2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

A Deep Learning Model for 3D Skeleton-Based Temporal Hand Gesture Localization

Jingtao Chen [1] and Jieyu Zhao [1] and Kedi Shen [1]

[1] Ningbo University

Abstract. Temporal action localization has recently garnered widespread attention. Current studies on temporal action localization mostly focus on 2D temporal action localization for the human body, with little research on more complex hand gestures. 2D-based temporal hand gesture localization has notable drawbacks, such as motion ambiguity and difficulty in handling complex gestures. To address these issues, we innovatively designed a 3D skeleton-based temporal action localization model, which processes 3D hand skeleton sequences and has a more robust capacity for learning and representing complex hand gestures. As far as we know, this is the first 3D skeleton-based method for temporal action localization.The model includes a backbone network, a temporal action localization module, and a classification head. The GCN-based backbone network is responsible for feature extraction from the skeleton sequence. The temporal action localization module fuses multi-scale temporal features through a pyramid structure, then performs action localization. The classification head predicts the action category. Additionally, we designed an innovative loss function to guide the model's learning. We validated our model on a self-constructed 3D hand skeleton dataset, and the results show that our model demonstrates good performance in temporal hand gesture localization.

Keywords: Temporal action localization, hand gesture, skeleton, multi-view.

1 Introduction

Unlike simple action recognition, temporal action localization involves not only identifying the action category but also determining the precise temporal boundaries of actions. This requires the model to effectively distinguish between target actions and background, necessitating a deep understanding of the spatial and temporal dependencies within the action sequences.

Early approaches involved manual feature extraction or key point detection [1]-[4], but these methods struggled to effectively handle complex actions. The rise of deep learning has been a breakthrough for pattern recognition, as studies show that utilizing deep learning networks [5]-[9] significantly improves the capacity for repre-senting and learning complex hand gestures. Most existing temporal action localiza-tion methods rely on 2D video data, where frame sequences are extracted from vide-os for recognition and detection [10]-[12]. These methods have limited capability for precise action localization tasks.Consequently, we shifted our focus to 3D methods.

On one hand, 3D action sequences inherently provide richer spatiotemporal semantics, facilitating a clearer distinction between different actions. On the other hand, 3D action representation can alleviate occlusion issues to some extent.

Skeleton-based representations provide a compact and efficient means of describing 3D action sequences, while graph structures are particularly suited to capture the topological spatial relationships inherent in skeleton data. Since the introduction of ST-GCN [13], graph-based approaches to action recognition have gained significant attention [14]-[17], demonstrating notable success in various applications. Therefore, we chose a graph-based skeleton representation method and developed our temporal hand gesture localization model.

Our model processes 3D skeleton input data in an end-to-end manner and consists of three main components: a backbone, a Temporal Action Localization module, and a Classification Head. The backbone takes skeleton action sequences as input and extracts meaningful 3D motion features. The Temporal Action Localization module identifies action boundaries by predicting the start and end frame indices, while the Classification Head determines the action category. The overall pipeline of our method is illustrated in **Fig. 1**. First, frames are extracted from the original video, and 3D skeleton data is extracted frame by frame. Then, the skeleton data is preprocessed and input into the model, ultimately producing the output results.

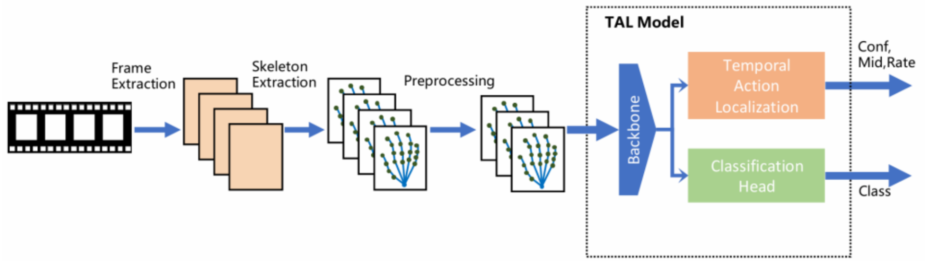

Fig. 1. The overall pipeline of the recognition algorithm.

Similar to 2D object detection tasks, temporal action localization can be viewed as a one-dimensional object detection problem, where the goal is to identify the action intervals along a time axis. Mainstream object detection methods [19] frame the task as a regression problem. Drawing inspiration from these methods, we formulated the temporal action detection task as a regression problem, where the model predicts frame indices to localize actions in time.

