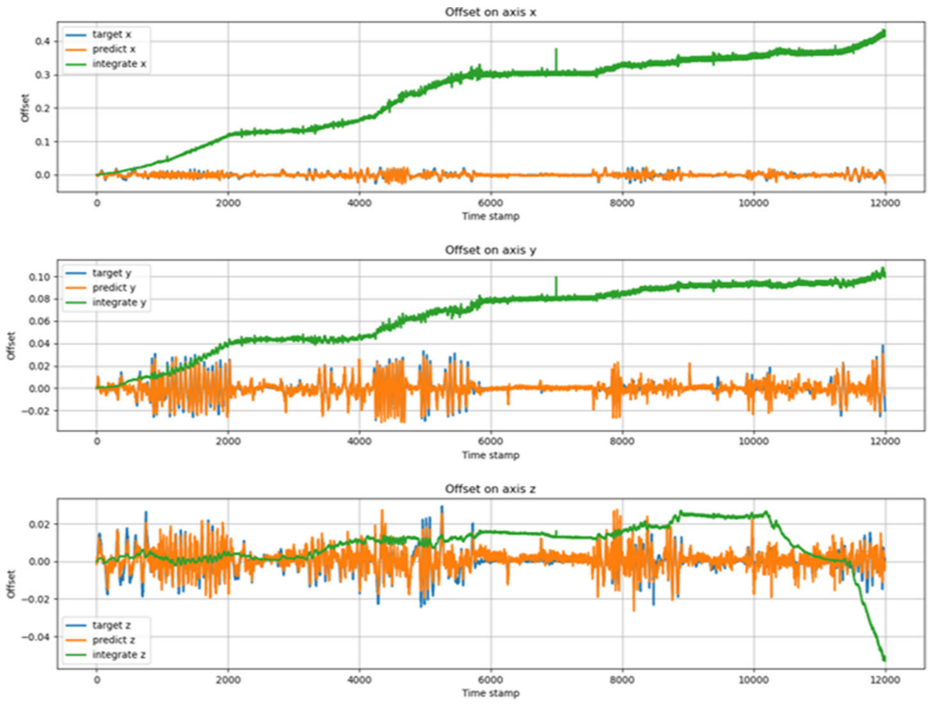

Fig. 15 x, y, z axis of the target, prediction and integration displacement. Green line is integration result, as for blue and orange line are the target and prediction result respectively

worse than the prediction results from deep model. When applying integral method, slightly small errors could be accumulated and cause significant deviations in displacement when integration is applied.

5 Conclusion

We propose a novel method based on the representation of motion bases for action recognition and an evaluation of deep models for action tracking. In the action recognition, features are learned from the CAE model. To further improve the CAE feature, the idea of feature learning based on motion bases is proposed. Clustering method is applied and the centroids of clusters are exploited as motion bases. The cluster number of top-k closest bases are extracted as Sequence of Cluster Centroids feature. The experimental results show that the CAE-SoCC feature is suitable for XGBoost classifier and demonstrate high accuracy for action recognition.

In action tracking, our method learns the relationship between sensor signals and displacement by deep learning model. It handles the accumulative errors caused by low-cost IMU during tracking in MVR, where a space locator is not required at all. In addition, we also compare the results of calculating the displacement by integrating acceleration. The results show that the accumulative errors would be increased over time, and the result predicted by deep model is much better than the traditional method. In our work, the action recognition framework is a two-stage framework consisting of clustering by clustering and classification by XGBoost and hence is challenging to integrate. Caron et al. [3] propose DeepCluster that produces pseudo-labels from clustering procedures as ground-truth for classification. This approach allows the clustering to train in an end-to-end fashion with deep neural networks and can remove the stages in the framework.

