International Journal for Housing Science and Its Applications
Publish August 10, 2025. Volume 46, Issue 4 Pages 7590-7604

https://doi.org/10.70517/ijhsa464654

Human Skeletal Joint Pose-Action Bimodal Recognition Model Based on Graph Neural Networks

Ge Zhang[1,*]
[1] Network Information Service Center, Henan University of Economics and Law, Zhengzhou, Henan, 450000, China
Corresponding authors: (e-mail: zzge0212@163.com).

Abstract Human action recognition technology plays a crucial role in the field of computer vision, where it is constantly advancing and being widely adopted. Among the various techniques, graph neural networks are currently the mainstream method for processing unstructured skeleton sequences. However, research on action recognition based on skeleton data still faces several key challenges. This paper establishes a graph convolutional neural network (GCN) model based on the theoretical framework of GCN and key pose estimation/reduction for human skeletons. It extracts features from human skeleton data and introduces a 3D concept, approaching the task from both temporal and spatial dimensions to perform action recognition on the extracted features. The model was tested on the NTU-RGBD dataset under the CS and CV standards. The recognition accuracy rates of the GCN and 3D-GCN models under the CS standard were 75.853% and 78.251%, respectively. Under the CV standard, the recognition accuracy rates were 82.294% and 86.381%, respectively. The 3D-GCN model proposed in this paper achieved a higher recognition accuracy rate. The 3D-GCN model achieved an accuracy rate of 91.941% for recognizing four actions: falling, running, kicking, and squatting, demonstrating good performance in human action recognition.

Index Terms Graph Convolutional Neural Network, Human Skeleton Key Poses, 3D-GCN Model, Human Action Recognition

I. Introduction

Human motion recognition is one of the key research directions in the field of computer vision. Computers extract features from video or image data to analyze human behavioral patterns. The development of this technology holds significant implications for education, medical rehabilitation, accident prevention, and other fields [1]-[3]. Among these, human pose recognition and estimation involve designing network models to process input images and extract key points representing human poses within the images. It can be observed that pose recognition and estimation constitute a regression task [4]. Action recognition involves designing corresponding networks based on human posture to extract skeletal semantic features and determine the corresponding action categories, which is a classification task [5]. Human action recognition is a multidisciplinary field, with the research focus on accurately extracting human action features from continuous image or video sequences and converting them into quantifiable labels [6]. Traditional action recognition methods often rely on manually designed feature extractors, such as optical flow methods and scale-invariant feature transform (SIFT), which have certain limitations when processing complex scenes [7], [8]. The main issue is that the modeling of human bodies is still largely based on image-like approaches, resulting in high computational overhead and performance bottlenecks. Designing a method that can automatically learn the intrinsic representation of data has become a key focus of current research.

Current human action recognition algorithms struggle to achieve efficient and accurate recognition results in complex scenes, so human action recognition still holds significant research and commercial value. However, with the rapid development of intelligent technologies, new opportunities have emerged for complex scene recognition [9]-[11]. With the increasing availability of skeleton data, the role of skeleton-based motion recognition has become increasingly prominent. It can utilize skeleton data rather than video data, thereby significantly improving recognition speed with lower data sizes [12]-[14]. Additionally, skeleton pose data avoids interference from environmental noise or background in raw image data, which helps models effectively improve recognition accuracy [15], [16].

In recent years, researchers have creatively proposed novel deep learning methods, achieving a significant leap in model performance [17]. With the rapid development of deep learning technology, graph neural networks, as a powerful node representation learning method, have demonstrated significant advantages in various fields [18]. In the complex and challenging task of human action recognition, graph neural networks have also demonstrated

great potential. Graph neural networks effectively process unstructured data by mapping nodes in a graph to a high-dimensional space and utilizing graph convolution operations to capture interactions between nodes [19], [20]. In human action recognition, GNNs can naturally represent human actions as nodes in a graph, with the execution order and key points of the action serving as edge weights. They can learn the interrelationships between different body parts and the overall flow of the action, thereby achieving high-precision recognition of human actions [21], [22].

Early human action recognition methods primarily relied on traditional computer vision techniques and pattern recognition methods, requiring manual extraction of behavioral features that represent changes in human movement, such as silhouette outlines, motion trajectories, and spatiotemporal points of interest, followed by the selection of appropriate classification algorithms for recognition [23]-[25]. Early human motion recognition methods lacked flexibility and accuracy in complex scenes, and manual feature extraction was often designed for specific scenes and task requirements, resulting in certain subjectivity and limitations [26]. In recent years, the emergence of deep learning technology has provided powerful tools and methods for human motion recognition. Compared to traditional methods, deep learning-based methods can automatically learn higher-level features and more accurately recognize various human motions.

Reference [27] constructed a two-layer temporal convolutional neural network with three-dimensional convolutional layers and convolutional long short-term memory layers, using spatio-temporal features and activity locations as analysis parameters for human activity recognition, improving recognition accuracy while maintaining non-invasiveness. Reference [28] integrated radar data with a stacked recurrent neural network incorporating long short-term memory units, proposing a human motion recognition technique capable of classifying six different types of human motions. Literature [29] compared the accuracy of various techniques in human sign language recognition based on deep sensor data, with the DeepConv LSTM model showing the best performance, followed by convolutional neural networks. Embedding convolutional layers into deep learning models may yield better recognition results. Literature [30] integrates decision trees, linear regression methods, noise removal mechanisms, and recurrent neural network models to design a human action recognition method based on WiFi channel state information, which demonstrates better recognition performance compared to basic methods. Literature [31] designs an end-to-end dual-stream attention recurrent neural network, focusing on the effective features of the initial input graph and the deep feature output graph, and combines deep feature-related layers to optimize network parameters, achieving better human action recognition performance. Literature [32] utilizes attention mechanism-embedded capsules and gated recurrent units to construct a spatio-temporal multi-feature extraction framework, combined with a threshold-based aggressive activity detection method, to extract human action features in prison settings for human action recognition. Literature [33] employs deep reinforcement learning for multi-feature fusion in multi-feature scenarios, combined with an attention model, to form a multi-feature fusion human behavior recognition algorithm, achieving a 98% accuracy rate.

Reference [34] focuses on multiple gait patterns of exoskeletons, employing a distance-oriented feature selection method to extract optimal features from multi-modal information of lower limb movements. It introduces a multi-layer backpropagation neural network to construct a linear mapping model between feature quantities and motion states, achieving an accuracy of 92.7%–97.4%. Literature [35] applied a spatial transformation framework and temporal convolutional networks to develop a skeleton gait recognition model, enhancing its robustness and accuracy, with an accuracy rate of 90%. Literature [36] used data captured by depth sensors and daily human skeleton activity data as input, integrating image processing and deep learning to develop an intelligent human motion recognition system that is not affected by environmental or domain constraints. Reference [37] proposed a convolutional recurrent neural network and long short-term memory network as the generator and discriminator for a generative adversarial network, where the generator is responsible for reconstructing occluded skeletal parts, and the discriminator evaluates the reconstructed parts, thereby improving the recognition performance of human skeletal joints under occlusion conditions. Reference [38] applies an end-to-end fast graph neural network to human action recognition, achieving nearly 100% recognition accuracy due to its efficient extraction of single-sample information and the form of an undirected graph structure.

