The Journal of Supercomputing (2024) 80:17952–17979
https://doi.org/10.1007/s11227-024-06138-1

LGAFormer: transformer with local and global attention for action detection

Haiping Zhang[1,2] · Fuxing Zhou[2] · Dongjing Wang[1] · Xinhao Zhang[2] · Dongjin Yu[1] · Liming Guan[3]

Accepted: 9 April 2024 / Published online: 6 May 2024

Abstract

Temporal action detection is a very important task in video understanding, aiming at predicting the start and end time boundaries of all action instances in an unedited video and their action classification. This task has been widely studied, especially after the transformer has been widely used in the field of vision. However, transformer model brings massive computational resource consumption when processing long sequence input data. At the same time, due to the ambiguity of the video action boundary, many proposal and instance-based methods cannot predict the video boundary accurately. Based on the above two points, we propose LGAFormer: a very concise model that combines the local self-attention mechanism with the global self-attention mechanism, using local self-attention in the shallow layer of the network to process short-range temporal data to model local representations while reducing a large amount of computational consumption, and using global self-attention in the deep layer of the network to model long-range temporal context. This allows our model to achieve a good balance between effectiveness and efficiency. And in terms of detection head, we combine the advantages of the segment feature and instance feature to predict action boundaries more accurately. Thanks to these two points, our method achieves comparable results on all three datasets (THUMOS14, ActivityNet 1.3, and EPIC-Kitchens 100). On THUMOS14, an average mAP of 67.7% was obtained. On ActivityNet 1.3, the best performance is obtained with an average mAP of 36.6%. On EPIC-Kitchens 100, an average mAP of 24.6% was achieved.

Keywords Action detection · Transformer · Self-attention · Video understanding

Haiping Zhang, Dongjing Wang, Xinhao Zhang, Dongjin Yu and Liming Guan have contributed equally to this work.

Extended author information available on the last page of the article

1 Introduction

Temporal action detection is to detect the temporal boundaries of each action instance in an unedited video, i.e., the start and end moments of the action, and to classify the action instance, which is an important and very challenging task in video understanding because it requires a model with strong spatiotemporal feature extraction capability.

The mainstream approaches for this task can be broadly classified into three categories: first, convolution-based approaches [1–5]. Convolutional neural networks(CNNs) have played a major role in vision for many years. This is also true in the video understanding task. 3D convolutional networks can naturally handle 3D data like video, so many 3D convolutional networks [6, 7] have made great breakthroughs in this task. However, such 3D networks tend to have a very large number of parameters, while the limited perceptive field of convolutional networks makes these networks unable to capture long-term spatiotemporal dependencies. Therefore, in today's increasingly diverse video landscape, pure convolutional network models are increasingly limited. The second category is the transformer-based approaches [8–12]. The transformer [13] originally used in NLP have been gradually applied to the vision domain [14–18] and have achieved significant results (e.g., classification, detection, etc.). Since the transformer model is more suitable for processing sequential information compared to the convolutional model, the transformer model has been naturally extended to the video understanding domain, producing quite a lot of excellent work [19–23]. However, such methods have a huge drawback in that they consume very high computational resources when the input sequence of the model is too long and therefore have high requirements for the equipment. So a third class of approaches [24–27] also emerged: combining CNN with transformer. By combining convolution, or MLP, with transformer, the computational complexity is reduced. However, the overall design of these models is usually very complex, adding many self-designed extra modules to the original transformer structure, making the whole model quite bloated.

We chose the second one for the above three types and constructed a clean and pure transformer model. One of the key modules in the transformer model is the self-attention mechanism. In recent years, research in various fields has confirmed the powerful spatiotemporal representation learning capabilities of the self-attention mechanism. For instance, in the medical imaging domain, RadFormer [28] achieves higher accuracy in gallbladder cancer detection by combining global attention and local attention. In the field of action recognition, DANet [29] use the dual-attention to learn robust video representations for view-invariant action recognition. In addition, there are many transformer-based models [9, 30] that solely utilize global self-attention mechanisms to model sequence features. However, from previous studies on CNN networks and transformer networks [12, 31–33], the shallow layer of the network tends to capture more local information in the image. In contrast, the deep layer tends to capture more high-level semantic information. Therefore, using global self-attention in the shallow layer will

focus on unnecessary distant information, resulting in computational redundancy and consuming huge computational resources when the global self-attention layer processes long input sequences at once. To reduce the computational complexity, some transformer-based models [34] replace the global self-attention with local self-attention all together to restrict the self-attention to a small local window. However, this will limit the deep layer of the network to learn high-level semantics and model long-range dependencies. As shown in Fig. 1, a transformer model that uses only local self-attention significantly differs in detection accuracy when detecting actions with a smaller amplitude and actions with a larger amplitude. Detection accuracy is much higher for short-duration actions such as diving and shot put throwing than for long-duration actions such as playing tennis and gymnastic sports. This is because local self-attention cannot capture long-term dependencies well for actions with larger amplitude and longer duration.

According to the characteristics of global self-attention and local self-attention, in this paper, we construct a transformer-based model with a changeable self-attention mechanism: LGAFormer. In the shallow layer, the network focuses only on the

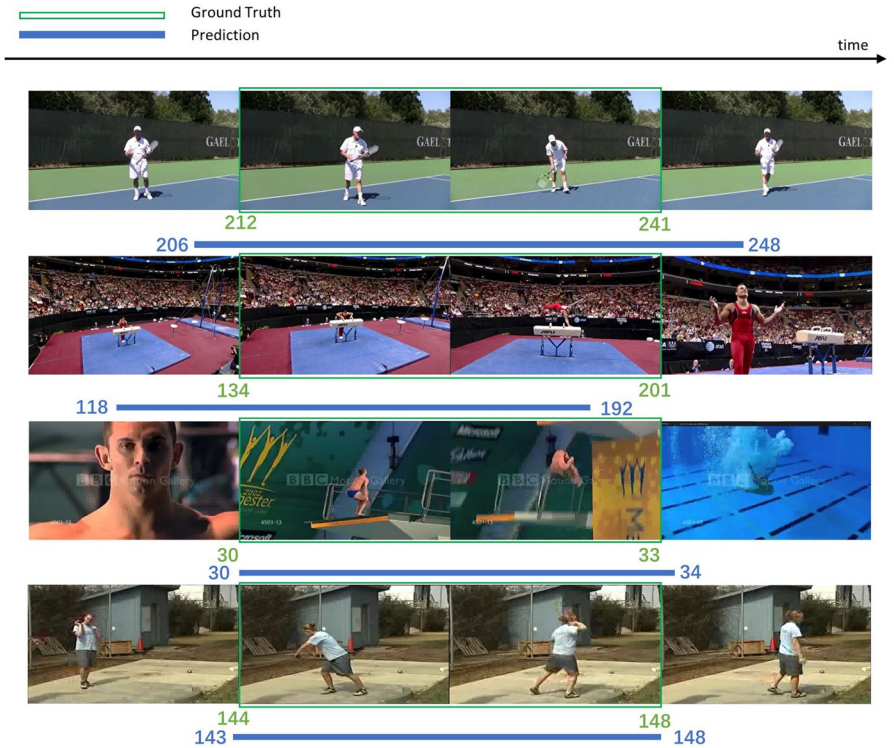

Fig. 1 Detection results of TAD method based on local self-attention. Large difference in effect between long-duration and short-duration action detection. Detection accuracy is much higher for short-duration actions(bottom2) such as diving and shot put throwing than for long-duration actions(top2) such as playing tennis and gymnastic sports

changes in adjacent frames and the local style information within the image, so we use the Local Attention Transformer Module (LATM) layer to capture the pattern information within the local window, and in the deep layer, the Global Attention Transformer Module (GATM) model the long-term spatiotemporal dependencies. This ensures that the LGAFormer can model both short-term local spatiotemporal information and long-term global spatiotemporal information. By constructing such features into a feature pyramid structure, with a detection head module attached to each layer of the feature pyramid, the output features can be decoded into class and temporal boundaries for each action instance.

In addition, most of the current detection heads are divided into two categories: one is the anchor-based approach [35, 36], which divides the output feature pyramid according to the anchor to get a candidate segment of a specific size, and then corrects the boundaries of the candidate segments by the regression head to get the final prediction results. However, the boundary of the action in the video is often very ambiguous, and this regression method, based on the global features of the whole clip, cannot capture the detailed information of the boundary. Another type of method is the anchor-free method [34, 37–39], which generates temporal proposals by predicting the offsets between each moment and the start or end boundaries. However, this method does not consider the relative relationship between adjacent moments, so the confidence of the obtained boundaries is not high. Therefore, our detection head combines the above two methods, first generating a time segment with a fixed time length at each moment, and predicting the offset between each moment of this time segment and the boundary, then estimating the relative probability distribution of the boundary between all moments of this segment, finally obtaining the temporal boundary by means of expectation.