Acknowledgements This work was supported by the Advanced Institute of Manufacturing with High-tech Innovations and Center for Innovative Research on Aging Society (CIRAS) from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by Ministry of Education (MOE) in Taiwan. It was also supported by Ministry of Science and Technology under the grant 109-2218-E-194-009.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Aparecido Garcia F, Mazzoni Ranieri C, Aparecida Francelin Romero R (2019) Temporal approaches for human activity recognition using inertial sensors. In: 2019 Latin american robotics symposium (LARS), 2019 brazilian symposium on robotics (SBR) and 2019 workshop on robotics in education (WRE), pp. 121–125
2. Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271
3. Caron M, Bojanowski P, Joulin A, Douze M (2018) Deep clustering for unsupervised learning of visual features. In: Proceedings of the European conference on computer vision (ECCV), pp 132–149
4. Chavarriaga R, Sagha H, Calatroni A, Digumarti ST, Tröster G, Millán JdR, Roggen D (2013) The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. Pattern Recogn Lett 34(15):2033–2042
5. Chen C, Lu X, Markham A, Trigoni A (2018) Ionet: Learning to cure the curse of drift in inertial odometry. arXiv:abs/1802.02209
6. Chen T, Guestrin C (2016) Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp 785–794
7. Dauphin YN, Fan A, Auli M, Grangier D (2017) Language modeling with gated convolutional networks. In: International conference on machine learning, PMLR, pp 933–941
8. Jaouedi N, Boujnah N, Bouhlel MS (2020) A new hybrid deep learning model for human action recognition. Journal of King Saud University - Computer and Information Sciences 32(4):447–453. https://doi.org/10.1016/j.jksuci.2019.09.004, http://www.sciencedirect.com/science/article/pii/S1319157819300412. Emerging Software Systems
9. Jegham I, Khalifa AB, Alouani I, Mahjoub MA (2020) Vision-based human action recognition: an overview and real world challenges. Forensic Science International: Digital Investigation 32:200901
10. Ji S, Xu W, Yang M, Yu K (2012) 3d convolutional neural networks for human action recognition. IEEE Trans Pattern Anal Mach Intell 35(1):221–231
11. Jozefowicz R, Zaremba W, Sutskever I (2015) An empirical exploration of recurrent network architectures. In: International conference on machine learning, PMLR, pp 2342–2350
12. Kapoor A, Singhal A (2017) A comparative study of k-means, k-means++ and fuzzy c-means clustering algorithms. In: 2017 3Rd international conference on computational intelligence & communication technology (CICT), IEEE, pp 1–6
13. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inform Process Syst 25:1097–1105
14. Lea C, Flynn MD, Vidal R, Reiter A, Hager GD (2017) Temporal convolutional networks for action segmentation and detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
15. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1(4):541–551
16. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324

17. Li F, Shirahama K, Nisar MA, Köping L, Grzegorzek M (2018) Comparison of feature learning methods for human activity recognition using wearable sensors. Sensors 18(2):679
18. Liu J, Wang G, Hu P, Duan LY, Kot AC (2017) Global context-aware attention lstm networks for 3d action recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
19. Ma S, Sigal L, Sclaroff S (2016) Learning activity progression in lstms for activity detection and early detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1942–1950
20. Nafea O, Abdul W, Muhammad G, Alsulaiman M (2021) Sensor-based human activity recognition with spatio-temporal deep learning. Sensors 21(6):2141
21. Poppe R (2010) A survey on vision-based human action recognition. Image and Vision Computing 28(6):976–990
22. Qian H, Pan SJ, Da B, Miao C (2019) A novel distribution-embedded neural network for sensor-based activity recognition
23. Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20:53–65
24. Saker M, Frith J (2018) From hybrid space to dislocated space: Mobile virtual reality and a third stage of mobile media theory. New Media —& Society 21:146144481879240. https://doi.org/10.1177/1461444818792407
25. Simonyan K, Zisserman A (2014) Two-stream convolutional networks for action recognition in videos. arXiv:1406.2199
26. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
27. Ullah A, Ahmad J, Muhammad K, Sajjad M, Baik SW (2018) Action recognition in video sequences using deep bi-directional lstm with cnn features. IEEE Access 6:1155–1166
28. Van Hees VT, Gorzelniak L, Leon ECD, Eder M, Pias M, Taherian S, Ekelund U, Renström F, Franks PW, Horsch A et al (2013) Separating movement and gravity components in an acceleration signal and implications for the assessment of human daily physical activity. PloS One 8(4):e61691
29. Wang L, Liu R (2020) Human activity recognition based on wearable sensor using hierarchical deep lstm networks. Circuits, Systems, and Signal Processing 39(2):837–856
30. Yang J, Nguyen MN, San PP, Li XL, Krishnaswamy S (2015) Deep convolutional neural networks on multichannel time series for human activity recognition. In: Twenty-fourth international joint conference on artificial intelligence
31. Yong D, Wang W, Wang L (2015) Hierarchical recurrent neural network for skeleton based action recognition. In: 2015 IEEE Conference on computer vision and pattern recognition (CVPR), pp 1110–1118
32. Zhang P, Lan C, Xing J, Zeng W, Xue J, Zheng N (2017) View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In: Proceedings of the IEEE international conference on computer vision (ICCV)
33. Zhou P, Shi W, Tian J, Qi Z, Li B, Hao H, Xu B (2016) Attention-based bidirectional long short-term memory networks for relation classification. In: Proceedings of the 54th annual meeting of the association for computational linguistics (Volume 2: Short Papers). Association for Computational Linguistics, Berlin, pp 207–212. https://doi.org/10.18653/v1/P16-2034, https://www.aclweb.org/anthology/P16-2034
34. Zhu A, Wu Q, Cui R, Wang T, Hang W, Hua G, Snoussi H (2020) Exploring a rich spatial–temporal dependent relational model for skeleton-based action recognition by bidirectional lstm-cnn. Neurocomputing 414:90–100. https://doi.org/10.1016/j.neucom.2020.07.068, http://www.sciencedirect.com/science/article/pii/S0925231220311760