This paper introduces a typical graph convolutional network based on spatial domain, GraphSage, and conducts research on graph convolutional neural networks based on this model. Two approaches similar to Martinez's fully connected model and fully convolutional model are proposed to regress the Euclidean distance matrix, establishing a human skeleton joint pose estimation regression model. The established model is used to extract human skeleton data features, and the skeleton data is stacked frame by frame to ensure that the resulting skeleton spatiotemporal map contains both temporal and spatial information. A 3DCNN action recognition framework is constructed, and the model's training and learning strategy is adjusted using the SGDR gradient descent method. Two datasets, NTU-RGB+D and Kinetics, are constructed to validate the action recognition performance of the two

datasets. Through ablation experiments, the recognition performance of the three information streams used in this paper is evaluated.

II. Related technical theories of graph neural networks and action recognition

II. A. Graph Neural Networks

II. A. 1) Graph Convolutional Networks

Neural networks have achieved tremendous success over the past decade or so, but early neural networks could only be applied to conventional Euclidean data. However, a portion of real-world data is based on non-Euclidean data structures. Common examples of Euclidean data include images and audio. One characteristic of Euclidean data is its regular arrangement. For images, they are composed of a series of pixels, and the arrangement of pixels follows a fixed pattern. Except for pixels at the image edges, each pixel has eight adjacent nodes. Therefore, for example, convolution operations can use a globally shared fixed-size convolution kernel to extract features from images. However, for non-Euclidean data, the number of adjacent nodes for different nodes may vary, so it is not possible to use a globally shared convolution kernel to extract features. Figure 1 compares the neighboring nodes of Euclidean and non-Euclidean data. For example, a social network forms a graph model, where social participants are the nodes in the graph, and the edges of the graph represent the social relationships between people. Analyzing the social network graph can enable user profiling. To effectively analyze and process graph data, graph neural networks have emerged. Models that apply deep learning methods to graph data are collectively referred to as graph neural networks. Graph neural networks include graph convolutional networks, graph generative networks, etc. [39]. This section only introduces the graph convolutional networks relevant to this paper.

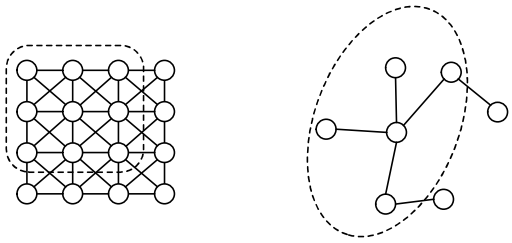


Figure 1: Comparison of neighbor nodes with Euclidean data and non-Euclidean data

Spatial-based graph convolutional networks define graph convolution operations based on spatial relationships. An image can be viewed as a special type of non-Euclidean graph, where each pixel represents a node in the graph. As shown in Figure 1, each pixel is directly connected to eight adjacent pixels, and the order of these eight pixels is fixed. Traditional convolution uses a fixed-size convolution kernel to perform a weighted average on all values within the convolution kernel's range. Due to the ordered nature of neighboring pixels, the convolution kernel shares its learned weights as it slides across the image. Analogous to traditional convolution, graph convolution theoretically also uses the convolution kernel to fuse the values of the central node and its neighboring nodes to update the central node's feature values. The difference lies in the fact that the number of neighboring nodes for different central nodes in a graph is not fixed. In summary, to eliminate the impact of the varying number of neighboring nodes and ensure weight sharing, GraphSage is a typical graph convolutional network based on spatial domain.

For a general graph, we can use $G=(V,E)$ to represent it. Here, V denotes the set of all nodes in the graph, $V=\{v_1,v_2,\cdots,v_N\}$ where N is the number of nodes in the graph, and E denotes the set of all edges in the graph. For a graph G, there are an adjacency matrix A, a degree matrix D, and an eigenmatrix X.

The adjacency matrix A represents the connection relationships between the nodes in the graph, $A\in R^{N\times N}$:

$$A_{ij}=\begin{cases}1 & v_iv_j\in E\\ 0 & Other\end{cases} \tag{1}$$

The degree matrix D is a diagonal matrix, $D\in R^{N\times N}$, and the diagonal elements are the degrees of each node, representing the number of edges connected to that node:

$$D_{ii}=\sum_j A_{ij} \tag{2}$$

The feature matrix X represents the feature information of all nodes in the graph, $X \in R^{N \times F}$, where F is the feature dimension. Figure 2 shows traditional convolution and graph convolution.

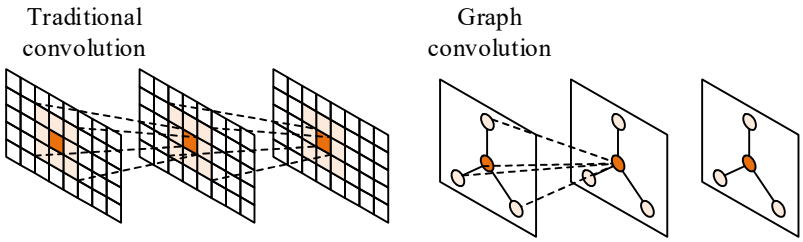


Figure 2: Traditional convolution and graph convolution

For convolution operations on Euclidean data, the essence is to perform a weighted sum of all node features within the neighborhood of each node to obtain the feature values of the next layer nodes. Analogously, for non-Euclidean data, when the convolution kernel is set to 1 for the neighborhood, its state update function is:

$$H^{k+1} = f\left(H^k, A\right) = \sigma\left(\hat{D}^{-0.5}\hat{A}\hat{D}^{-0.5}H^k W^k\right) \tag{3}$$

Among them, H^k is the feature of the k th layer, W^k is the weight learned by the k th layer, and $\sigma(\cdot)$ is the activation function. $A = A + I_N$, $D_u = \sum_j A_{yj}$, and I_N is the N th identity matrix.

Since using the adjacency matrix directly would cause nodes to lose their original node data, self-connections need to be added to each node, and the adjacency matrix is directly added to the identity matrix. Additionally, to avoid large feature values due to a single node having a large degree, normalization is performed using the degree matrix with added self-connections.

II. A. 2) **Image Feature Extraction Network**

Early convolutional neural networks often improved their learning capabilities by increasing the depth of the network. However, increasing the number of layers in the network leads to an increase in the number of parameters, reducing computational efficiency; simultaneously, the increase in parameters also makes the model prone to overfitting. For image classification tasks, the size of the main subject in an image, which significantly influences classification, can vary greatly, and the position of the main subject within the image is not fixed. Using convolutional kernels of different sizes can also yield different results in image recognition. Some researchers proposed the Inception module in GoogLeNet, which introduces convolutional kernels of different sizes in parallel within the same layer to obtain receptive fields of different scales without increasing the network depth. The outputs from each branch are then concatenated into a feature map with more channels [40].

*II. B. **Regression algorithm for estimating human skeletal joint posture***

Since the academic community has already developed mature research on 2D pose estimation algorithms, many 3D pose estimation algorithms use 2D poses to regressively estimate 3D poses. Algorithms based on this approach outperform those that directly estimate 3D poses [41]. This section will introduce 3D pose estimation algorithms that adopt the 2D regression approach.

In 2017, some scholars proposed the nearest neighbor matching method for 3D pose estimation. The method matches the closest pose in a pre-constructed 3D pose library. By calculating the error between the projection in the 3D pose library and the 2D pose, the probability $P(X \mid x)$ that the projection belongs to the 3D pose is estimated:

$$P\left(X = X_i \mid x\right) \propto e^{-\frac{1}{\sigma^2}\left\|M_i(x) - x\right\|_2} \tag{4}$$

Among them, $\{X_i \mid i = 1, 2, \cdots, n\}$ is a 3D pose library, x is a 2D pose, and M_i is the projection transformation matrix of this 3D pose. The advantage of this method is that it can perform 3D pose regression even if there are a few undetected 2D keypoints. However, it relies on a large amount of 3D prior pose information, and it is difficult to construct a complete 3D pose library.