Research on 3D skeleton-based temporal action localization remains limited, and there is a lack of comprehensive related datasets. To validate our proposed model and methods, we collected a hand gesture dataset that includes skeleton sequences, class labels, and action localization annotations. Furthermore, we developed a comprehensive suite of tools for data annotation, skeleton extraction, and data augmentation, designed to support dataset construction, model training, and inference processes.

The contributions of this paper are summarized as follows:

- We propose a compact and efficient deep learning model for temporal action localization, specifically designed for hand gestures using 3D skeleton data.

- We collected and curated a 3D hand skeleton dataset to facilitate the evaluation of our model's performance.
- We trained and validated the model on our self-constructed dataset, achieving promising performance.

2 Related Work

2.1 Skeleton-based Method

Early approaches for skeleton-based methods [20]-[26] relied on Convolu-tional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks. These methods have achieved certain results, but they suffer from limited receptive fields and are unable to effectively handle long-term dependencies.

With the emergence of Graph Convolutional Networks (GCN) [27], an increasing number of studies have started to explore GCN-based methods. The initial ST-GCN [13] introduced the spatio-temporal graph convolutional structure, providing a new perspective for research in GCN-based methods. Since then, a large number of GCN-based approaches have emerged. Existing methods either focus on constructing different graph structures from a data perspective to capture richer spatio-temporal semantics [28], [29], or meticulously design network architectures to enhance learning performance [14], [30]-[32].

However, most current methods suffer from severe parameterization issues, resulting in low inference efficiency and limiting their practicality in real-world applications. The Efficient GCN [18] introduces a lightweight action recognition baseline. This method innovatively utilizes joint, velocity, and skeleton branches to process three distinct features, integrating them into the main stream via mid-level fusion, while employing separable convolutions to reduce the computational load.

2.2 Temporal Action Localization

Temporal action localization aims to accurately identify the start and end frame indices of actions based on the recognized action categories. This task requires high precision in learning action patterns and distinguishing the foreground from the background. Current mainstream methods [33]-[36] use feature pyramid networks (FPN) [37] for multi-scale learning of action features, followed by the extraction of temporal dimension features using various approaches.

3 Temporal Action Localization for Hand Gestures Based on 3D Skeletons

This section primarily discusses the implementation details of our method. The first part introduces the design details of the network, the second part describes the model's loss function, the third part introduces the data preprocessing methods, and the fourth part covers the training and inference processes.

3.1 Network Design

The network is divided into three main parts: the backbone network, Temporal Action Localization, and the Classification Head, as shown in the **Fig. 2**. In the figure, both the Temporal Action Localization and Classification Head modules receive the output features from the backbone network as input. The former is responsible for hand gesture localization, outputting the start and end frame indices of the action, while the latter is independent of the former and is responsible for predicting hand gesture categories. We have verified the necessity and effectiveness of this configuration through experiments, as detailed in the Experiment Section 4.3.

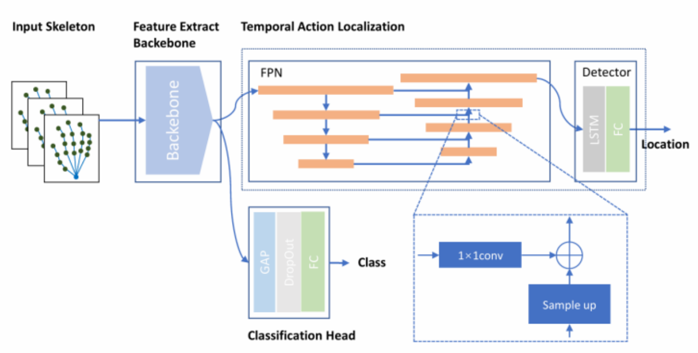

Fig. 2. Network structure schematic. It mainly includes a backbone, Temporal Action Localization module and classification head. The Temporal Action Localization includes a feature pyramid network(FPN), LSTM, and a full connection layer, while the Classification Head consists of global average pooling, dropout, and fully connected layers.

Backbone Network. The backbone network takes the 3D skeleton action sequence as input and extracts key spatiotemporal action features. To select the most suitable network as the backbone, we compared the performance of several GCN-based models and ultimately chose EfficientGCN-B0 as the backbone network, as detailed in the Experiment Section 4.2.

Temporal Action Localization. The Temporal Action Localization module consists of an FPN (Feature Pyramid Network) and a Detector. The FPN downsamples the features extracted by the backbone to different resolutions, and then performs upsampling and fusion step by step from lower to higher resolutions.