In summary, our main contributions are three:

1. We constructed a pure transformer network for temporal action detection tasks, which does not rely on predefined anchor boxes or other complex designs. This approach aims to achieve a better balance between efficiency and effectiveness in the model.
2. We design a stage-changeable self-attention mechanism in the transformer structure, using LATM at shallow levels to capture local features while reducing computational redundancy and GATM at deep levels to model long-term dependency information. Moreover, the detection head is carefully designed to combine segment and momentary features to predict action boundaries more accurately.
3. Finally, we conducted a series of comparative tests on a standard test set, and we achieved good results in testing results while keeping computational resources within manageable limits.

2 Related work

2.1 Temporal action detection

Action recognition and temporal action detection are two important tasks in video understanding, where action recognition is the classification of edited videos into

actions, and temporal action detection is to locate the start and end moments of each action instance in an unedited video and then classify them.

Most of the current TAD methods use the action recognition model as the video encoder, input the video into the action recognition model first to get the initial video spatiotemporal features, then use these features as the input to the TAD model, further process the features and decode them to get the final output. Some of the more used video encoders are TSN [40], I3D [41], SlowFast [42], etc. TSN is based on the idea of long-range temporal structure modeling. It combines a sparse temporal sampling strategy and video-level supervision to enable efficient and effective learning using the whole action video. I3D, a very widely used 3D CNN model, inflates the inception network directly into a 3D network, which can be pretrained using the dataset of an otherwise 2D network. SlowFast network is a two-stream network divided into a slow path and a fast path to extract spatiotemporal information. Both use the classical 3D CNN network. In order to save computational cost, our method LGAFormer is also constructed on video features encoded with a pre-trained video classification network.

As for the TAD model, which is only a detector, most of the current methods can be divided into three categories: the multistage, two-stage, and single-stage methods. The multistage method [43–45] first generates a number of candidate segments and then classifies each segment as action foreground or background by generating a confidence score through a binary classifier and further feeds the candidate boxes with high confidence scores to the multi-category classifier for action classification, and refines the boundaries of the candidate boxes by a regressor. The two-stage methods [46–50] generate candidate video segments as action proposals in one step and then further classify these proposals into specific action categories and modify their timing boundaries. Both methods generate proposals first and then perform classification and regression correction. In generating proposals, some manually predefined fixed-size initial segment is usually used, which significantly limits the flexibility of the model, and this multistage model cannot be trained end-to-end. The one-stage methods [51–53] detect both the boundaries and categories of action segments in a single shot. Most of them are anchor-based methods, which generate a predefined number of fixed-size anchor boxes by anchor points, and then classify and regress these anchor boxes to get the boundaries and classes of action instances. Our approach also borrows this idea by generating a fixed-size segment for each prediction point and combining the features of the segment with the moment features of the prediction points to help precisely locate the temporal boundaries.

2.2 Anchor-free action detection

Since the anchor-based approach requires setting a fixed size and number of anchor frames to generate action candidates, which will affect the generalization ability of the model to some extent, the anchor-free action detection approach emerged. Before this, there were many anchor-free methods [54–56] in the object detection task, the most famous of which is the YOLO [57] directly used to predict coordinates of bounding boxes from raw images. Fcos [52] aims to learn the distance to boundaries of each spatial location and utilizes feature pyramid for objects with diverse scales. Inspired by

these methods, many anchor-free methods have emerged for action detection task, such as A2Net [36], which directly predicts the distance to the action boundaries and the action category for each frame. In addition, AFSD [37] adds a saliency-based refinement module to the convolutional network to extract boundary saliency features to optimize the boundary position of each proposal. Our method belongs to the category of anchor-free, by matching scores of start boundary and end boundary for each moment and also predicting the offset between each moment to the boundary. Multiple moments are combined into one segment, and the relationship between each instance in the segment finally derives the most accurate boundary position.

2.3 Video transformer

Due to the great success of transformer [13] in the field of NLP, the ViT model [58] first applied transformer to the field of vision, which immediately attracted the attention of many researchers. Subsequently, various transformer-based methods have made their mark in the image field with quite good results [14–16]. At the same time, this series of work has also promoted the development of transformer-based model in the video field [59, 60]. The core of transformer lies in its self-attention mechanism, which can easily capture the long-range context compared to CNN networks. Therefore, a large number of transformer-based approaches have emerged in video understanding. In the action recognition task, ViViT [61] and TimeSformer [62] propose to factorize along spatial and temporal dimensions on the granularity of encoder, attention block, or dot-product computation. Since global self-attention requires significant computational resources, the Video Swin transformer [22] developed from Swin Transfomer [17] dramatically reduces the cost of the transformer by introducing local self-attention.

On the other hand, many transformer-based methods have emerged for action detection task, among which TadTr [30] detects the action with the DETR-like transformer-based decoder. To reduce costs, ActionFormer [34] uses local self-attention to build a pure and straightforward transformer model with excellent results. E2E-TAD [39] explores the specific benefits of an end-to-end training approach based on the transformer model.

As most of the previous methods are based on the fixed self-attention mechanism, we have constructed a straightforward model by summarizing previous research experience with the changing self-attention between network layers, choosing the corresponding global self-attention and local self-attention according to the characteristics of different network layers processing features, saving computational resources while maximizing the ability to capture contextual information with the self-attention mechanism.

3 Method

3.1 Problem formulation

Following common video action detection methods [63, 64], we consider feature sequences extracted from video frames by a 3D CNN as input to LGAFormer. Each

video V_i is divided into T clips, and accordingly, there will be T feature clips corresponding to each video clip. In this way, the input feature sequence for the pipeline can be written as $X = \{x_1, x_2, \ldots, x_T\}$, T is not a fixed value. Each video has a different size of T depending on its video length. Furthermore, for each video sequence, there is a set of labels with number N relative to it: $K = \{k_n = (t_{s,n}, t_{e,n}, C_n)\}_{n=1}^{N}$, where k_n represents the n-th action instance, and $t_{s,n}$, $t_{e,n}$ and C_n are its start time, end time and action class, respectively.

The goal of TAD is to predict M possible action instances $\Lambda = \{\lambda_m = (\bar{t}_{s,m}, \bar{t}_{e,m}, \bar{C}_m, P_m)\}_{m=1}^{M}$ based on the input X. Here, λ_m represents the m-th predicted action in the video, it contains four indicators $\bar{t}_{s,m}$, $\bar{t}_{e,m}$, $\bar{C}_m$ and P_m. $\bar{t}_{s,m}$ and $\bar{t}_{e,m}$ represent the predicted start time and end time for the m-th predicted action; $\bar{C}_m$ and P_m are its predicted action class and confidence score, respectively.

3.2 Overall architecture

The overall architecture of LGAFormer is illustrated in Fig. 2, which can be divided into four main parts, data preprocessing, backbone, feature pyramid neck, and detection head.

Firstly, in the data preprocessing part, due to the limited memory in GPUs, we use the pre-extracted features $X = \{x_1, x_2, \ldots, x_T\}$ by the pre-trained network as the input of the network. We will select different pre-trained networks according to the characteristics of different datasets and then project the obtained pre-extracted features into embedded features by the Projection layer. For the design of the Projection layer, we use a simple convolutional network with RELU as the activation function, following [34]. This can be expressed in the form of

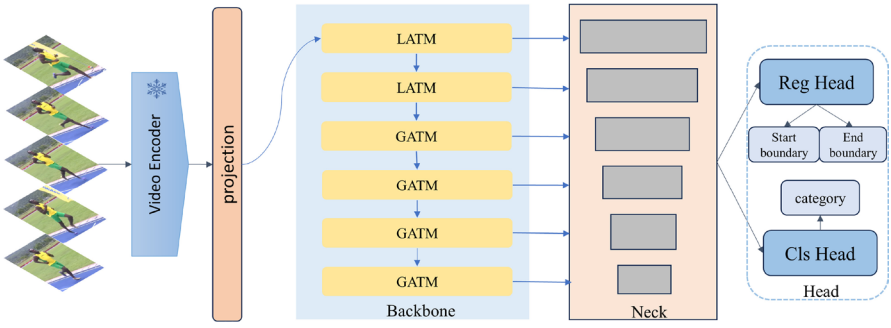

Fig. 2 Overview of LGAFormer architecture. The overall structure can be divided into four parts: data preprocessing, backbone, neck, and head. First, a sequence of video clip features are extracted using a pretrained model in the data preprocessing phase and mapped into feature embeddings. These feature embeddings are further processed by the backbone network (consisting of LATM and GATM) to obtain the feature pyramid. Finally, each layer of the feature pyramid is input to the detection head, the classification head obtains the classification result, and the start and end boundaries are obtained by combining the moment feature and segment feature by the regression head. Finally, the detected action instances are output

$$L_0 = ReLu(Conv_2(ReLu(Conv_1(X)))) \tag{1}$$

Here L_0 is used as the input features of the feature pyramid network (FPN), and due to the nature of CNN networks, adding the convolution net in the shallow layer of the network can aggregate the local features well.

Then comes the backbone part, the transformer network in the backbone uses L_0 as input. The backbone structure can be divided into two parts; the first part is the Local Attention Transformer Module(LATM), which consists of L transformer blocks with local self-attention mechanism [65]. The role of LATM is to encode local spatiotemporal representations in the shallow layers of the network. Followed by Global Attention Transformer Module (GATM), it use global self-attention mechanism to tackle long-term dependencies in the deeper layer.