The regression problem is abstracted as finding the optimal mapping function from 2D pose to 3D pose:

$$f^* = \min_f \frac{1}{N}\sum_{i=1}^{N} L\left(f(x_i) - y_i\right) \tag{5}$$

A simple and effective fully connected residual regression module was proposed for 3D pose estimation. At the time, its prediction performance surpassed most end-to-end 3D pose estimation algorithms, also demonstrating the reliability of 2D pose regression for 3D pose estimation. The entire algorithm model achieves 3D pose prediction using only linear layers, batch normalization layers, activation function layers, and Dropout layers, thereby reducing the complexity of 3D pose prediction.

Some scholars have also proposed the idea of Euclidean distance matrix regression rather than directly using pose for regression [42]. By constructing a Euclidean distance matrix between 2D pose and 3D pose:

$$edm(y)_{m,n} = \left\| p_m - p_n \right\|_2 \tag{6}$$

Among them, edm is the Euclidean distance matrix, y is the 3D pose, $y \in R^{3\times N}$, $edm(y) \in R^{N\times N}$, N is the number of key points, and p_m is the spatial coordinates of the m th point. Similarly, construct the 2D pose Euclidean distance matrix $edm(x)$, where $x \in R^{2\times N}$. This transforms the original regression problem from $R^{2\times N} \to R^{3\times N}$ to $R^{N\times N} \to R^{N\times N}$. The representation of the Euclidean distance matrix is invariant under rotation, translation, and reflection, and the Euclidean distance matrix is a symmetric matrix, allowing the parameters of half the matrix to be estimated using a fully connected network. This paper proposes two approaches for estimating the Euclidean distance matrix: a fully connected model similar to Martinez's and a fully convolutional model, both of which achieve good estimation performance.

For 3D pose estimation tasks, the commonly used evaluation metric is the mean perpendicular joint position error (MPJPE), which calculates the Euclidean distance between the predicted 3D coordinates and the true 3D coordinates for each joint. A smaller MPJPE value indicates better model performance:

$$MPJPE = \frac{1}{N}\sum_{i=1}^{N}\left\| J_i - J_i^* \right\|_2 \tag{7}$$

where N is the total number of joints, J_i and J_i^* are the true 3D coordinates and predicted 3D coordinates of the i th joint, respectively.

III. Human skeleton posture and motion recognition based on graph convolutional neural networks

III. A. Feature extraction of human skeleton data based on graph convolutional neural networks

III. A. 1) **Feature extraction from human skeleton models**

The key to human motion recognition algorithms is feature extraction from skeletal data (information aggregation between joints). This data is then fed into a pooling layer and classifier for motion recognition classification [43]. Since skeletal data is graph-type data, using a graph convolutional neural network for feature extraction is the optimal solution. The principle of feature extraction from skeletal data is as follows:

The principle of feature extraction is as follows: the main joints of the human body are identified. Let the input human skeleton data be f_{in}; based on structural relationships, the adjacency matrix A_j can be calculated.

The adjacency matrix is normalized. According to graph convolution theory, the traditional graph convolution kernel A is obtained:

Let the features after graph convolution be f_{out}, and the simplified graph convolution formula for the human skeleton model is obtained:

$$f_{out} = Af_{in}W \tag{8}$$

Among them, matrix A is the convolution kernel of graph convolution and is the core of the entire graph convolution network. W is the learnable weight parameter matrix.

III. A. 2) **Skeletal pose action recognition based on graph convolutions**

In the process of action recognition, the majority of the original input data consists of video streams. Therefore, when using graph convolutional neural networks, it is necessary to preprocess the input data. The input data type is converted into a skeleton spatiotemporal map. The construction of the spatiotemporal map involves estimating the pose of each frame in the video to extract skeleton data, followed by stacking the skeleton data for each frame. The resulting skeleton spatiotemporal map has two dimensions: spatial and temporal.

Figure 3 shows the overall process structure of the STGCN spatio-temporal graph convolution module, with the main inputs being spatio-temporal graph data and a convolution kernel matrix A constructed based on the human skeleton. First, the spatio-temporal map is convolved with a graph convolutional kernel covered by a mask. Then, the result is added to the convolutional output via ordinary convolution and a BN layer using residual

connections to enhance stability. Finally, the output is fed into a TCN for temporal convolution, completing a full spatio-temporal feature extraction.

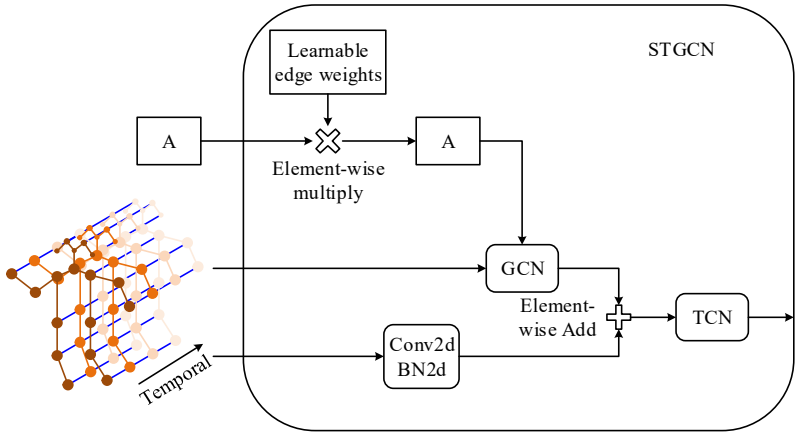


Figure 3: The internal process of the spatio-temporal graph convolution module

The process of extracting features from the internal spatial GCN of the action recognition algorithm based on skeletal data (taking STGCN as an example) is assumed to input a spatiotemporal graph data X. X is a four-dimensional tensor, including batchsize set to N, feature dimension C, number of time frames T, and number of joint points V. Assuming that the number of channels in the next layer of the current network model is 64, the convolution operation will increase the feature dimension of the data to 64 times, and then use the dimension transformation operation to split the bone data to expand an additional dimension K, and the number of channels of the split dimension K will be the same as the convolution kernel $C*V*V$ (STGCN The convolution kernel is a matrix of $3*25*25$, where $V=25$ is the number of joint points, $C=3$ is the number of channels in the feature dimension C of the three subgraphs divided by the partition strategy, and C also represents the number of convolution kernels. In the STGCN algorithm, due to the partitioning strategy, $C=3$, the new five-dimensional tensor and the convolution kernel perform matrix multiplication operation, so that a graph convolution is completed, and the $C=64$ after the graph convolution is completed, which is conducive to the network to capture spatial action information.

III. B. ***Image Convolution Module Data Processing***

Let any graph convolution operation be treated as a nonlinear function:

$$H^{l+1}=F\left(H^{l},A\right) \tag{9}$$

Among them, $H^{0}=X$ is the first layer input, $X\in R^{N\times D}$, N is the number of points in the graph, D represents the feature dimension, and A is the adjacency matrix, which contains information about each node and its neighboring nodes, i.e., the direction of the feature extraction channel. The difference between different graph convolutions lies in the implementation of method F. A classic graph convolution method involves averaging and weighting neighbor information, followed by information aggregation.