The specific implementation of FPN can be written as the following equation

$$Layer_{i+1} = AP(Layer_i) \tag{1}$$

$$f = Conv(Layer_{i+1}) \oplus Up(Layer'_{i+1}) \tag{2}$$

$$Layer_i' = AP(f) \tag{2}$$

$Layer_i$ represents the i-th layer in the top-down path on the left side, while $Layer_i'$ represents the i-th layer in the bottom-up path. Conv refers to a 1×1 convolution, Up indicates interpolation-based upsampling, and AP stands for global average pooling. The final output is $ayer_0'$, which is the highest resolution upsampling layer. Unlike traditional FPN networks, the FPN here only retains the highest resolution features as output, in order to reduce the computational burden.We can also see the structure of the FPN from the Temporal Action Localization module in **Fig. 2** .

By fusing features from different time steps, the FPN maximizes the extraction of multi-scale features, which helps generalize to actions with varying time spans. Finally, the fused features are passed to the Detector. The Detector consists of an LSTM and fully connected layers. The LSTM extracts features from the time series (Experiment Section 4.3 verified the effectiveness of LSTM feature extraction), and the fully connected layer outputs the localization predictions. By converting the action frame localization task into a regression problem, the detector only needs to predict the center frame of the action and the action proportion. The start and end frame indices can then be calculated.

Classification Head. The Classification Head includes a global average pooling layer, Dropout, and fully connected layers. A 3D average pooling layer is used for temporal downsampling to obtain video-level features. A 3D convolution implements the final fully connected layer, which is responsible for predicting the action class and outputting the action's one-hot encoding.

3.2 Loss Function

The model's loss function consists of three components: classification loss, localization loss, and confidence loss.

The classification loss $\mathcal{L}_{\text{class}}$ measures the discrepancy between the predicted categories and the actual categories, using cross-entropy as with other methods. As shown in the following equation.

$$\mathcal{L}_{\text{class}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\log\left(\hat{y}_{ic}\right) \tag{3}$$

N represents the number of samples, C represents the number of categories, y_{ic} represents the predicted probability that the i-th sample belongs to category c , and $\hat{y}_{ic}$ represents the actual probability that the i-th sample belongs to category c.

The intermediate frame loss $\mathcal{L}_{\text{mid}}$ is used to measure the discrepancy between the predicted positions of the intermediate frames within the action range and their actual positions. m represents the index of the predicted intermediate frame, and $\widehat{m}$ represents the index of the actual intermediate frame. As illustrated in the following equation,

$$\mathcal{L}_{\text{mid}} = \frac{1}{N}\sum_{i=1}^{N}\left(\widehat{m}_i - m_i\right)^2 \tag{4}$$

The action proportion loss measures the discrepancy between the predicted proportion of frames occupied by an action and the actual proportion. Here, r_i represents the predicted proportion of the total number of frames taken up by the action, and $\hat{r}_i$ represents the actual proportion. The square root is used to mitigate the differences caused by the duration of long or short actions.

$$\mathcal{L}_{\text{rate}} = \frac{1}{N}\sum_{i=1}^{N} (\sqrt{\hat{r}_i} - \sqrt{r_i})^2 \tag{5}$$

The confidence loss $\mathcal{L}_{\text{conf}}$ measures the gap between the actual confidence and the prediction, guiding the model to predict confidence that is more biased towards results with a higher Intersection over Union (IoU).

$$\mathcal{L}_{\text{conf}} = -\frac{1}{N}\sum_{i=1}^{N} c_i \text{IoU}_i \log(\hat{c}_i) \tag{6}$$

Here, c_i represents the model's output confidence, and $\hat{c}_i$ is set to 1, while IoU_i is the result of the Intersection over Union between the model's output action localization and the ground truth.

To balance the multiple losses, we set several hyperparameters, as shown in the following equation. We determined these hyperparameters through experiments, as described in Section 4.4.

$$\mathcal{L}_{\text{total}} = \lambda_{\text{class}}\mathcal{L}_{\text{class}} + \lambda_{\text{mid}}\mathcal{L}_{\text{mid}} + \lambda_{\text{rate}}\mathcal{L}_{\text{rate}} + \lambda_{\text{conf}}\mathcal{L}_{\text{conf}} \tag{7}$$

3.3 Data Preprocessing

Data Preprocessing. The input is constructed using the EfficientGCN [18] structure, forming a graph based on the topology of a 21-point hand skeleton model. The original 3D skeleton sequence is processed into joint positions, velocities, and bone segment characteristics, as illustrated in **Fig. 3**.