The third component in our modular TAD pipeline is the neck. The neck module plays an essential role in the overall TAD task in the detection pipeline as a component after the backbone and before the detection head. The neck's purpose is to align video features and downstream detection tasks. It provides a flexible mechanism to handle significant variations in the duration of action instances by constructing a multi-resolution representation.

The last component in our modular TAD framework is the detection header. Its goal is to accomplish the detection task of generating action instances and their labeled time intervals. It usually consists of sub-networks for classification and regression tasks, respectively. Also, specific sample allocation is vital for the training of these detection heads. Well-designed sub-networks and a reasonable sample allocation mechanism complement the final detection performance. In later sections, we look in detail at the design of the backbone network and the detection heads.

3.3 Backbone design

The backbone part of the network is composed of a pure transformer structure. In our design, we divide the backbone network into two parts, LATM and GATM, both of which are complete transformer network modules as shown in the Fig. 3, where there are two key sub-layers, the first is a multi-head self-attention mechanism and the second is a simple, position-wise fully connected feed-forward network (MLP). A residual connection exists around each of the two sub-layers, followed by layer normalization. Finally, there is a downsampling operation at the end of each layer as a way to build the FPN. In our model, the maximum pooling layer does the downsampling operation here. The overall process can be expressed by:

$$\begin{aligned} \hat{L}_n &= MSA(LN(L_{n-1})) + L_{n-1}, \\ \bar{L}_n &= MLP(LN(\hat{L}_n)) + \hat{L}_n, \\ L_n &= Downsample(\bar{L}_n), \quad n = 1 \ldots N \end{aligned} \tag{2}$$

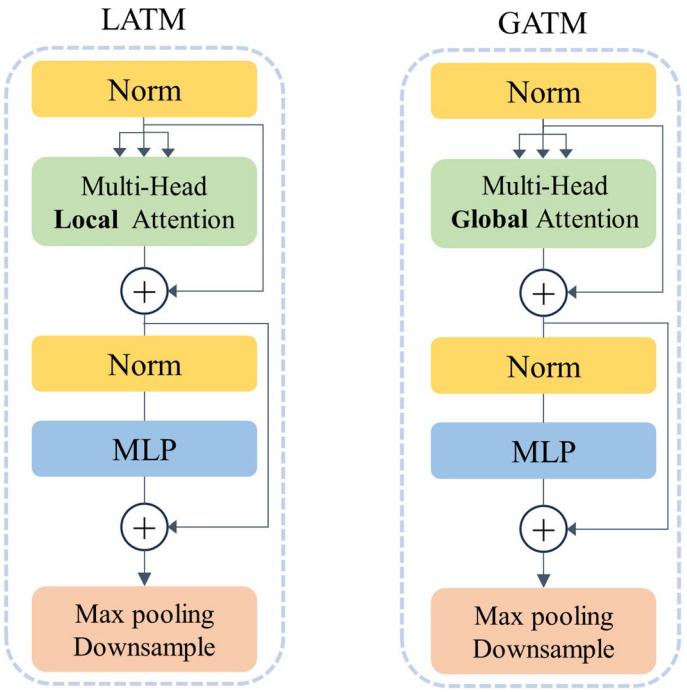

Fig. 3 Illustration of the structure of the LATM and GATM. The LATM and GATM are both standard transformer structures, with the most crucial component being the self-attention mechanism. The primary difference between LATM and GATM lies in the self-attention mechanism employed. LATM utilizes a local multi-head self-attention mechanism, whereas GATM employs a global multi-head self-attention mechanism

Where $L_{n-1}, \hat{L}_n$ and $\bar{L}_n \in \mathbb{R}^{T^{n-1} \times D}$, $L_n \in \mathbb{R}^{T^n \times D}$. And $MSA()$ represents the multi-head self-attention mechanism, while $LN()$ denotes layer normalization operations. Then after passing through N layers in this way, a feature pyramid with different time-scale feature layers can be constructed, and each layer of the FPN is input to the detection head separately for detection, which facilitates the detection of actions of different durations.

3.3.1 Local attention transformer module

As proposed in the Sect. 1, the use of global self-attention in the shallow part of the network causes a large amount of computational consumption due to the generally long input sequence at the beginning of the network. Moreover, it is known from previous experience that self-attention only focuses on local contextual information at the beginning of the network, so we use LATM as the backbone in the front part of the network, and the core part of LATM is local self-attention. And its computation process can be described as the input $L_0 = \{l_i\}_{i=1}^{T} \in \mathbb{R}^{T \times D}$, for each head $j \in \{1, \ldots, H\}$ of multi-head self-attention, l_i is projected into

$Q_{ji} = W_{ji}^{Q} l_i$, $K_{ji} = W_{ji}^{K} l_i$ and $V_{ji} = W_{ji}^{V} l_i$, where W_{ji}^{Q},W_{ji}^{K} and $W_{ji}^{V} \in \mathbb{R}^{D_h \times D}$ represent the weights of linear layers, $D_h = D/H$, represents the feature dimension of each head. The output of the j-th head self-attention is given by:

$$A_{ji} = Softmax\left(\frac{Q_{ji}K_{ji}^{\mathrm{T}}}{\sqrt{D_h}}\right) \times V_{ji} \tag{3}$$

where $A_{ji} \in \mathbb{R}^{T \times D}$ is A weighted average of the features. Then, the combination of multi-head self-attention can be shown as:

$$P_i = W_i^{out} Concat(A_{1i}, \ldots, A_{ji}) + L_n \tag{4}$$

The output of multiple headers is contacted and then passed through a linear layer, where $W_i^{out} \in \mathbb{R}^{D \times D}$ denotes the weight of the linear layer. From the above equation, we can calculate that single-head self-attention has a complexity of $O(T^2D)$, so to reduce the computational complexity, we replace global self-attention with local self-attention in the shallow network, which forces each query token only attend to tokens within the same local window. This has two advantages: on the one hand, it forces attention in the local window, which makes the network pay more attention to local patterns in the early stage, and thus aggregates local spatiotemporal context relations; on the other hand, it can reduce the computational complexity to $O(W^2TD)$, and since the input dimension T of shallow networks is usually large, and the manually set self-attention window size W will be much smaller than T, which can reduce most of the computational consumption. The model significantly alleviates computation redundancy by applying LATM in the early stages and efficiently encodes local spatiotemporal representations.

3.3.2 Global attention transformer module

In order to capture long-term dependencies, we use GATM at the deeper layers. The only difference between GATM and LATM is the type of self-attention mechanism, in which GATM uses global self-attention. Due to the FPN structure, the length of the input features is already much reduced in the last stage of the network than in the initial stage, and the computational resources consumed by GATM at this time are correspondingly much reduced. In GATM, given a query token, GATM compares it with all the tokens used for aggregation. In this way, we ensure that the model captures global dependencies in the final stage. By combining LATM in the shallow layer, the model forms an efficient way to learn the spatiotemporal representation of action detection.

3.3.3 Feature pyramid network

Like object detection, a critical structure in the temporal action detection model is the FPN because in both target detection and action detection, the size of the detection target is not fixed, enabling the model to detect action instances of different

lengths. We construct feature pyramid with different temporal lengths and then feed the feature outputs of each layer into the detection head for action detection; the specific structure is shown in Fig. 4. The construction method of feature pyramid in our model is straightforward; to ensure model simplicity and avoid introducing additional parameters, we chose not to adopt the complex PFN design used in the past. Instead, we implemented downsampling of feature maps by adding a max-pooling operation at the end of each transformer layer. The downsampled features are then fed into the next transformer layer, allowing for the extraction of feature maps with varying temporal dimensions through multiple layers of processing. The reason we chose the max-pooling layer as the downsampling method is that the feature changes between successive frames of the video in motion detection are very subtle, and the video clips exhibit a high redundancy which leads to the high similarity of the pre-extracted features. Therefore, Max Pooling is deemed the most fitting block for preserving the most discriminative features. Finally, we construct a FPN structure with 6 layers through the maximum pooling layer, and the downsampling ratio between successive feature maps is set to 2.

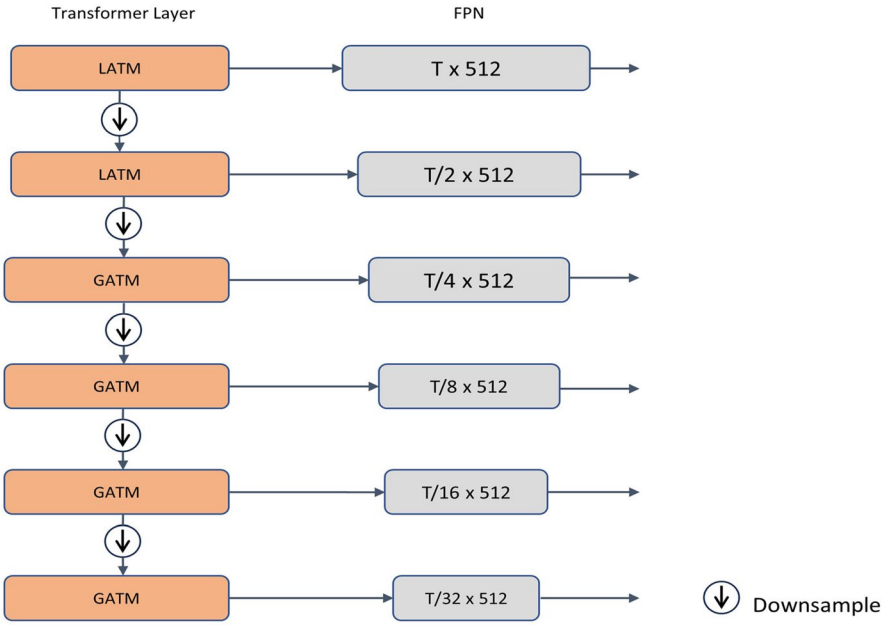

Fig. 4 Illustration of the structure of the feature pyramid. There are seven transformer blocks, with six layers of output used to build the feature pyramid. The output of each transformer block is downsampled to obtain the pyramid features of each layer. The maxpooling layer does the downsampling operation, and the downsampling rate of each layer is different. The final feature pyramid is composed of feature vectors of different scales

3.4 Classifier and regressor

After obtaining the temporal feature pyramid L_n, an anchor-free prediction module is utilized to predict the boundary distances and class scores at each location t on l_n. The prediction module contains a classification module and a regression module.