For the specific implementation of graph convolution, let W^{l} be the weight parameter matrix of the lth layer, and σ be a nonlinear activation function, such as ReLU. The implementation approach is very straightforward. The features of a node are influenced by the features of all its neighbors. The adjacency matrix A is multiplied by the features H, combined with the weight parameters and activation function. The resulting node features are the aggregated information features of the neighboring nodes. Multiple convolutions can obtain higher-order features of neighboring nodes. The implementation method is as follows:

$$H^{l+1}=\sigma\left(AH^{l}W^{l}\right) \tag{10}$$

However, the above methods have two problems:

(1) They do not take into account the influence of the node's own characteristics.

(2) Neighboring nodes are not normalized, which may lead to the structure of the graph itself having a greater influence than the node characteristics when extracting graph data features. For example, nodes with many neighboring nodes have more influential characteristics.

To address these two issues, a new graph convolution implementation method is proposed:

$$H^{l+1}=\sigma\left(LH^{l}W^{l}\right) \tag{11}$$

In equation (11), the Laplacian matrix is defined as $L=D-A$, where D is the degree matrix of the graph. By the self-definition of the degree matrix, it is known that D contains information about the nodes themselves. Introducing the degree matrix solves the problem that graph convolution does not include information about the nodes themselves. Further optimization of L yields the current mainstream graph convolution method:

$$H^{l+1}=\sigma\left(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H^{l}W^{l}\right) \tag{12}$$

The Laplacian matrix $L=D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ in Equation (12) is essentially an improvement on the above two methods. First, add the self-degree matrix to ensure that the node's own information is not lost. Then, normalize the self-adjacency matrix (multiply both sides of the adjacency matrix by the inverse square root of the degree matrix) to obtain it.

Specifically, each node in matrix L for i,j can be obtained by the following equation:

$$L_{i,j}^{sym}=\begin{cases}1 & i=j \ and \ \deg(v_i)\neq 0\\ -\dfrac{1}{\sqrt{\deg(v_i)\deg(v_j)}} & i\neq j \ and \ v_i \ is \ adjacent \ to \ v_j\\ 0 & otherwise\end{cases} \tag{13}$$

In the formula, $\deg(v_i)$ and $\deg(v_j)$ are the degrees of nodes i,j respectively, which are the values of the degree matrix at the nodes.

III. C. Action recognition based on 3D convolutional networks

III. C. 1) 3D Convolutional Networks

In 2DCNN, the convolution operation of the convolution layer generally consists of three parts. First, feature extraction is performed, then the extracted features are added to the bias, and finally they are passed into the activation function. Generally, the value of the j th feature map of the i th layer at position (x,y) is denoted as $c_i^j(x,y)$, and the convolution formula is shown in Equation (14):

$$c_i^j(x,y)=\sigma\left(b_i^j+\sum_{m=0}^{M-1}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}w_{im}^j(p,q)c_{i-1}^m(x+p,y+q)\right) \tag{14}$$

In the equation, σ denotes the activation function, b_i^j denotes the bias of this layer, m denotes the index of the feature map set of the $(i-1)$ th layer connected to the current feature map, $w_{im}^j(p,q)$ denotes the value of the kernel connected to the m th feature map at position (p,q), P_i and Q_i are the height and width of the convolution kernel, respectively.

In 2DCNN, features are calculated solely from the spatial dimension. When dealing with time-series problems, such as human motion capture, 2D convolution cannot capture motion information in the temporal dimension. Therefore, 3D convolution was developed to calculate features simultaneously from both the temporal and spatial dimensions. 3D convolution is achieved by performing convolution operations on a cube composed of multiple consecutive video frames using a 3D convolution kernel. Through this construction, the feature maps in the convolution layer are connected to multiple consecutive frames in the previous layer, thereby capturing motion information. Generally, the value of the j th feature map in the i th layer at position (x,y,z) is denoted as $c_i^j(x,y,z)$, and the 3D convolution is shown in Formula (15):

$$c_i^j(x,y,z)=\sigma\left[b_i^j+\sum_{m=0}^{M-1}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}\sum_{r=0}^{R_i-1}w_{im}^j(p,q,r)c_{i-1}^m(x+p,y+q,z+r)\right] \tag{15}$$

In the equation, R_i is the size of the 3D kernel in the time dimension, $w_{im}^j(p,q,r)$ is the (p,q,r) th value of the kernel connected to the m th feature map in the previous layer, and σ represents the activation function.

III. C. 2) Action recognition framework based on 3D CNN

Based on the above 3D convolution description, a 3DCNN framework for human action recognition is generated. Here, seven consecutive frames centered on the current frame, each with a size of 60×40, are used as input to the 3DCNN model. Then, a set of hard-wired kernels is used to generate multi-channel information from the input frames. The 33 feature maps in the second layer contain five distinct channels: $gray$, $gradient-x$, $gradient-y$, $optflow-x$, and $optflow-y$. The $gray$ channel contains the grayscale pixel values of the seven input frames,

while the $gradient-x$ and $gradient-y$ feature maps are obtained by calculating the horizontal and vertical gradients, respectively, while $optflow-x$ and $optflow-y$ channels represent the optical flow fields in the horizontal and vertical directions for each of the 7 input frames, respectively.

III. D. Construction of a 3D graph convolution action recognition model

After gaining a detailed understanding of the structure and construction methods of 3D convolutional networks, the next step is to apply the construction methods of 3D convolutional networks to 3D graph convolutional networks. By constructing 3D graph convolutional operators, an action recognition model is built. This model is then used to extract features from human skeleton sequences and classify them, thereby achieving action recognition. When constructing the action recognition model, this section employs dilated convolution to expand the receptive field during feature extraction and uses the SGDR gradient descent method to train the model, enabling it to escape local optima during training and maximize the likelihood of finding the global optimum.

III. D. 1) 3D Spatio-Temporal Convolution Module

The spatio-temporal graph convolution action recognition model introduced in Section 3.1.2 is composed of a series of spatio-temporal graph convolution blocks, each of which consists of a spatial convolution and a temporal convolution. These blocks alternately extract spatial and temporal features from the human skeleton sequence. To enable the extraction of temporal features in the spatio-temporal graph convolution, this section designs a 3D spatio-temporal graph convolution network. During the spatial graph convolution process, the coordinates of the next frame's 3D or 2D joint nodes are included. That is, the spatial graph convolution in this section performs graph convolution (3D-GCN) on the skeleton graphs of two consecutive frames, followed by convolution in the spatio-temporal dimension. The spatial graph convolution operation is a crucial component of 3D-GCN, essentially performing a weighted average of each joint's neighboring features and its own features from the next frame.

The spatial method is used to divide the neighborhood into three parts: the root node itself, the centripetal nodes, and the centrifugal nodes. To avoid conflicts with symbols used in previous sections, new symbols are redefined in this section. Based on the spatial division of the neighborhood, the adjacency matrix A can be successively divided into A^{root}, A^{cprl}, and A^{ctfg}. Let $G=\{root, ctpl, ctfg\}$ denotes the set of neighborhood types resulting from the partition, then $\sum_{g\in G} A^{(g)} = A$. Additionally, this section defines $\chi \in R^{v\times c\times T}$ to represent the coordinates of all 3D or 2D skeleton nodes in T video frames, where v denotes the number of human joint nodes in a frame, and c denotes the number of features per joint node. For example, in the NTU-RGB-D dataset, a frame contains $v=25$ joint nodes, and each joint node has $c=3$ features, namely the x, y, and z coordinates. $f_{in} \in R^{v\times c\times 2}$ represents the coordinates of all 3D or 2D joint nodes in consecutive 2 frames, and $x_t^i \in R^c$ represents the coordinate vector of the i th joint node in the t th frame. $f_{out} \in R^{d_{out}\times 2}$ represents the features obtained after 3D spatial graph convolution, where d_{out} is the dimension of the output features. The 3D spatial graph convolution is shown in formula (16):

$$f_{out} = \sum_{g\in G} M^{(g)} \circ \tilde{L}^{(g)} f_{in} W^{(g)} \tag{16}$$

In the equation, $\tilde{L}^{(g)} = D^{(g)-\frac{1}{2}} A^{(g)} D^{(g)-\frac{1}{2}} \in R^{n\times n}$ is the symmetric normalized adjacency matrix of each partition, $\circ$ denotes the Hadamard product. $M^{(g)} \in R^{n\times n}$ is the trainable matrix used to capture edge weights for each partition of the partitioning, and $W^{(g)} \in R^{d_{out}\times 2}$ is the trainable feature weight matrix.