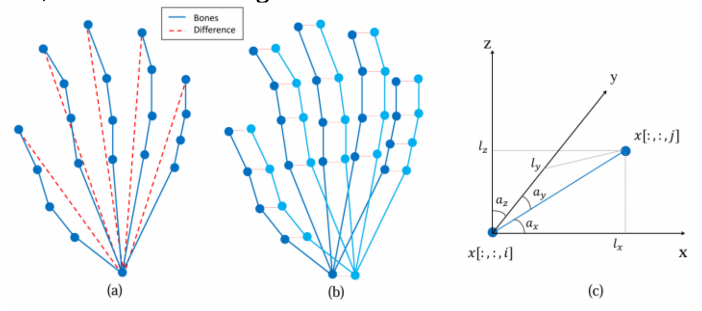

Fig. 3. Input data representation. (a) represents the relative position of joints, (b) represents joint motion velocity, and (c) represents the 3D length and angle of a bone segment.

Assume that the absolute position of the input joints is given by $\mathcal{X} = \{x \in \mathbb{R}^{C_{in} \times T_{in} \times V_{in}}\}$, where $\mathcal{R} = \{r_i | i = 1,2,\dots,V_{in}\}$ represents the relative position of each point in a single frame with respect to the wrist joint. Together, $\mathcal{X}$ and $\mathcal{R}$ form the joints. Joint motion velocity is divided into $\mathcal{F}$ and $\mathcal{S}$, where $\mathcal{F} = \{f_t | t = 1,2, \ldots, T_{in}\}$ and $= \{s_t | t = 1,2, \ldots, T_{in}\}$, calculated by subtracting the coordinates of corresponding joints in adjacent frames. The bones are represented by the three-dimensional bone length $\mathcal{L} = \{l_i | i = 1,2, \ldots, V_{in}\}$ and the three-dimensional angle $= \{a_i | \bar{\iota} = 1,2, \ldots, V_{in}\}$. The details are shown in the formulas. Here, i denotes the joint indices, and c represents the wrist joint index.

$$r_i \quad = x[:,:,i] - x[:,:,c] \tag{8}$$

In the following equation, t denotes the current frame index, f_t refers to the fast velocity calculated with a frame interval of 2, and s_t represents the slow velocity with an interval of 1.

$$\begin{aligned} f_t &= x[:,t+2,:] - x[:,t,:], \\ s_t &= x[:,t+1,:] - x[:,t,:]. \end{aligned} \tag{9}$$

The following equation represents the construction of bone data, i_{adj} represents the adjacent joint index, and a denotes the angle on the x , y , and z axes, where $w \in \{x, y, z\}$.

$$\begin{aligned} l_i &= x[:,:,i] - x[:,:,i_{adj}], \\ a_{i,w} &= \arccos\left(\frac{l_{i,w}}{\sqrt{l_{i,x}^2 + l_{i,y}^2 + l_{i,z}^2}}\right), \end{aligned} \tag{10}$$

This yields inputs for three branches with a shape of $\times C \times T \times V \times M = 3 \times 6 \times 150 \times 21 \times 1$, where $N = 3$ represents the three branches, C is the number of channels (each branch has two channels, totaling six channels), T denotes the number of frames in the dataset, $V = 21$ indicates the number of joints, and $M = 1$ signifies that there is one hand.

3.4 Training and Inference.

Training. The backbone network uses EfficientGCN, with the output structure adjusted to $N \times T \times C \times V \times M$ for input to the Temporal Action Localization (TAL) module and Classification Head. The model outputs the “rate” and “mid” through the TAL module. We designed the model output as “*mid*” and “*rate*” , where “*mid*” represents the predicted center frame index of the action, and “*rate*” represents the proportion of the predicted action duration relative to the total input frames. This design is inspired by [19]. Unlike the 2D detection problem addressed in that work, our task is a one-dimensional detection problem in the temporal domain. Finally, we compute the predicted start and end frame indices, as shown in the following equation, which facilitates the calculation of IoU.

$$\begin{aligned} f_{\text{pre}}^{\text{start}} &= \text{mid} - \frac{\text{frame}_{\text{gt}} \times \text{rate}_{\text{pre}}}{2}, \\ f_{\text{pre}}^{\text{end}} &= \text{mid} + \frac{\text{frame}_{\text{gt}} \times \text{rate}_{\text{pre}}}{2} \end{aligned} \tag{11}$$

The Intersection over Union (IoU) of the predicted and ground truth segments is then calculated, with the final confidence "$conf$" determined by multiplying IoU and the predicted class probability, as shown in the following equation:

$$\mathrm{conf} \quad = \mathrm{IoU} \times P_{\mathrm{class}} \tag{12}$$

The model is optimized using losses $\mathcal{L}_{\mathrm{class}}$, $\mathcal{L}_{\mathrm{mid}}$, $\mathcal{L}_{\mathrm{rate}}$, and $\mathcal{L}_{\mathrm{conf}}$.