3.4.1 Classifier

Formally, the classify head is defined as:

$$C_n = Con_{1d}(E_2(E_1(L_n))) \tag{5}$$

Here, L_n is the latent feature of level n, $C_n = \{c_0, c_{2^{l-1}}, \ldots, c_T\} \in \mathbb{R}^{\frac{T}{2^{n-1}} \times D}$ denotes the classification probability with $c_i \in \mathbb{R}^D$. E denotes 1D convolution, followed by layer normalization and ReLU activation function. Note that all weights of the decoder are shared among different features in the multi-scale feature pyramid L. A sigmoid function is attached to each output dimension to predict the probability of C action categories.

3.4.2 Regressor

In order to locate the boundary position more precisely, we propose a regress head that combines the instance feature and segment feature to capture the position information of each moment while modeling the relative relationship between multiple neighboring moments to determine the final boundary position. The procedure is shown in Fig. 5: the feature of each FPN layer is input to the detection head. Given the feature of the n-th layer as L_n, it is first input to the boundary regressor to obtain the location feature $f_{loc} \in \mathbb{R}^T$ that represents the response value at the current moment as a boundary point, which can be expressed as

$$f_{loc} = ReLu(Con_{1d}(LN((Con_{1d}(LN(Con_{1d}(L_i))))))) \tag{6}$$

For a moment t, when moment t is located in action duration, S moments adjacent to t are selected, and the left adjacent moments (i.e., moments close to the action start boundary) are selected to form a segment at the prediction start boundary, denoted $Seg[t-s, \ldots, t]$, and accordingly, S right adjacent moments are also selected to form $Seg[t, \ldots, t+s]$ at the prediction end boundary, so that each segment contains s+1 moments. This segment feature is then fed into the offset Regressor to obtain the offset $f_{os} \in \mathbb{R}^{S+1}$ for each moment to the boundary within the segment. The Regressor is again implemented using a 1D convolutional network, following the same design as the classification network, with the difference that the ReLU connection is used at the end for distance estimation. We then sum the outputs of these two regressors, after a softmax function, to obtain the probability distribution P_{tb} for each moment inside the segment as the action start boundary, which can be expressed as

$$P_{tb} = Softmax(f_{loc}^{(t-s):t} + f_{os}) \tag{7}$$

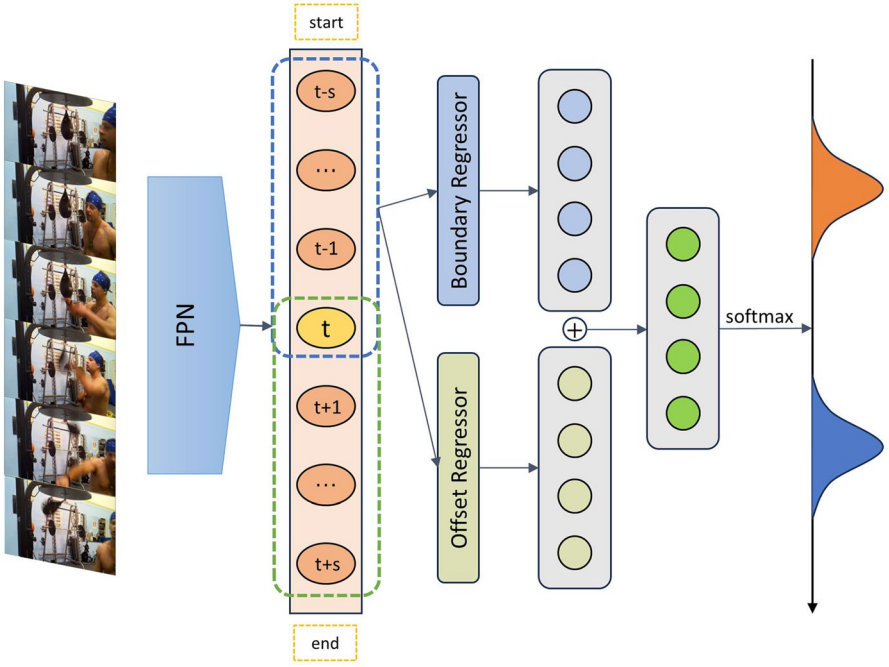

Fig. 5 The process of boundary localization in regressor. The corresponding segment feature for the start and end boundary is obtained at each moment of the feature pyramid. Then, each feature in the segment is input into the boundary regressor and offset regressor to derive the location feature and offset feature, respectively. These two features are added together to get the boundary feature for each moment in the segment, and then the softmax function is used to obtain the boundary probability distribution in the segment. Finally, the expectation value generates the boundary position with the highest confidence

Then the expectation of distance between the moment t and boundary can be derived from the distance between each moment in the segment and moment t, which is the final predicted distance between t and the boundary d_{tb}, and can be expressed by

$$d_{tb} = \sum_{s=1}^{S} (s \times P_{tb}^{s}) \tag{8}$$

Next we can get start and end time for tth time step in i-th level as follow

$$\psi_t = 2^{l-1} \times (t - d_{tb}) \tag{9}$$

$$\xi_t = 2^{l-1} \times (t + d_{tb}) \tag{10}$$

By the above method, the feature pyramid of each layer can be decoded into action instances with time boundary (ψ_t, ξ_t) and action category c_t.

3.5 Training and inference

3.5.1 Training

After the output features of each FPN layer are input to the detection head, there is an output for each moment t of each feature pyramid layer, defined as $o_t^l = (p(a_t^l), d_{st}^l, d_{et}^l)$. Where $p(a_t^l)$ denotes the probability of action categories, d_{st}^l and d_{et}^l denote the distance between the instance t and the start boundary and end boundary, respectively. The Focal Loss [66] and IoU loss [67] are employed to supervise classification and regression outputs, respectively. The overall loss function consists of positive loss and negative loss

$$L_{total} = L_{pos} + L_{neg} \tag{11}$$

where L_{pos} is the loss function for positive samples, and accordingly, L_{neg} is the loss function for negative samples. They can be written as

$$L_{pos} = \frac{1}{N_{pos}} \sum_{n,t} 1\circ\{c_t^n > 0\}(\tau_{IoU} L_{cls} + L_{reg}) \tag{12}$$

$$L_{neg} = \frac{1}{N_{neg}} \sum_{n,t} 1\circ\{c_t^n = 0\} L_{cls} \tag{13}$$

Here, N_{pos}, N_{neg} is the number of positive and negative samples. Following previous methods [68, 69], center sampling is adopted to determine the positive samples. Namely, the instants around the center of an action instance are labeled as positive, and all the others are considered as negative. $1\circ\{\}$ is a indicator function.τ_{IoU} is the temporal Interaction of Union(tIoU) between predicted segment and the ground truth action instance, L_{cls} and L_{reg} is focal loss and IoU loss.

3.5.2 Inference

For inference, our model takes the input video X and outputs $(p(a_t), d_t^s, d_t^e)$ for each time step t at all pyramid levels. Each output further decodes to action instancd $a = \{p(a_t), s_t = t - d_t^s, e_t = t + d_t^e\}$. $p(a_t)$ is the action confidence score, and s_t, e_t is the start and end of the action, respectively. The action candidates are further processed using soft NMS [70] to remove highly overlapping instances to produce the final output of the action.

4 Experiment

4.1 Datasets and metrics

We perform extensive experiments on the datasets of THUMOS14 [71], ActivitiyNet1.3 [72], and EPIC-Kitchens 100 [73] to demonstrate the effectiveness of

our LGAFormer. We employ a widely-used evaluation metric for TAL, the mean average precision (mAP) calculated at various temporal intersections over union (tIoU). We report the mAP scores for all action categories based on the given tIoU thresholds and further report an averaged mAP value across all tIoU thresholds. For THUMOS14 and EPIC-Kitchens 100, we report the tIoU thresholds at [0.3:0.7:0.1] and [0.1:0.5:0.1], respectively. For ActivityNet, we report the result at tIoU threshold [0.5, 0.75, 0.95], and the average mAP is computed at [0.5:0.95:0.05].