III. D. 2) 3D Convolutional Action Recognition Model

This model consists of 9 3D spatio-temporal convolutional unit layers, each containing 1 3DGCN module and 1 TCN module. The number of output channels for layers 0 to 2 is 64, for layers 3 to 5 is 128, and for layers 6 to 8 is 256. The TCN in layers 3 and 6 has a stride of 2, functioning as a pooling layer. The number of output channels is adjusted using a 1×1 convolution kernel. Each layer of the 3D spatio-temporal graph convolutional network applies a residual mechanism. To prevent overfitting, the Dropout parameter for each layer is set to 0.5. The final output feature map is then subjected to global pooling, and the result is input into a SoftMax classifier to output the action category.

III. D. 3) SGDR Gradient Descent Method and Model Training

The SGDR learning rate adjustment strategy can be roughly divided into three steps: (1) First, determine the upper and lower bounds of the learning rate and the cycle period step size. (2) According to the following formula (17), reduce the learning rate to the minimum value at the end of each training step using the maximum learning rate and the set cycle period step size, and then restore the learning rate to the maximum value. (3) Repeat step 2 until training is complete.

$$\gamma_t = \gamma_{\min}^i + \frac{\gamma_{\max}^i - \gamma_{\min}^i}{2}\left(1+\cos\left(\frac{T_{cur}}{T_i}\pi\right)\right) \tag{17}$$

Among these, $\gamma_{\max}^i$ and $\gamma_{\min}^i$ represent the range of the learning rate, and T_{cur} denotes the number of training cycles since the last restart. T_{cur} is updated after each batch of training is completed. When $t=0$ and $T_{cur}=0$, we have $\gamma_t = \gamma_{\max}^i$. When $T_{cur} = T_i$, the output of the cosine function is -1, and we have $\gamma_t = \gamma_{\min}^i$. The parameters for the SGDR method used in this paper are set as follows: $T_0 = 10$, $T_{mult} = 1$, the initial learning rate $\gamma_{\max^i}$ is 0.1, and the minimum learning rate $\gamma_{\min^i}$ is 10^{-6}.

IV. Analysis of human skeletal joint posture estimation and motion recognition results

IV. A. Analysis of simulation results

IV. A. 1) Introduction to the dataset

NTU-RGB+D is currently one of the largest and most widely used datasets in human action recognition research. The dataset includes 50 action categories with a total of 55,000 video action samples, all of which are composed of 3D joint coordinate sequences. All video samples were recorded by 50 volunteers, with each action captured by four cameras positioned at the same height but at different horizontal angles (-45°, 0°, and 45°). The volunteers' ages ranged from 10 to 35 years old. Human behavior action samples were collected using a Kinect depth camera. The action samples consist of continuous skeleton sequences, each containing 25 joint coordinate data points for each object, with no more than two test subjects in each video. The dataset can be divided into two benchmarks based on different detection objects and different perspectives:

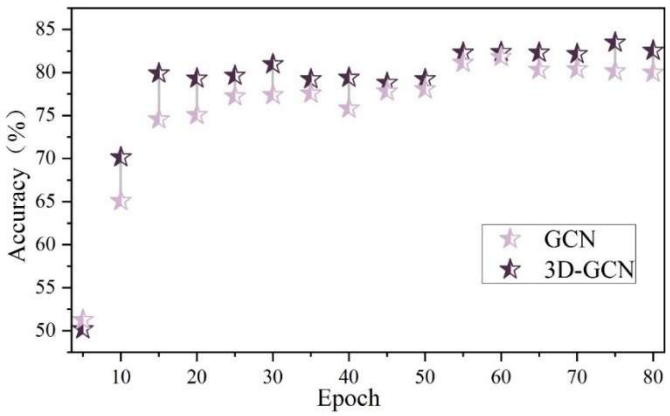
(1) Cross-Subject: Under this standard, the training set contains 40,548 video samples and the test set contains 16,780 video samples, with the training set and test set recorded by different test subjects.

(2) Cross-View: In this benchmark, the training set contains 37,980 video samples, and the test set contains 18,930 video samples. The training set was captured by cameras 2 and 3, while the test set was captured by camera 1. Following this convention, report the recognition rates for the two benchmarks under the Top-1 and Top-5 metrics.

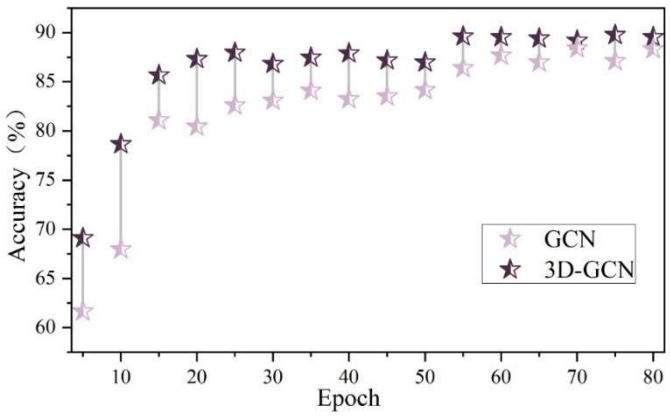
The Kinetics human action dataset includes 320,000 action videos from various complex scenarios such as daily life, sports, and entertainment, with over 500 distinct action types. To standardize the action samples, each action clip lasts approximately 15 seconds. The data in this dataset is not raw RGB video data but rather processed skeleton joint coordinate sequences. Unlike the NTU-RGBD dataset, the skeleton joint data in the Kinetics dataset is 2D, whereas it is 3D in NTU-RGBD. This paper uses the previously released Kinetics dataset to evaluate the proposed model. The dataset is divided into a training set (250,000 clips) and a test set (180,000 clips). The model is trained on the training set and evaluated on the test set using Top-1 and Top-5 accuracy metrics.

IV. A. 2) Experiments and Analysis on the NTU-RGBD Dataset

This paper proposes a 3D-GCN (3D Graph Convolutional Network) based on 3D space and tests it under the CS and CV standards of the NTU-RGBD dataset. Action features are extracted using the 3D-GCN, and the captured action features are finally classified through a convolutional function. Figure 4 shows the comparison of recognition rates between the GCN and the proposed 3D-GCN model. Figure (a) shows the recognition rate comparison of the models under the CS standard, and Figure (b) shows the recognition rate comparison under the CV standard. The recognition accuracy rates of the GCN and 3D-GCN models under the CS standard are 75.853% and 78.251%, respectively. Under the CV standard, the recognition accuracy rates are 82.294% and 86.381%, respectively.