Inference. During inference, the probability of the predicted class is directly used as the value for "$conf$", with the rest of the process following the same steps as in training.

4 Experiment

4.1 Data Collection and Skeleton Extraction

Data Collection and Video Acquisition. We designed 11 types of movements, including reciprocating hand gestures such as "wave hand", "No", and gestures that are opposite in temporal sequences, such as "making a fist", "opening the fist", and "upward" and "downward swipes".

Skeleton Extraction. To facilitate the loading of videos and the precise marking of action boundaries, we created a semi-automatic annotation program. The program requires only the manual selection of the video, after which it is automatically loaded, and frames where the skeleton cannot be extracted are filtered out. Similar to video editing software, users can quickly browse through the actions by dragging the video playback slider. By clicking on the ruler, users can mark the start and end positions of the action. Once confirmed, the program will automatically calculate and output the position of the middle frame of the action and the proportion of the action, significantly improving the efficiency of data annotation.

Since Google Mediapipe is used for extracting 3D skeletons, there may be instances where abnormal skeletons are extracted. To address this, we have designed a skeleton browsing module that allows users to preview the extracted 3D skeletons, ensuring their correctness by dragging the slider.

Data Augmentation. Data augmentation techniques, including Gaussian noise, random scaling, random rotation, and mirror processing, are employed to enhance the robustness of the model. However, for certain actions, such as "left swipe" and "right swipe," mirror processing would alter the nature of the action, causing them to become distinct actions. As a result, mirror processing is not applied to these specific actions. The final dataset includes 3D frame sequences, frames, and gesture localization, with a total of 4,400 samples for the training and validation sets.

Fig. 4 visualizes the data and annotations for three actions.

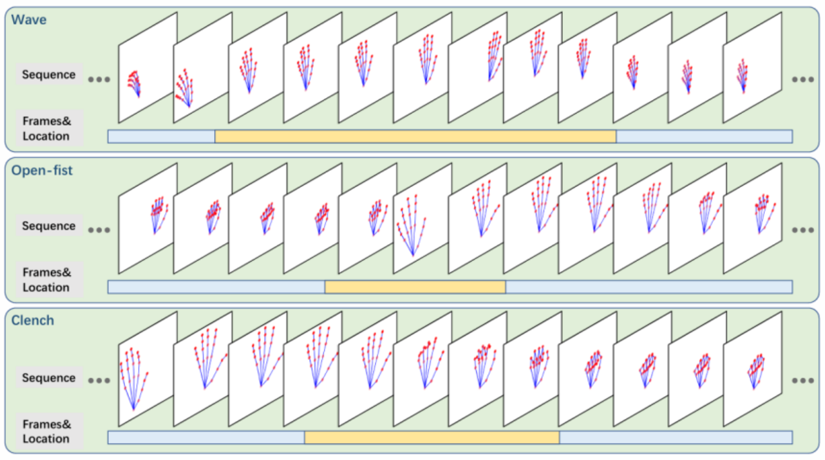

Fig. 4. The figure illustrates three categories from our created dataset: “wave hand,” “open-fist,” and “clench.”The “Sequence” includes the 3D skeletons for all frames, “Frames” represents the total number of frames in the sequence, and “Location” denotes the action localization, which includes mid and rate.

4.2 Comparison of Different Backbone Networks

Since classification and localization are performed in two separate stages, the performance of the chosen backbone network plays a critical role in feature extraction effectiveness. We compared the performance of several GCN-based networks, including accuracy, number of parameters, FLOPs, evaluation time, and IoU metrics, as shown in **Table 1** . The performance of the two newer methods, HD-GCN and GAP, is not ideal, possibly due to the incompatibility in the way the graph is constructed. In our implementation, we uniformly modified the output of each backbone model to have a shape of $N * T * C * V * M$, applied 3D global average pooling in the pyramid module, and implemented the fully connected layer with 3D convolution. After considering all metrics, we selected EfficientGCN as the backbone network.

Table 1. Experimental results of different models serving as the backbone network.