4.1.1 THUMOS14

The dataset for temporal action detection tasks comprises a total of 413 untrimmed videos, annotated with temporal labels for 20 action categories. Among these, 213 videos are designated for validation, while 200 videos are designated for testing. On average, each video contains 15 action instances, with an average duration of 4.04 s per action instance. We utilize the validation set for model training and the test set for evaluating the final detection performance.

4.1.2 ActivityNet1.3

ActivityNet is a large-scale dataset containing 10,024 training videos, 4926 validation videos, and 5,044 test videos belonging to 200 activities covering sports, household, and working actions. ActivitiesNet1.3 only contains 1.5 action instances per video on average, and most videos contain a single action category with 36% background on average.

4.1.3 EPIC-Kitchens 100

The EPIC Kitchens 100 dataset is a comprehensive collection of self-centered action videos that includes 100 h of footage from 700 sessions documenting a variety of kitchen cooking activities. Furthermore, compared to THUMOS14, EPI-Kitchens 100 is three times larger in total video hours and more than ten times larger in action instances (128 action instances per video on average). The videos were recorded from a first-person perspective, generating significant camera movement, which was a new challenge for Good Future's research.

4.2 Implementation details

Our model follows an end-to-end training approach for all datasets, and we use Adam [74] with warm-up for training. First, we limit the maximum input length to 2309 for data processing, and for too long or too short input sequences, we crop or pad accordingly. The initial learning rate is set to 10^{-4} for THUMOS14 and EPIC-Kitchens 100, and $10^{-3}$for ActivityNet1.3. In the proposed network, the total number of transformer block is 7, the number of LATM in our model is (6, 2, 2) and the number of GATM is (1, 5, 5) for THUMOS14, ActivityNet1.3 and EPIC-Kitchens

100. In addition, the local window size of local self-attention in LATM is 19, 7, and 9 for THUMOS14, ActivityNet1.3 and EPIC-Kitchens 100, respectively. Regarding FPN structure, the number of layers of FPN is 6 and the downsampling ratio between each layer is set to 2. In the detection head the segment step S is set to 16 for Thumos14 and EPIC and 14 for ActivityNet. Finally, We implemented and compiled our framework using PyTorch 1.11, Python3.8, and CUDA 11.3 on a single Nvidia Geforce RTX4090.

4.3 Main results

In this subsection, we compare LGAFormer with state-of-the-art action detection methods on THUMOS14, ActivitiyNet1.3, and EPIC-Kitchens 100 in Tables 1, 2, and 3.

On the THUMOS14 dataset, we divide these methods into CNN-based networks and transformer-based networks to report results, and also report the backbone used by each method, e.g., I3D [83], TSN [40], R(2+1)D [84], Video Swin Transformer [22]. Our method achieves an average mAP of 67.7% ([0.3: 0.1: 0.7]), with an mAP of 82.4% at tIoU=0.5 and an mAP of 45.2% at tIoU = 0.7, reaching the state of the art. Especially compared to CNN-based methods, our results are significantly better than these methods. And from the experimental results, our model is more advantageous when tIoU is 0.7. This is due to the design of our detection head, which can predict the action boundary more accurately.

On ActivityNet1.3, we used pre-trained methods from TSP [85], and the results are shown in Table 2, and our results are still comparable. Our model outperforms most transformer-based methods and is second only to TCANet [11]. From the

Table 1 Result on THUMOS14. We report results on two types of methods, the CNN-based method and the transformer-based method, and report the pre-trained features used by each method, where bolded text indicates the best results. Our LGAFormer achieves the SOTA level

Type	Model	Feature	0.3	0.4	0.5	0.6	0.7	Avg.
CNN	BSN [75]	TSN	53.5	45.0	36.9	28.4	20.0	36.8
	BMN [76]	TSN	56.0	47.4	38.8	29.7	20.5	38.5
	DCAN [38]	TSN	68.2	62.7	54.1	43.9	32.6	52.3
	AFSD [37]	I3D	67.3	62.4	55.5	43.7	31.1	52.0
	BasicTad [77]	R50-SlowOnly	75.5	70.8	63.5	50.9	37.4	59.6
	BCNet+PGCN [78]	I3D	69.8	62.9	52.0	39.8	24.0	49.7
	ReAct [79]	TSN	69.2	65.0	57.1	47.8	35.6	55.0
Transformer	TALLFormer [80]	swin	76.0	–	63.2	–	34.5	59.2
	STPT [81]	–	70.6	65.7	56.4	44.6	30.5	53.6
	TCANet [11]	TSN	60.6	53.2	44.6	36.8	26.7	44.3
	TadTR [30]	I3D	74.8	69.1	60.1	46.6	32.8	56.7
	Actionformer [34]	I3D	82.1	77.8	71.0	59.4	43.9	66.8
	LGAFormer (ours)	I3D	**82.4**	**78.9**	**71.8**	**60.4**	**45.2**	**67.7**

Table 2 Result on ActivityNet1.3. We report mAP at different tIoU thresholds. Average mAP in [0.5:0.05:0.95] is reported on ActivityNet1.3, where bolded text indicates the best results

Model	Feature	0.5	0.75	0.95	Avg.
BSN [75]	TSN	46.5	30.0	8.0	30.0
BMN [76]	TSN	50.1	34.8	8.3	33.9
G-TAD [46]	TSN	50.4	34.6	9.0	34.1
AFSD [37]	I3D	52.4	35.2	6.5	34.3
P-GCN [82]	I3D	48.3	33.2	3.3	31.1
TadTR [30]	I3D	49.1	32.6	8.5	32.3
TALLFormer [80]	swin	54.1	36.2	7.9	35.6
STPT [81]	–	51.4	33.7	6.8	33.4
TCANet [11]	SlowFast	54.3	**39.1**	8.4	**37.6**
VSGN [49]	I3D	52.3	35.2	8.3	34.7
Actionformer [34]	R(2+1)D	54.7	37.8	8.4	36.6
LGAFormer (ours)	R(2+1)D	**55.0**	37.9	**8.4**	36.7

Table 3 Results on EPIC-Kitchens 100.We report mAP at different tIoU thresholds and the average mAP in [0.1:0.1:0.5], where bolded text indicates the best results. V. and N. denote the verb and noun sub-tasks, respectively

	Method	0.1	0.2	0.3	0.4	0.5	Avg.
V.	BMN [76]	10.8	8.8	8.4	7.1	5.6	8.4
	G-TAD [46]	12.1	11.0	9.4	8.1	6.5	9.4
	Actionformer [34]	26.6	25.4	24.2	22.3	19.1	23.5
	LGAFormer(ours)	**27.9**	**27.0**	**25.3**	**23.2**	**19.6**	**24.6**
N.	BMN [76]	10.3	8.3	6.2	4.5	3.4	6.5
	G-TAD [46]	11.0	10.0	8.6	7.0	5.4	8.4
	Actionformer [34]	25.2	24.1	**22.7**	**20.5**	**17.0**	21.9
	LGAFormer(ours)	**25.4**	**24.4**	22.6	20.0	16.9	**21.9**

specific experimental results, the effect of our detection head is not significant in ActivityNet, which may be because each video in this dataset contains few action instances, resulting in our classification head's uneven classification of action background and foreground. The TCANet, which performs best in ActivityNet, is a two-stage model. Thanks to its complex design, it has a significant advantage in generating proposals and localizing action boundaries. Moreover, the model utilizes more powerful SlowFast features, which is also one of its strengths. In addition, the longer duration of each action also indicates that our model is still deficient in long duration action detection.

On EPIC-Kitchens 100, we achieved 24.6% and 21.9% average mAP for verb and noun, respectively. Since this dataset is a first-person dataset, the detection includes not only the action itself, but also the direct relationship between the action and the target object of the action. The average mAP of our method in the verb subtask is 1.1% higher than that of Actionformer. However, the effect in the noun subtask is only the same as that of Actionformer, indicating that there is space for improvement in the recognition of the relationship between action and target object in our model.

4.4 Ablation study

We performed a large number of ablation experiments on THUMOS14 to verify the impact of different structural designs and different choices of hyperparameters on the final detection results and model efficiency.

Firstly, we present a detailed listing of the structure of each module in the LGAFormer network in Table 4, including the Projection layer, LATM, GATM, and the detection head. We also provide the specific implementations, output feature sizes, parameter counts, and computational complexities of each module. Both LATM and GATM have multiple layers, and their output feature dimensions vary. Here, l denotes that the module is the l-th transformer layer in the network. We represent the computational complexity using the GMACs value. Since subsequent feature scales are continuously changing, leading to different computational requirements for each layer of LATM and GATM, we respectively use the computational complexity required when placing LATM and GATM in the first layer of the network as reference values. The following ablation experiments will analyze the impact of these modules on the overall model's detection performance and efficiency.