(a) Comparison of recognition rates of the model under CS

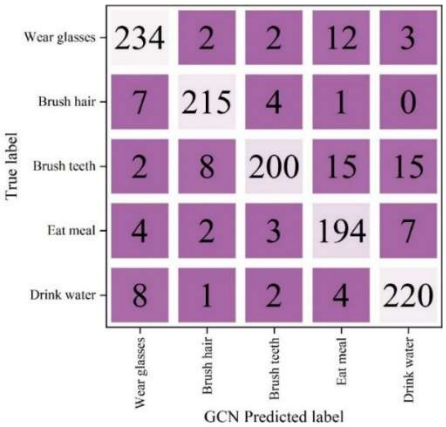


(b) The comparison of the model in CV

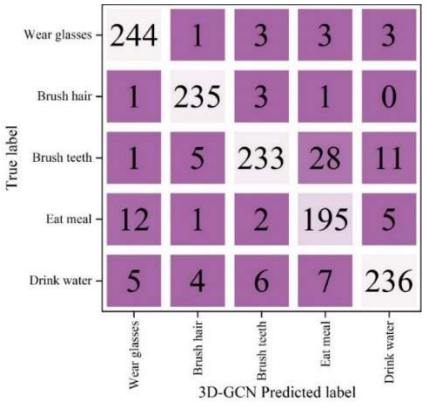
Figure 4: GCN compares the identification rate of the 3D-GCN model proposed

As shown by the comparison of recognition rates above, the model proposed in this paper, 3D-GCN, achieves better recognition performance than GCN under both CS and CV standards after incorporating 3D technology. This preliminarily demonstrates that the 3D graph convolutional network proposed in this paper is effective. To further validate that the incorporation of 3D technology can improve the model's fixed physical structure and thereby enhance recognition of certain specific action categories, this paper selected five relatively similar action categories from the NTU-RGBD dataset for experimentation. These categories are “drinking water,” “eating food,” “brushing teeth,” “washing hair,” and “wearing glasses.”

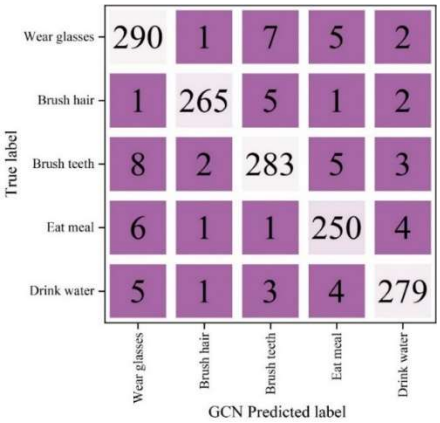
Figure 5 shows the confusion matrices for the model's recognition of similar actions. Among them, (a) and (c) are the confusion matrices for the GCN model without 3D in the CS and CV evaluation metrics for similar action recognition. (b) and (d) are the confusion matrices obtained by the 3D-GCN model after incorporating 3D data. As shown in the figure, for the same action category, the recognition performance of (b) and (d) is significantly better than that of (a) and (c). Taking the action “drinking water” as an example, among the 280 test samples under the CS standard, the original GCN model correctly identified 220 action samples, resulting in an action recognition rate of only 78.571%. Similarly, for the action “drinking water,” among the 320 test samples under the CV standard, the original GCN model achieved an action recognition rate of 87.188%. However, for the same action, the 3D-GCN model with 3D technology achieved recognition rates of 84.286% and 90.313% under the two standards, respectively, which are 5.715% and 3.125% higher than the GCN model. This is because the addition of 3D creates a new connection between two physically unconnected nodes, giving the model greater flexibility to recognize certain similar actions.



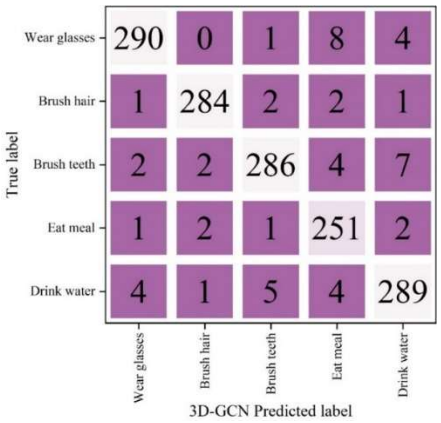
(a)The prediction results of GCN on CS



(b)The prediction results of 3D-GCN on CS



(c)The prediction results of GCN on CV

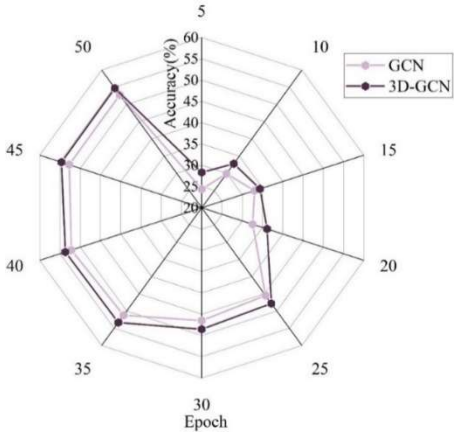


(d)The prediction results of 3D-GCN on CV

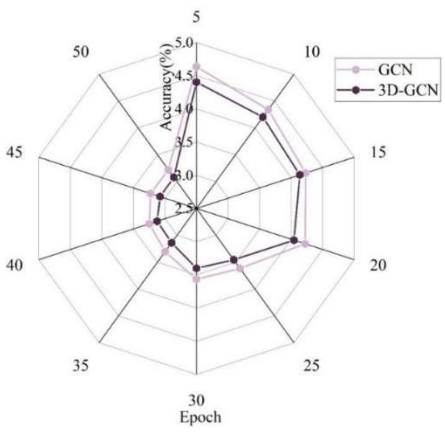
Figure 5: The model identifies the confusion matrix of similar actions

IV. A. 3) Experiments and Analysis on the Kinetics Dataset

To further validate the action recognition performance and generalization ability of the 3D graph convolutional network (3D-GCN) model, this chapter adds the Kinetics dataset for comparative experiments. This dataset contains as many as 500 action categories and has extremely rich application scenarios, effectively verifying the model's recognition performance in different scenarios and among different populations. During the experiments, the model was first trained using 250,000 samples from the training set to obtain the corresponding model. Then, the model was tested on 180,000 samples from the test set to obtain the final recognition performance. Figure 6 shows the model performance comparison, with Figure (a) showing the recognition accuracy comparison, Figure (b) shows the comparison of the loss function. The Top-5 final recognition rate of the 3D-GCN model is 44.425%, which is higher than the 42.067% of the GCN model, resulting in an overall improvement of approximately 2.4% in recognition accuracy. Additionally, during each epoch of training, the 3D-GCN model consistently achieved higher recognition rates than the GCN model. In the loss curve comparison in Figure (b), it is evident that the loss function values of the 3D-GCN model decrease more rapidly and to a greater extent after incorporating 3D features. At the 20th epoch, the 3D-GCN model's recognition rate and loss function value experienced significant fluctuations. This was due to a change in the learning rate at this point, which allowed the model to obtain a set of optimal weight parameters locally, thereby significantly improving model performance. As training progressed, the recognition rate and loss function curve gradually stabilized around the 45th epoch, with a loss value of 3.077. Through model comparison experiments on the Kinetics dataset, it is demonstrated that the introduction of 3D technology plays a significant role in improving the physical structure of the model and enhancing action recognition performance.