BackBone	Conference	Mean ± Std.	FLOPs	#Param.	Eval Time	IoU
ST-GCN	AAAI2018	87.14	4.38	5.56	4.41s	89.21
MST-GCN	AAAI2021	88.64	2.11	1.59	6.71s	78.91
CTR-GCN	ICCV2021	68.95	2.63	3.91	6.10s	56.99
HD-GCN	ACCV2023	79.05	2.44	3.83	7.37	58.28
GAP	ACCV2023	70.95	2.63	3.91	6.00	59.46
DE-GCN	TIP2024	90.41	1.93	3.77	8.99s	62.83
EfficientGCN	**TPAMI2022**	**94.50**	**2.49**	**1.62**	**5.25s**	**90.29**

4.3 Module Validation and Ablation Study.

The **Fig. 5** shows the confusion matrix of the predicted results for 11 categories. The left subfigure in **Fig. 5** represents the classification results using features output by the pyramid module, while the right one represents the classification results using features extracted by the backbone.It can be observed in left one, category 0 “wave index finger” is easily misclassified as category 1 “wave palm,” and category 3 “double finger merge” is confused with category 4 “double finger spread.” We believe that this may be due to the loss of some spatiotemporal features after feature extraction by the pyramid, leading to a decline in classification performance. In contrast, using the initially extracted features directly yields better results, which may benefit from the robustness of the feature extraction performance of the backbone network.

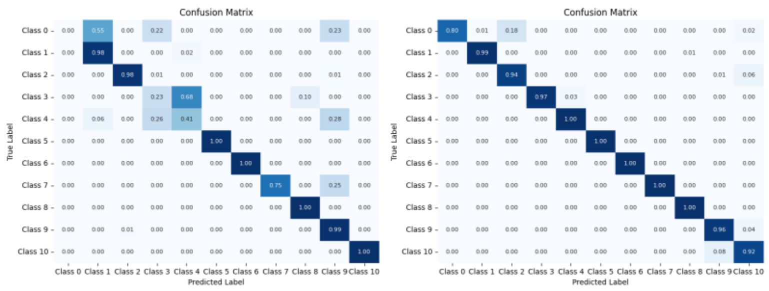

Fig. 5. Confusion matrix of classification results using features output at different stages.

Pyramid Module. Fig. 6 illustrates the confusion matrices of the classification results with and without the pyramid module. After incorporating the pyramid module, the model's performance improves by eliminating misclassifications in category 0 (“wave index finger”) and reducing the confusion between category 3 (“double finger merge”) and category 4 (“double finger spread”). Guided by the localization loss, the model is able to learn more category-specific information. This resulted in an improvement in accuracy, increasing the average Intersection over Union (IoU) from 85.15% to 90.29%.

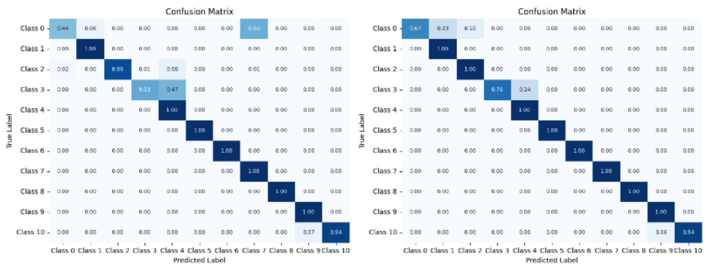

Fig. 6. Confusion matrices of classification results with and without using the pyramid module.

Comparison of Different TAL Methods. Table 2 presents the accuracy results of implementing the TAL layer using different methods. Among them, LSTM achieves the best performance, followed by one-dimensional convolution, with attention-based methods performing the worst. Although LSTM yields the best results, it also consumes more computational resources.

Table 2. Comparison of different TAL methods

TAL Type	Mean ±Std.	FLOPs	#Param.	Eval Time	IoU
MLP	81.95	2.45	1.33	4.54s	81.36
Attention	74.5	2.45	1.33	4.62s	65.54
Conv1d	85.64	2.45	1.33	6.48s	81.3
LSTM	**94.5**	**2.49**	**1.62**	**5.25s**	**90.29**

Comparison of Classification at Different Stages. Table 3 presents the classification results using features from different stages. The experiment shows that the model with the pyramid module achieves the best classification performance when using features output by the backbone network. The model without the pyramid module performs second best, while using features extracted from the pyramid for classification yields the worst results. This is likely due to the loss of certain temporal features during the downsampling process in the pyramid, which further validates the effectiveness of the pyramid module. The results indicate that the pyramid module helps the model learn time-related features, effectively improving classification accuracy.