4.4.1 Choice of backbone and head

We first conduct a series of experiments to investigate the effects and computational consumption of different backbone designs, namely transformer with

Table 4 The structure and detailed information of each module in LGAFormer, including the composition of the modules, parameter count, and output feature dimensions

Module	Layers	Output size	Params	GMACs
Projection	2*(conv(k=3,s=1),layernorm(),ReLu())	$T \times 512$	3.9M	9.06
LATM	layernorm(),self-attention(),layernorm()			
	MLP{conv(k=1,s=1),GELU(),conv(k=1,s=1)}	$T/2^{l-2} \times 512$	3.1M	7.33
	maxpooling()			
GATM	layernorm(),self-attention(),layernorm()			
	MLP{conv(k=1,s=1),GELU(),conv(k=1,s=1)}	$T/2^{l-2} \times 512$	3.1M	12.69
	maxpooling()			
Cls-head	2*(conv(k=3,s=1),ReLu()),conv(k=3,s=1)	$\lfloor T/32 \times output, \cdots, T \times output \rfloor$	1.6M	7.27
Reg-head	2*(conv(k=3,s=1),ReLu()),conv(k=3,s=1)			
	2*(conv(k=3,s=1),ReLu()),conv(k=3,s=1)	$\lfloor T/32 \times output, \cdots, T \times output \rfloor$	4.8M	21.91
	2*(conv(k=3,s=1),ReLu()),conv(k=3,s=1)			

all local self-attention, transformer with all global self-attention and our LGA-Former. When these backbones do not use the detection head proposed in this paper, but just the typical classification head and the regression head, the results are shown in the first three rows of Table 5. From the results, we can also verify the idea we proposed in Sect. 1; here, the detection result using global self-attention is the same as that of local self-attention. If we choose to use global self-attention as a whole, it will cause a lot of waste of computational consumption, the MAC of global self-attention will be 28% higher than that of local self-attention. While using our local and global self-attention structure, the final average mAP would be 0.1% higher than the other structures on the THUMOS14 dataset. At the same time, the MAC is only 3.8% higher than local self-attention, and 19% lower than global self-attention. These data also demonstrate that our structure design strikes a good balance between efficiency and detection effectiveness.

In terms of detection heads, when each of the three backbones uses our specially designed detection head, it is found that although the models of the first two (local self-attention, global self-attention) do not improve in the final average mAP, the respective mAPs at tIoU = 0.7 improve by 0.4% and 1.4%, indicating the improvement in detection boundary accuracy of our proposed detection head, but the reason for the reduced effect at lower tIoU needs to be further explored. In contrast, when paired with our local and glocal self-attention backbone, the detection head exerts a more tremendous advantage, with mAP at tIoU = 0.7 and average mAP improving by 1.7% and 0.8%, respectively.

4.4.2 Number of LATM and GATM

After determining the structure of our backbone network, we also performed many ablation experiments with the specific design of the backbone network. Since our backbone network is mainly composed of LATM and GATM, the specific number of LATM and GATM modules will also have a significant impact on the experimental results. The total number of transformer blocks in our backbone network is 7. We then start from the structure of all LATM and keep adjusting the number of LATM and GATM modules to get the experimental results as shown in Table 6 to reveal the changes of detection effect and GMACs by different structure designs. In this table, L()G() represents LATM at the shallow layers of the network, and GATM at the

Table 5 Comparison between different backbone types and head types. We experiment between three backbones (transformer with local self-attention, global self-attention, local and global self-attention) and two detection heads to compare their detection results and computational consumption

Local	Global	Local and Global	Head	0.7	Avg.	GMACs
√				43.9	66.8	45.3
	√			43.9	66.8	57.8
		√		43.5	66.9	47.0
√			√	44.3	66.8	59.9
	√		√	**45.3**	66.8	72.4
		√	√	45.2	**67.7**	61.9

Bolded values indicate the best results

Table 6 Analysis of the number of LATM and GATM. We set different amounts of LATM and GATM in the experiment to analyze the effect on the detection results

		0.3	0.4	0.5	0.6	0.7	Avg.	GMACs
L(0)	G(7)	81.4	77.2	70.5	59.5	45.3	66.8	72.4
L(2)	G(5)	**82.4**	**78.9**	**71.8**	60.4	45.2	**67.7**	61.7
L(3)	G(4)	81.6	77.5	71.2	59.7	44.4	66.9	60.5
L(4)	G(3)	81.8	77.8	71.0	60.2	44.4	67.0	60.2
L(5)	G(2)	82.0	78.2	70.9	59.4	44.7	67.0	60.1
L(6)	G(1)	82.0	78.2	71.6	**60.6**	45.8	67.6	60.1
L(7)	G(0)	81.5	77.6	71.4	59.0	44.3	66.8	59.9
G(2)	L(5)	81.5	78.1	71.3	59.7	**46.7**	67.5	70.7
G(3)	L(4)	82.2	78.3	70.7	59.7	44.8	67.1	72.1
G(4)	L(5)	82.2	78.2	71.1	59.3	44.8	67.1	72.4

Bolded values indicate the best results

deep layers. Additionally, to validate the point we previously raised, we conducted a set of comparative experiments where GATM is placed in the shallow layers of the network and LATM in the deep layers, denoted by G()L(). The results show that when LATM is placed at the shallow layers and the quantity is 2, the best performance is achieved. At the same time, the computational complexity is significantly reduced compared to when GATM is placed in the shallow layers of the network. The detection results for the other designs are mostly the same but slightly higher than all LATM or all GATM designs. Therefore, in our final model structure, we choose the number of LATM to be 2 and the number of GATM to be 5 for better detection results.

4.4.3 Window size of LATM

In this section, we follow Actionformer to investigate the window size of local self-attention in LATM, because the window size affects how large the self-attention mechanism attends to a range of visual information. We preset four window sizes and one global window, and the results are shown in Table 7, demonstrating the different detection results and computational complexity of different window sizes. The results show that different window sizes have a large impact on the detection effect, while the change in computational complexity is small. However,

Table 7 Ablation on window size of local self-attention. We report MACs and mAP by varying the local window size for self-attention in our model

Win size	0.3	0.5	0.7	Avg.	GMACs (Backbone)
9	82.1	70.7	44.5	67.1	32.49
19	**82.4**	**71.8**	45.3	**67.7**	32.57
25	81.4	70.6	44.9	66.8	32.62
37	82.2	71.6	**45.6**	67.6	32.72
Global	81.3	70.5	45.3	66.8	43.29

Bolded values indicate the best results

Table 8 Ablation on the number of feature pyramid layers

#Levels	0.3	0.5	0.7	Avg.
1	71.4	52.3	16.0	48.8
2	75.3	59.1	25.8	56.2
3	76.9	64.5	36.5	60.8
4	79.8	67.9	40.8	64.7
5	81.2	70.7	43.4	66.4
6	**82.4**	**71.8**	**45.3**	**67.7**
7	81.5	71.0	44.2	67.2

Bolded values indicate the best results

Table 9 Ablation on the effectiveness of detection of head

Head	(average mAP)		
	Thumos14	ActivityNet1.3	EPIC-Kitchens100
No-head	66.91	36.64	23.81
Use-head	67.71	36.73	24.63

the local window has a great advantage over the global window not only in terms of detection effect but also in terms of computational cost.

4.4.4 Level number of feature pyramid

After the experiments on the backbone part, we conducted some experiments on the design of the neck network to explore the effect of the number of the feature pyramid layers on the detection effect and computational cost. The results are shown in Table 8, when the feature pyramid design is not used ($N = 1$), the detection effect is greatly reduced, while when N is set to 2 there is already a great improvement. This proves the importance of the feature pyramid design for our model. Also, in the final results, it is shown that the best results are achieved when $N = 6$.

4.4.5 The effectiveness of detection head

Next, to validate the effectiveness of the proposed detection head, we conducted experiments on three different datasets. By comparing the results of using only conventional detection heads (no-head) with those using the specially designed detection head proposed in this paper(use-head), we found that the detection head in this paper improved the performance to varying degrees across these

three datasets. The results are shown in Table 9, particularly, significant improvements were observed in the Thumos14 and EPIC-Kitchens100 datasets, with average mAP increases of 0.8% and 0.82%, respectively. However, the improvement on the ActivityNet dataset was only 0.09%. This may be attributed to the relatively small number of action instances contained in ActivityNet videos, which tend to have longer durations. The fixed segment step size set in the regression head of this paper may not be conducive to detecting long-duration action instances, resulting in only marginal improvements in detection performance on ActivityNet.

4.4.6 Segment step size of detection head

Finally, we show the experimental results of setting different sizes of segment step S in the detection head, as shown in Table 10. We find that the best results are obtained when S is set to 16. However, when S is set to small, the final detection effect will be reduced, even lower than the effect without using the detection head(66.8%), because GATM is used in the latter part of our backbone network, and the subsequent detection head detects the boundary in a single segment, when S is too small, it will limit the detection effect of the features extracted by GATM. Therefore, the appropriate step S should be chosen.

4.5 Visualization

In this section, we present our detection results visually, as shown in Fig. 6, which mainly shows the boundary prediction results. From the figure, we can see that the prediction results for the start and end boundaries are mostly distributed near the actual boundaries, and there is a high probability distribution near the boundary labels, thanks to the processing of the boundary features in our detection head section, which makes the localization of the boundaries more accurate.