(a)Comparison of spoon recognition accuracy



(b)Loss function comparison

Figure 6: Performance comparison of spoon models

IV. A. 4) Ablation experiment

Table 1 presents the ablation experiment results. In the table, MS-DGCN-J denotes inputting only joint information, MS-DGCN-B denotes inputting only skeletal information, MS-DGCN-M denotes inputting only motion information, MS-DGCN-J+B denotes inputting both joint and skeletal information, and MS-DGCN-J+M denotes inputting both joint and motion information. MS-DGCN-B+M denotes input of skeletal information and motion information. The method proposed in this paper uses input of joint information, skeletal information, and motion information. The recognition rates of the proposed method on the Top-1 and Top-5 metrics are 91.248% and 97.187%, respectively, both ranking first among all methods. Experiments show that using two input information streams yields better recognition performance than using one stream, and using three streams yields better performance than using two.

Table 1: Ablation experiment results

Method	Top-1	Top-5
ST-GCN	81.485	88.348
2s-AGCN	88.548	95.164
DGNN	89.948	96.139
MS-DCN-J	90.854	96.748
MS-DCN-B	88.945	95.135
MS-DCN-M	91.048	96.948
MS-DCN-J+B	90.785	96.748
MS-DCN-J+M	91.168	97.048
MS-DCN-B+M	89.596	96.485
This method	91.248	97.187

IV. B. Motion Recognition Analysis

IV. B. 1) Action Recognition Classification

The action types in the dataset are further categorized into three classes: 40 daily activity actions, 9 health-related actions, and 11 multi-person interaction actions. Additionally, to evaluate the performance of the 3D-GCN network architecture in recognizing different types of actions, experiments were conducted to measure recognition accuracy across the 60 action categories in the X-Sub dataset. The experimental results are shown in Figure 7, where the horizontal axis represents the ID numbers of the 60 action categories in the NTU RGB+D 60 dataset, and the vertical axis represents the accuracy rates for each action category. As shown in the table, while the 3D-GCN network can accurately identify most action categories in the NTU RGB+D 60 dataset, However, for certain specific action categories, such as 10 (clapping), 11 (reading), 12 (writing), 26 (hopping on one foot), and 42 (staggering), the recognition accuracy rates are 74.604%, 49.548%, 37.868%, 64.876%, and 70.079%, respectively, which are significantly lower than those of other action categories.

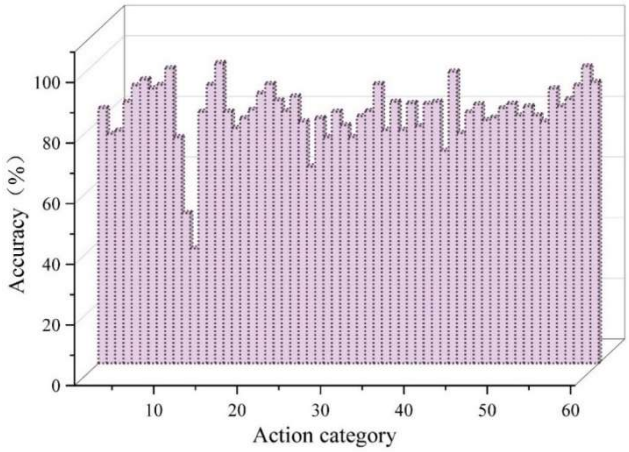


Figure 7: 3D-GCN network is accurate in 60 actions on the x-sub data set

IV. B. 2) Evaluation Index Results for Different Actions

This paper optimizes the model through training, obtains the optimal model, and then tests it on the test set. The experimental results are shown in Table 2. The algorithm achieves an accuracy rate of 91.941% for recognizing

four types of movements (falling, running, kicking, and squatting) and an F1 score of 89.348%, thereby improving the original algorithm's performance in recognizing human movements to a certain extent.

Table 2: The evaluation index results of different actions

/	P(%)	R(%)	F1(%)	A(%)
Fall down	86.758	84.378	85.548	89.948
Run	93.248	90.648	91.948	93.818
Kick the leg	90.248	88.596	89.348	92.048
Squatting down	91.598	89.536	90.548	91.948
Average value	90.463	88.2895	89.348	91.9405

IV. B. 3) Evaluation metric results for different algorithms

The same data was used for training the AlphaPose+LSTM and AlphaPose+ST-GCN algorithms. During LSTM training, the epoch was set to 200, the batch size to 35, the number of steps to 480, and the learning rate to 0.002. The parameters for the original GCN training were set identically. The training results were compared with those of this paper (3D-GCN), as shown in Table 3. As can be seen, the accuracy, recall rate F1, and precision of the algorithm in this paper are 90.496%, 88.248%, 89.348%, and 92.089%, respectively, performing exceptionally well among the three algorithms and demonstrating excellent performance in human action recognition.

Table 3: The evaluation index results of different algorithms

/	P(%)	R(%)	F1(%)	A(%)
AlphaPose+LSTM	84.098	81.435	82.631	88.198
AlphaPose+ST-GCN	88.869	85.593	87.196	89.245
This method	90.496	88.248	89.348	92.089

V. Conclusion

This paper explores graph neural networks and human skeleton joint pose estimation algorithms, utilizing graph convolutional neural networks to estimate human skeleton poses and recognize actions. Pose features are calculated from both temporal and spatial dimensions, and a motion recognition model based on 3D graph convolutions is constructed, with SGDR gradient descent used for model training.

Simulation experiments were conducted on the NTU-RGBD and Kinetics datasets to test the model's practicality. Taking the action of “drinking water” as an example, the original GCN model achieved recognition rates of 78.571% and 87.188% for this action in test samples under the CS and CV standards, respectively. The recognition rates of 3D-GCN under the two standards were 84.286% and 90.313%, respectively, which were 5.715% and 3.125% higher than those of the GCN model in terms of recognition rate.

To test the 3D-GCN network structure for different types of action recognition, experiments were conducted on the recognition accuracy of 60 action categories in the X-Sub dataset, and the recognition rates of most actions were above 80%. The precision, recall, F1 score, and accuracy of the proposed algorithm are 90.496%, 88.248%, 89.348%, and 92.089%, respectively, demonstrating outstanding algorithm performance and effective human action recognition.

References

[1] Gu, Y., Xu, Y., Shen, Y., Huang, H., Liu, T., Jin, L., ... & Wang, J. (2022). A review of hand function rehabilitation systems based on hand motion recognition devices and artificial intelligence. Brain sciences, 12(8), 1079.

[2] Guo, H., Zhang, Z., Yu, R., Sun, Y., & Li, H. (2023). Action recognition based on 3D skeleton and LSTM for the monitoring of construction workers' safety harness usage. Journal of Construction Engineering and Management, 149(4), 04023015.

[3] Shan, S., Sun, S., & Dong, P. (2023). Data driven intelligent action recognition and correction in sports training and teaching. Evolutionary Intelligence, 16(5), 1679-1687.

[4] Ali, M. A., Hussain, A. J., & Sadiq, A. T. (2022). Human body posture recognition approaches. ARO-The Scientific Journal of Koya University, 10(1), 75-84.

[5] Morshed, M. G., Sultana, T., Alam, A., & Lee, Y. K. (2023). Human action recognition: A taxonomy-based survey, updates, and opportunities. Sensors, 23(4), 2182.

[6] Ali, H. H., Moftah, H. M., & Youssif, A. A. (2018). Depth-based human activity recognition: A comparative perspective study on feature extraction. Future Computing and Informatics Journal, 3(1), 51-67.

[7] Peng, C., Huang, H., Tsoi, A. C., Lo, S. L., Liu, Y., & Yang, Z. Y. (2020). Motion boundary emphasised optical flow method for human action recognition. IET Computer Vision, 14(6), 378-390.