Table 3. Comparison of Classification at Different Stages

Type	Mean ± Std.	FLOPs	#Param.	Eval Time	IoU
w/o_TAL	89	0.63	0.24	4.24s	85.15
TAL_cls	75.86	2.49	1.62	5.35s	67.14
w_TAL_cls	**94.5**	**2.49**	**1.62**	**5.25s**	**90.29**

Comparison of Fusion at Different Stages. We conducted ablation studies on feature fusion at different stages, with the results shown in **Table 4** . The experimental configurations were divided into two setups: one with four stages and another with three stages. Fusion experiments were performed after the 1st, 2nd, and 3rd stages for both configurations, as well as after the 1st and 2nd stages. The results indicated that the configuration with fusion after the first stage of the three-stage model exhibited the best overall performance.

Table 4. Ablation study on the selection of fusion stages.

Type	Mean ± Std.	FLOPs	#Param.	Eval Time	IoU
after stage1	93.14	2.48	1.62	5.47s	86.32
after stage2	92.95	2.54	1.64	5.76	71.34
after stage3	89.45	2.78	1.73	6.26	78.41
3stages	-	-	-	-	-
after stage1	**94.5**	**2.49**	**1.62**	**5.25s**	**90.29**
after stage2	91.05	2.71	1.7	5.64s	93.09

4.4 Balancing the Loss Function Coefficients

Using grid search, the optimal loss function coefficients were obtained, with λ_{mid} , λ_{rate} , and λ_{conf} set to 0.1, 0.1, and 0.2, respectively. For λ_{class} , after testing values of 0.2, 0.3, 0.5, 0.7, and 1.0, it was found that a setting of 1.0 yielded the highest accuracy. We ultimately adopted this configuration as the final setup.

5 Conclusion

We innovatively developed a 3D skeleton-based temporal hand gesture localization model, using a new loss function to guide the model's learning. To facilitate the training and validation of the model, we col-lected and estab-lished a 3D skeleton dataset containing 11 actions, including action and localiza-tion labels. We conducted effectiveness validation and performance evaluation of the model on this dataset. The results indicate that our model performs effec-tively, demonstrating the validity and practicality of the proposed architecture.

Acknowledgements. This research is supported by the National Natural Science Foundation of China under grant no. 62471266, the Natural Science Foundation of Zhejiang Province under grant no. LZ22F020001,and the 2025 Key Technological Innovation Program of Ningbo City under grant no. 2023Z224

References

1. Fang, Y., Wang, K., Cheng, J., Lu, H.: A real-time hand gesture recognition method. In: Multimedia and Expo, 2007 IEEE International Conference on. IEEE (2007)
2. Li, Y.: Hand gesture recognition using Kinect. In: 2012 IEEE International Conference on Computer Science and Automation Engineering, pp. 196–199. IEEE (2012)
3. Pisharady, P.K., Saerbeck, M.: Recent methods and databases in vision-based hand gesture recognition: A review. Computer Vision and Image Understanding 141, 152–165 (2015)
4. Kılınç Boz, N.Ç., Güdükbay, U.: A hand gesture recognition technique for human-computer interaction. Journal of Visual Communication and Image Representation 28, 97–104 (2015)
5. Molchanov, P., Gupta, S., Kim, K., Kautz, J.: Hand gesture recognition with 3D convolutional neural networks. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE (2015)
6. Oyedotun, O.K., Khashman, A.: Deep learning in vision-based static hand gesture recognition. Neural Computing and Applications 28(12), 3941–3951 (2016)
7. Li, G., Tang, H., Sun, Y., Kong, J., Jiang, G., Jiang, D., Tao, B., Xu, S., Liu, H.: Hand gesture recognition based on convolution neural network. Cluster Computing 22(S2), 2719–2729 (2017)
8. Al-Hammadi, M., Muhammad, G., Abdul, W., Alsulaiman, M., Bencherif, M.A., Mekhtiche, M.A.: Hand gesture recognition for sign language using 3DCNN. IEEE Access 8, 79491–79509 (2020)