Table 10 Ablation on size of segment step S in the detection head

S	0.3	0.5	0.7	Avg	GMACs(Head)
8	81.4	70.8	44.2	66.5	29.07
10	81.8	70.8	44.3	66.8	29.13
12	81.2	70.5	44.3	66.3	29.20
14	81.9	70.3	44.9	67.0	29.24
16	**82.4**	**71.8**	**45.3**	**67.7**	29.29
18	81.9	70.9	43.6	66.5	29.35
20	81.6	71.5	44.4	67.0	29.40

Bolded values indicate the best results

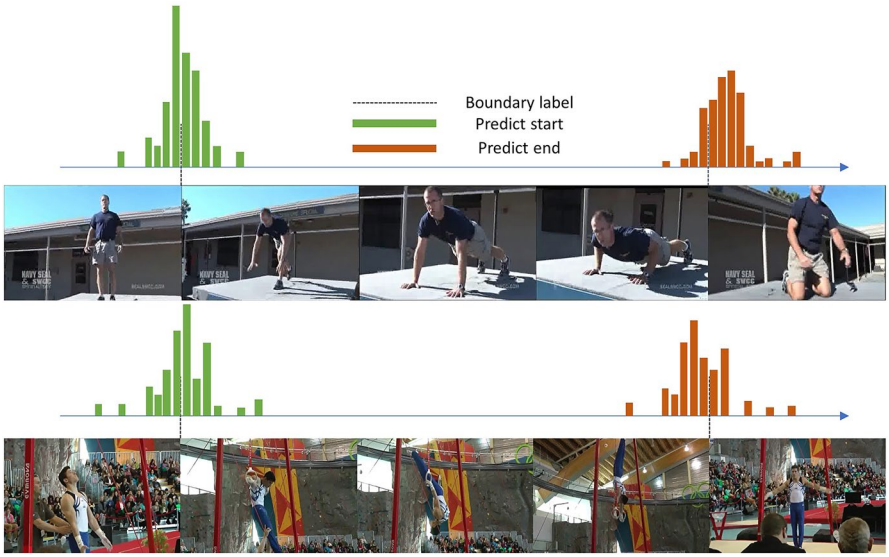

Fig. 6 Visualization of boundary localization result. The green line indicates the probability distribution of the predicted start boundary, and the red line indicates the probability distribution of the predicted end boundary

5 Conclusion

In this paper, we combine local self-attention and global self-attention, which are LATM and GATM in this paper, based on the observation of previous transformer-based TAD methods, and combine these two modules into a novel and concise model LGAFormer, using LATM to aggregate local pattern information in the shallow layer of the network and GATM to model long-term dependencies in the deeper layer of the network to obtain a feature representation with rich spatiotemporal information. Then, the final boundaries are determined by combining the instance feature and the segment feature in the detection head and obtaining the expectation within the segment through the probability distribution. Furthermore, the effectiveness of our model is verified in three public datasets (THUMOS14, ActivityNet1.3, and EPIC-Kitchens 100) with outstanding results, proving our proposed method's effectiveness.

However, there are still some shortcomings in our approach. One of the most important issues is that our method relies on features extracted by a pre-trained model rather than directly taking raw video data as input. Such a practice is not conducive to the practical deployment of subsequent algorithms. Therefore, in future work, we will consider incorporating a video feature extraction module at the frontend of the network, enabling it to participate in training along with the detector, thus making the extracted features more suitable for temporal action detection tasks. On the other hand, we found in experiments that for some complex video datasets such as EPIC-Kitchen100, the detection performance of our model did not reach a

satisfactory level. This also requires further exploration of the reasons behind it and improvements to the model.

Author contributions All authors contributed to the study conception and design. Fuxing Zhou programmed the algorithms, conducted the experiments, and wrote the main manuscript text; Haiping Zhang, Dongjing Wang, and Liming Guan supervised the experiment based on an analysis; Xinhao Zhang and Dongjin Yu designed the algorithm. All authors reviewed the manuscript.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest All the authors do not have any possible conflict of interest.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

1. Zhao Y, Xiong Y, Wang L, Wu Z, Tang X, Lin D (2017) Temporal action detection with structured segment networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp 2914–2923
2. Shou Z, Wang D, Chang S-F (2016) Temporal action localization in untrimmed videos via multi-stage cnns. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1049–1058
3. Shou Z, Chan J, Zareian A, Miyazawa K, Chang S-F (2017) Cdc: convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1417–1426
4. Dai X, Singh B, Zhang G, Davis LS, Chen YQ (2017) Temporal context network for activity localization in videos. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp 5727–5736
5. Liu Q, Wang Z (2020) Progressive boundary refinement network for temporal action detection. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 34, pp 11612–11619
6. Xu H, Das A, Saenko K (2017) R-c3d: region convolutional 3d network for temporal activity detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp 5783–5792
7. Tran D, Bourdev L, Fergus R, Torresani L, Paluri M (2015) Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp 4489–4497
8. Wang L, Yang H, Wu W, Yao H, Huang H (2021) Temporal action proposal generation with transformers. arXiv:2105.12043
9. Cheng F, Bertasius G (2022) Tallformer: temporal action localization with long-memory transformer. In: European Conference on Computer Vision
10. Li S, Zhang F, Zhao R-W, Feng R, Yang K, Liu L-N, Hou J (2022) Pyramid region-based slot attention network for temporal action proposal generation. In: British Machine Vision Conference
11. Qing Z, Su H, Gan W, Wang D, Wu W, Wang X, Qiao Y, Yan J, Gao C, Sang N (2021) Temporal context aggregation network for temporal action proposal refinement. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 485–494

12. Weng Y, Pan Z, Han M, Chang X, Zhuang B (2022) An efficient spatio-temporal pyramid transformer for action detection. In: Proceedings of Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Part XXXIV. Springer, pp. 358–375
13. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in Neural Information Processing Systems, vol 30
14. Li Y, Mao H, Girshick R, He K (2022) Exploring plain vision transformer backbones for object detection. In: European Conference on Computer Vision. Springer, pp 280–296
15. Li Y, Wu C-Y, Fan H, Mangalam K, Xiong B, Malik J, Feichtenhofer C (2022) Mvitv2: improved multiscale vision transformers for classification and detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 4804–4814
16. Carion N, Massa F, Synnaeve G, Usunier N, Kirillov A, Zagoruyko S (2020) End-to-end object detection with transformers. In: European Conference on Computer Vision. Springer, pp 213–229
17. Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B (2021) Swin transformer: hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 10012–10022
18. Ding M, Xiao B, Codella N, Luo P, Wang J, Yuan L (2022) Davit: dual attention vision transformers. In: European Conference on Computer Vision. Springer, pp 74–92
19. Tong Z, Song Y, Wang J, Wang L (2022) Videomae: masked autoencoders are data-efficient learners for self-supervised video pre-training. In: Advances in Neural Information Processing Systems, vol 35, pp 10078–10093
20. Li K, Wang Y, He Y, Li Y, Wang Y, Wang L, Qiao Y (2022) Uniformerv2: spatiotemporal learning by arming image vits with video uniformer. arXiv:2211.09552
21. Yan S, Xiong X, Arnab A, Lu Z, Zhang M, Sun C, Schmid C (2022) Multiview transformers for video recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 3333–3343
22. Liu Z, Ning J, Cao Y, Wei Y, Zhang Z, Lin S, Hu H (2022) Video swin transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 3202–3211
23. Qing Z, Zhang S, Huang Z, Wang X, Wang Y, Lv Y, Gao C, Sang N (2023) Mar: masked autoencoders for efficient action recognition. IEEE Trans Multimed
24. Dai R, Das S, Kahatapitiya K, Ryoo MS, Brémond F (2022) Ms-tct: multi-scale temporal convtransformer for action detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 20041–20051
25. Tolstikhin IO, Houlsby N, Kolesnikov A, Beyer L, Zhai X, Unterthiner T, Yung J, Steiner A, Keysers D, Uszkoreit J (2021) Mlp-mixer: an all-mlp architecture for vision. In: Advances in Neural Information Processing Systems, vol 34, pp 24261–24272
26. Yu W, Luo M, Zhou P, Si C, Zhou Y, Wang X, Feng J, Yan S (2022) Metaformer is actually what you need for vision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 10819–10829
27. Shi D, Zhong Y, Cao Q, Ma L, Li J, Tao D (2023) Tridet: temporal action detection with relative boundary modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 18857–18866
28. Basu S, Gupta M, Rana P, Gupta P, Arora C (2023) Radformer: transformers with global-local attention for interpretable and accurate gallbladder cancer detection. Med Image Anal 83:102676
29. Kumie GA, Habtie MA, Ayall TA, Zhou C, Liu H, Seid AM, Erbad A (2024) Dual-attention network for view-invariant action recognition. Complex Intell Syst 10(1):305–321
30. Liu X, Wang Q, Hu Y, Tang X, Zhang S, Bai S, Bai X (2022) End-to-end temporal action detection with transformer. IEEE Trans Image Process 31:5427–5441
31. Raghu M, Unterthiner T, Kornblith S, Zhang C, Dosovitskiy A (2021) Do vision transformers see like convolutional neural networks? In: Advances in neural information processing systems, vol 34, pp 12116–12128
32. Wu Z, Su L, Huang Q (2019) Cascaded partial decoder for fast and accurate salient object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 3907–3916
33. Hou Q, Cheng M-M, Hu X, Borji A, Tu Z, Torr PH (2017) Deeply supervised salient object detection with short connections. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 3203–3212
34. Zhang C-L, Wu J, Li Y (2022) Actionformer: localizing moments of actions with transformers. In: European Conference on Computer Vision. Springer, pp 492–510