[8] Uddin, M. Z., Khaksar, W., & Torresen, J. (2018, October). Activity recognition using deep recurrent neural network on translation and scale-invariant features. In 2018 25th IEEE International Conference on Image Processing (ICIP) (pp. 475-479). IEEE.
[9] Bagautdinov, T., Alahi, A., Fleuret, F., Fua, P., & Savarese, S. (2017). Social scene understanding: End-to-end multi-person action localization and collective activity recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4315-4324).
[10] Zhang, Y., Guo, Q., Du, Z., & Wu, A. (2023). Human action recognition for dynamic scenes of emergency rescue based on spatial-temporal fusion network. Electronics, 12(3), 538.
[11] Zhang, Y., Wang, X., Wang, Y., & Chen, H. (2020). Human activity recognition across scenes and categories based on CSI. IEEE Transactions on Mobile Computing.
[12] Yue, R., Tian, Z., & Du, S. (2022). Action recognition based on RGB and skeleton data sets: A survey. Neurocomputing, 512, 287-306.
[13] Manzi, A., Fiorini, L., Limosani, R., Dario, P., & Cavallo, F. (2018). Two-person activity recognition using skeleton data. IET computer Vision, 12(1), 27-35.
[14] Wang, Z., Ma, M., Feng, X., Li, X., Liu, F., Guo, Y., & Chen, D. (2022). Skeleton-based human pose recognition using channel state information: A survey. Sensors, 22(22), 8738.
[15] Xu, Y., Wei, S., & Yin, J. (2024). Optimization of human posture recognition based on multi-view skeleton data fusion. Information Technology and Control, 53(2), 542-553.
[16] Yan, X., Li, H., Wang, C., Seo, J., Zhang, H., & Wang, H. (2017). Development of ergonomic posture recognition technique based on 2D ordinary camera for construction hazard prevention through view-invariant features in 2D skeleton motion. Advanced Engineering Informatics, 34, 152-163.
[17] Hu, K., Jin, J., Zheng, F., Weng, L., & Ding, Y. (2023). Overview of behavior recognition based on deep learning. Artificial intelligence review, 56(3), 1833-1865.
[18] Veličković, P. (2023). Everything is connected: Graph neural networks. Current Opinion in Structural Biology, 79, 102538.
[19] Ogoke, F., Meidani, K., Hashemi, A., & Farimani, A. B. (2021). Graph convolutional networks applied to unstructured flow field data. Machine Learning: Science and Technology, 2(4), 045020.
[20] Jiang, X., Zhu, R., Ji, P., & Li, S. (2020). Co-embedding of nodes and edges with graph neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(6), 7075-7086.
[21] Raj, M. S., George, S. N., & Raja, K. (2024). Leveraging spatio-temporal features using graph neural networks for human activity recognition. Pattern Recognition, 150, 110301.
[22] Li, M., Chen, S., Chen, X., Zhang, Y., Wang, Y., & Tian, Q. (2021). Symbiotic graph neural networks for 3d skeleton-based human action recognition and motion prediction. IEEE transactions on pattern analysis and machine intelligence, 44(6), 3316-3333.
[23] Ma, R., Zhang, Z., & Chen, E. (2021). Human motion gesture recognition based on computer vision. Complexity, 2021(1), 6679746.
[24] Wong, P. K. Y., Luo, H., Wang, M., Leung, P. H., & Cheng, J. C. (2021). Recognition of pedestrian trajectories and attributes with computer vision and deep learning techniques. Advanced Engineering Informatics, 49, 101356.
[25] Wang, R., Cao, Z., Wang, X., Xue, W., & Cao, W. (2018). GA-STIP: Action recognition in multi-channel videos with geometric algebra based spatio-temporal interest points. IEEE access, 6, 56575-56586.
[26] Al-Faris, M., Chiverton, J., Ndzi, D., & Ahmed, A. I. (2020). A review on computer vision-based methods for human action recognition. Journal of imaging, 6(6), 46.
[27] Andrade-Ambriz, Y. A., Ledesma, S., Ibarra-Manzano, M. A., Oros-Flores, M. I., & Almanza-Ojeda, D. L. (2022). Human activity recognition using temporal convolutional neural network architecture. Expert Systems with Applications, 191, 116287.
[28] Wang, M., Zhang, Y. D., & Cui, G. (2019). Human motion recognition exploiting radar with stacked recurrent neural network. Digital Signal Processing, 87, 125-131.
[29] Hernandez, V., Suzuki, T., & Venture, G. (2020). Convolutional and recurrent neural network for human activity recognition: Application on American sign language. PloS one, 15(2), e0228869.
[30] Ding, J., & Wang, Y. (2019). WiFi CSI-based human activity recognition using deep recurrent neural network. IEEE Access, 7, 174257-174269.
[31] Dai, C., Liu, X., & Lai, J. (2020). Human action recognition using two-stream attention based LSTM networks. Applied soft computing, 86, 105820.
[32] Sun, X., Xu, H., Dong, Z., Shi, L., Liu, Q., Li, J., ... & Wang, Y. (2022). Capsganet: Deep neural network based on capsule and gru for human activity recognition. IEEE Systems Journal, 16(4), 5845-5855.
[33] Lu, C. (2021). Multifeature fusion human motion behavior recognition algorithm using deep reinforcement learning. Mobile Information Systems, 2021(1), 2199930.
[34] Song, J., Zhu, A., Tu, Y., Wang, Y., Arif, M. A., Shen, H., ... & Cao, G. (2020). Human body mixed motion pattern recognition method based on multi-source feature parameter fusion. Sensors, 20(2), 537.
[35] Zhang, C., Chen, X. P., Han, G. Q., & Liu, X. J. (2023). Spatial transformer network on skeleton-based gait recognition. Expert Systems, 40(6), e13244.
[36] Phyo, C. N., Zin, T. T., & Tin, P. (2019). Deep learning for recognizing human activities using motions of skeletal joints. IEEE Transactions on Consumer Electronics, 65(2), 243-252.
[37] Vernikos, I., & Spyrou, E. (2025). Skeleton Reconstruction Using Generative Adversarial Networks for Human Activity Recognition Under Occlusion. Sensors, 25(5), 1567.
[38] Mondal, R., Mukherjee, D., Singh, P. K., Bhateja, V., & Sarkar, R. (2020). A new framework for smartphone sensor-based human activity recognition using graph neural network. IEEE Sensors Journal, 21(10), 11461-11468.
[39] Xu Song & Hongyu Zhou. (2025). Improved Faster Region-based Convolutional Neural Network in graphic recognition and retrieval. Journal of Computational Methods in Sciences and Engineering,25(2),1341-1353.
[40] Hethu Avinash Dasari & Rammohan A. (2024). A novel image recognition based fault diagnostics of customized EV battery pack using optimized GoogLeNet. Engineering Research Express,6(3),035363-035363.
[41] Liu Liujun,Yang Jiewen,Lin Ye,Zhang Peixuan & Zhang Lihua. (2024). 3D human pose estimation with single image and inertial measurement unit (IMU) sequence. Pattern Recognition,149,110175-.

[42] Otgonbayar Agvaan,Gordon Cichon,Uuganbaatar Dulamragchaa,Hyun chul Kim,Seonuck Paek & Tseren Onolt Ishdorj. (2025). Optimal approximate computation of Euclidean distance in spiking neural P systems framework. Scientific Reports,15(1),18047-18047.
[43] Peng Zhang,Xinlei Zhao,Lijia Dong,Weimin Lei,Wei Zhang & Zhaonan Lin. (2024). A framework for detecting fighting behavior based on key points of human skeletal posture. Computer Vision and Image Understanding,248,104123-104123.