9. Gao, Q., Chen, Y., Ju, Z., Liang, Y.: Dynamic hand gesture recognition based on 3D hand pose estimation for human–robot interaction. IEEE Sensors Journal 22(18), 17421–17430 (2022)
10. Yang, L., Peng, H., Zhang, D., Fu, J., Han, J.: Revisiting anchor mechanisms for temporal action localization. IEEE Transactions on Image Processing 29, 8535–8548 (2020)
11. Liu, Y., Wang, L., Wang, Y., Ma, X., Qiao, Y.: FineAction: A fine-grained video dataset for temporal action localization. IEEE Transactions on Image Processing 31, 6937–6950 (2022)
12. Zhai, Y., Wang, L., Tang, W., Zhang, Q., Zheng, N., Doermann, D., Yuan, J., Hua, G.: Adaptive two-stream consensus network for weakly-supervised temporal action localization. IEEE Transactions on Pattern Analysis and Machine Intelligence 45(4), 4136–4151 (2023)
13. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1 (2018)
14. Li, M., Chen, S., Chen, X., Zhang, Y., Wang, Y., Tian, Q.: Actional-structural graph convolutional networks for skeleton-based action recognition. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2019)
15. Li, B., Li, X., Zhang, Z., Wu, F.: Spatio-temporal graph routing for skeleton-based action recognition. Proceedings of the AAAI Conference on Artificial Intelligence 33(01), 8561–8568 (2019)
16. Zhang, X., Xu, C., Tao, D.: Context aware graph convolution for skeleton-based action recognition. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2020)
17. Plizzari, C., Cannici, M., Matteucci, M.: Skeleton-based action recognition via spatial and temporal transformer networks. Computer Vision and Image Understanding 208–209, 103219 (2021)
18. Song, Y.-F., Zhang, Z., Shan, C., Wang, L.: Constructing stronger and faster baselines for skeleton-based action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 45(2), 1474–1488 (2022)
19. Redmon, J.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
20. Liu, J., Shahroudy, A., Xu, D., Kot, A.C., Wang, G.: Skeleton-based action recognition using spatio-temporal LSTM network with trust gates. IEEE Transactions on Pattern Analysis and Machine Intelligence 40(12), 3007–3021 (2018)
21. Cao, C., Lan, C., Zhang, Y., Zeng, W., Lu, H., Zhang, Y.: Skeleton-based action recognition with gated convolutional neural networks. IEEE Transactions on Circuits and Systems for Video Technology 29(11), 3247–3257 (2019)
22. Lai, K., Yanushkevich, S.N.: CNN+RNN depth and skeleton based dynamic hand gesture recognition. In: 2018 24th International Conference on Pattern Recognition (ICPR). IEEE (2018)
23. Zhang, S., Yang, Y., Xiao, J., Liu, X., Yang, Y., Xie, D., Zhuang, Y.: Fusing geometric features for skeleton-based action recognition using multilayer LSTM networks. IEEE Transactions on Multimedia 20(9), 2330–2343 (2018)
24. Tu, J., Liu, M., Liu, H.: Skeleton-based human action recognition using spatial temporal 3D convolutional neural networks. In: 2018 IEEE International Conference on Multimedia and Expo (ICME). IEEE (2018)
25. Zheng, W., Li, L., Zhang, Z., Huang, Y., Wang, L.: Relational network for skeleton-based action recognition. In: 2019 IEEE International Conference on Multimedia and Expo (ICME). IEEE (2019)

26. Duan, H., Zhao, Y., Chen, K., Lin, D., Dai, B.: Revisiting skeleton-based action recognition. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2022)
27. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
28. Lee, J., Lee, M., Lee, D., Lee, S.: Hierarchically decomposed graph convolutional networks for skeleton-based action recognition. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE (2023)
29. Myung, W., Su, N., Xue, J.-H., Wang, G.: DeGCN: Deformable graph convolutional networks for skeleton-based action recognition. IEEE Transactions on Image Processing 33, 2477–2490 (2024)
30. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Skeleton-based action recognition with directed graph neural networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2019)
31. Chen, Z., Li, S., Yang, B., Li, Q., Liu, H.: Multi-scale spatial temporal graph convolutional network for skeleton-based action recognition. Proceedings of the AAAI Conference on Artificial Intelligence 35(2), 1113–1122 (2021)
32. Chen, Y., Zhang, Z., Yuan, C., Li, B., Deng, Y., Hu, W.: Channel-wise topology refinement graph convolution for skeleton-based action recognition. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE (2021)
33. Zhao, Y., Xiong, Y., Wang, L., Wu, Z., Tang, X., Lin, D.: Temporal action detection with structured segment networks. In: 2017 IEEE International Conference on Computer Vision (ICCV). IEEE (2017)
34. Lin, C., Xu, C., Luo, D., Wang, Y., Tai, Y., Wang, C., Li, J., Huang, F., Fu, Y.: Learning salient boundary feature for anchor-free temporal action localization. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2021)
35. Liu, X., Bai, S., Bai, X.: An empirical study of end-to-end temporal action detection. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2022)
36. Shi, D., Zhong, Y., Cao, Q., Ma, L., Lit, J., Tao, D.: TriDet: Temporal action detection with relative boundary modeling. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2023)
37. Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2017)