35. Lin C, Li J, Wang Y, Tai Y, Luo D, Cui Z, Wang C, Li J, Huang F, Ji R (2020) Fast learning of temporal action proposal via dense boundary generator. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 34, pp 11499–11506
36. Yang L, Peng H, Zhang D, Fu J, Han J (2020) Revisiting anchor mechanisms for temporal action localization. IEEE Trans Image Process 29:8535–8548
37. Lin C, Xu C, Luo D, Wang Y, Tai Y, Wang C, Li J, Huang F, Fu Y (2021) Learning salient boundary feature for anchor-free temporal action localization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 3320–3329
38. Chen G, Zheng Y-D, Wang L, Lu T (2022) Dcan: improving temporal action detection via dual context aggregation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 36, pp 248–257
39. Liu X, Bai S, Bai X (2022) An empirical study of end-to-end temporal action detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 20010–20019
40. Wang L, Xiong Y, Wang Z, Qiao Y, Lin D, Tang X, Van Gool L (2018) Temporal segment networks for action recognition in videos. IEEE Trans Pattern Anal Mach Intell 41(11):2740–2755
41. Carreira J, Zisserman A (2017) Quo vadis, action recognition? A new model and the kinetics dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 6299–6308
42. Feichtenhofer C, Fan H, Malik J, He K (2019) Slowfast networks for video recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 6202–6211
43. Shou Z, Wang D, Chang S-F (2016) Temporal action localization in untrimmed videos via multi-stage cnns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1049–1058
44. Tan J, Tang J, Wang L, Wu G (2021) Relaxed transformer decoders for direct action proposal generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 13526–13535
45. Bai Y, Wang Y, Tong Y, Yang Y, Liu Q, Liu J (2020) Boundary content graph neural network for temporal action proposal generation. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII vol 16. Springer, pp 121–137
46. Xu M, Zhao C, Rojas DS, Thabet A, Ghanem B (2020) G-tad: sub-graph localization for temporal action detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 10156–10165
47. Su H, Gan W, Wu W, Qiao Y, Yan J (2021) Bsn++: complementary boundary regressor with scale-balanced relation modeling for temporal action proposal generation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 35, pp 2602–2610
48. Sridhar D, Quader N, Muralidharan S, Li Y, Dai P, Lu J (2021) Class semantics-based attention for action detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 13739–13748
49. Zhao C, Thabet AK, Ghanem B (2021) Video self-stitching graph network for temporal action localization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 13658–13667
50. Liao X, Yuan J, Cai Z, Lai J-h (2023) An attention-based bidirectional gru network for temporal action proposals generation. J Supercomput 79(8):8322–8339
51. Lin T, Zhao X, Shou Z (2017) Single shot temporal action detection. In: Proceedings of the 25th ACM International Conference on Multimedia, pp 988–996
52. Tian Z, Shen C, Chen H, He T (2019) Fcos: fully convolutional one-stage object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 9627–9636
53. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) Ssd: single shot multibox detector. In: Proceedings of Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Part I vol 14. Springer, pp 21–37
54. Law H, Deng J (2018) Cornernet: detecting objects as paired keypoints. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 734–750
55. Kong T, Sun F, Liu H, Jiang Y, Li L, Shi J (2020) Foveabox: beyound anchor-based object detection. IEEE Trans Image Process 29:7389–7398
56. Zhu C, He Y, Savvides M (2019) Feature selective anchor-free module for single-shot object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 840–849

57. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 779–788
58. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al (2020) An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929
59. Yang J, Dong X, Liu L, Zhang C, Shen J, Yu D (2022) Recurring the transformer for video action recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 14063–14073
60. Bulat A, Perez Rua JM, Sudhakaran S, Martinez B, Tzimiropoulos G (2021) Space-time mixing attention for video transformer. In: Advances in Neural Information Processing Systems, vol 34, pp 19594–19607
61. Arnab A, Dehghani M, Heigold G, Sun C, Lučić M, Schmid C (2021) Vivit: a video vision transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 6836–6846
62. Bertasius G, Wang H, Torresani L (2021) Is space-time attention all you need for video understanding? In: ICML, vol 2, p 4
63. Zhao P, Xie L, Ju C, Zhang Y, Wang Y, Tian Q (2020) Bottom-up temporal action localization with mutual regularization. In: Proceedings of Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Part VIII 16. Springer, pp 539–555
64. Liu D, Jiang T, Wang Y (2019) Completeness modeling and context separation for weakly supervised temporal action localization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 1298–1307
65. Choromanski K, Likhosherstov V, Dohan D, Song X, Gane A, Sarlos T, Hawkins P, Davis J, Mohiuddin A, Kaiser L, et al (2020) Rethinking attention with performers. arXiv:2009.14794
66. Lin T-Y, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp 2980–2988
67. Rezatofighi H, Tsoi N, Gwak J, Sadeghian A, Reid I, Savarese S (2019) Generalized intersection over union: a metric and a loss for bounding box regression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 658–666
68. Tian Z, Shen C, Chen H, He T (2019) Fcos: fully convolutional one-stage object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 9627–9636
69. Zhang S, Chi C, Yao Y, Lei Z, Li SZ (2020) Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 9759–9768
70. Bodla N, Singh B, Chellappa R, Davis LS (2017) Soft-nms-improving object detection with one line of code. In: Proceedings of the IEEE International Conference on Computer Vision, pp 5561–5569
71. Idrees H, Zamir AR, Jiang Y-G, Gorban A, Laptev I, Sukthankar R, Shah M (2017) The thumos challenge on action recognition for videos in the wild. Comput Vis Image Underst 155:1–23
72. Caba Heilbron F, Escorcia V, Ghanem B, Carlos Niebles J (2015) Activitynet: a large-scale video benchmark for human activity understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 961–970
73. Damen D, Doughty H, Farinella GM, Furnari A, Kazakos E, Ma J, Moltisanti D, Munro J, Perrett T, Price W et al (2022) Rescaling egocentric vision: collection, pipeline and challenges for epic-kitchens-100. Int J Comput Vis 130:1–23
74. Loshchilov I, Hutter F (2017) Decoupled weight decay regularization. arXiv:1711.05101
75. Lin T, Zhao X, Su H, Wang C, Yang M (2018) Bsn: boundary sensitive network for temporal action proposal generation. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 3–19
76. Lin T, Liu X, Li X, Ding E, Wen S (2019) Bmn: boundary-matching network for temporal action proposal generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 3889–3898
77. Yang M, Chen G, Zheng Y-D, Lu T, Wang L (2023) Basictad: an astounding rgb-only baseline for temporal action detection. Comput Vis Image Underst 232:103692
78. Yang H, Wu W, Wang L, Jin S, Xia B, Yao H, Huang H (2022) Temporal action proposal generation with background constraint. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 36, pp 3054–3062

79. Shi D, Zhong Y, Cao Q, Zhang J, Ma L, Li J, Tao D (2022) React: temporal action detection with relational queries. In: European Conference on Computer Vision. Springer, pp 105–121
80. Cheng F, Bertasius G (2022) Tallformer: temporal action localization with a long-memory transformer. In: European Conference on Computer Vision. Springer, pp 503–521
81. Weng Y, Pan Z, Han M, Chang X, Zhuang B (2022) An efficient spatio-temporal pyramid transformer for action detection. In: European Conference on Computer Vision. Springer, pp 358–375
82. Zeng R, Huang W, Tan M, Rong Y, Zhao P, Huang J, Gan C (2019) Graph convolutional networks for temporal action localization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 7094–7103
83. Carreira J, Zisserman A (2017) Quo vadis, action recognition? A new model and the kinetics dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 6299–6308
84. Tran D, Wang H, Torresani L, Ray J, LeCun Y, Paluri M (2018) A closer look at spatiotemporal convolutions for action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 6450–6459
85. Alwassel H, Giancola S, Ghanem B (2021) Tsp: temporally-sensitive pretraining of video encoders for localization tasks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 3173–3183

Authors and Affiliations

Haiping Zhang[1,2] · Fuxing Zhou[2] · Dongjing Wang[1] · Xinhao Zhang[2] · Dongjin Yu[1] · Liming Guan[3]

✉ Fuxing Zhou
shiyichen1213@gmail.com

Haiping Zhang
zhanghp@hdu.edu.cn

Dongjing Wang
dongjing.wang@hdu.edu.cn

Xinhao Zhang
zxh8991@163.com

Dongjin Yu
yudj@hdu.edu.cn

Liming Guan
glm@hdu.edu.cn

[1] School of Computer Science, Hangzhou Dianzi University, Qiantang, Hangzhou 310018, Zhejiang, China

[2] School of Electronics and Information, Hangzhou Dianzi University, Qiantang, Hangzhou 310018, Zhejiang, China

[3] School of Information Engineering, Hangzhou Dianzi University, Qiantang, Hangzhou 310018, Zhejiang, China