ASQuery: A Query-based Model for Action Segmentation

1st Ziliang Gan*
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
ganzl@bupt.edu.cn

2nd Lei Jin* ✉
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
jinlei@bupt.edu.cn

3rd Lei Nie *
Baidu AI Cloud
Baidu
Beijing, China
nielei01@baidu.com

4th Zheng Wang
School of Computer Science
Wuhan University
Wuhan, China
wangzwhu@whu.edu.cn

5th Li Zhou
Peking University
Beijing, China
zhouli2025@126.com

6th Liang Li
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China
liang.li@ict.ac.cn

7th Zhecan Wang
Columbia University
New York, America
zw2627@columbia.edu

8th Jianshu Li
Ant Group
Hangzhou, China
jianshu.l@antgroup.com

9th Junliang Xing
Tsinghua University
Beijing, China
jlxing@tsinghua.edu.cn

10th Jian Zhao✉
EVOL Lab, Institute of AI (TeleAI), China Telecom, Beijing, China
School of Artificial Intelligence, Optics and Electronics (iOPEN)
Northwestern Polytechnical Univerisity (NWPU), Xi'an Shanxi, China
zhaoj90@chinatelecom.cn

***Abstract*—For the task of temporal action segmentation, existing works commonly treat it as a frame-wise classification problem. In this paper, we propose a straight but effective model namely ASQuery by learning central representation of each action category, which transforms the classification problem to the similarity calculation between category-specific queries and frame features. These central representations are dynamically generated through our Transformer decoder module, endowing them more flexible and comprehensive perception of the whole video. Moreover, we first introduce the boundary query for refining segmentation results, aiding to alleviating the troublesome over-segmentation problem. ASQuery demonstrates superior performance compared to state-of-the-art models, achieving improvements of 0.9% and 4.1% in the mean metrics on two public action segmentation datasets, *i.e.*, Breakfast and Assembly101, respectively. The source codes are available at https://github.com/zlngan/ASQuery.**

***Index Terms*—Action Segmentation, Action Query, Boundary Query, Over-segmentation**

I. Introduction

Temporal action segmentation is a pivotal task in human activities understanding, aiming to predict the action category for each frame within an untrimmed video. Existing action segmentation methods typically formulate the task as a frame-wise classification problem and employ various network architectures such as temporal convolution, graph convolution, and Transformer to capture temporal dynamics.

As depicted in Fig. 1, previous works commonly utilize an action branch to aggregate features for all frames, followed by a classifier for predicting their categories. In this manner, the predictions of frames are heavily depending on the weights of frame-wise classifier, which are fixed once trained, and could be inflexible and inaccurate in the inference phase. In addition, existing methods commonly suffer from over-segmentation errors [1], which largely influence the consistency of segments. To alleviate the phenomenon, researches, *e.g.*, MS-TCN [2] and ASFormer [3] incorporate multiple refinement layers to enhance predictions but largely increase the parameters. BCN [4] and ASRF [1] introduce a boundary branch for refining nevertheless is trained independently.

To address the limitations of existing methods, we first introduce the query-based pipeline into action segmentation and propose an efficient and elegant model named ASQuery. In contrast to previous approaches, our method transforms frame-wise classification into similarity calculation between category-specific queries and frame features. Specifically, we initialize a set of learnable parameters as the action queries, which are corresponding to action categories. We design a decoder ingesting these initial queries and output updated queries which represent the cluster centers of categories feature space. The decoder consists of multiple decoder layers, and each layer comprises a cross-attention layer and a self-attention layer. In the cross-attention layer, action queries perceive the whole video features. Then these queries interact with each other in the self-attention layer. Finally, the updated action queries are employed to compute their similarity with frame features, thereby generating action scores. Compared

* equal contribution.
✉ Corresponding authors.
Work done during Gan's internship at Baidu AI Cloud.

The paper is supported by Natural Science Foundation of China No.62102039, No.82274685, and the Fundamental Research Funds for the Central Universities No.500422813.

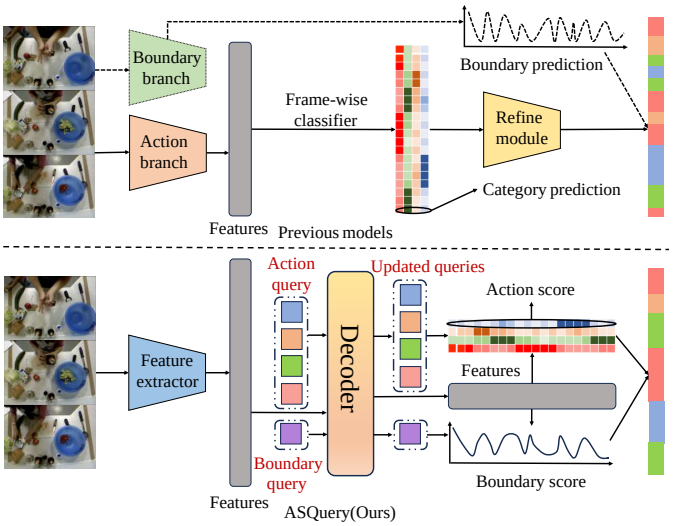

Fig. 1. Comparison of action segmentation pipelines between existing works and ASQuery. Previous works typically employ an action branch for extracting frame features and a classifier for predicting category for each frame, followed by a refining module for smoothing predictions. Some methods incorporate an additional boundary branch to predict boundaries and fuse them with class predictions for final results. In contrast, ASQuery utilizes the action and boundary queries to interact with extracted features in the decoder and output the action and boundary scores, respectively. The two types of scores are then combined for final predictions.

to weights of frame-wise classifier of existing methods, these action queries are dynamically generated. Flexibly perceiving features of all frames in the cross-attention layer and learning the interdependence of actions in the self-attention layer render their more comprehensive and robust capability. Our approach is essentially a type of metric learning, and later experiments demonstrate that the learned classificatory central representations are superior than weights of classifier. For the issue of over-segmentation, we take boundary information into consideration [4]. To be specific, we incorporate a boundary query and train it alongside the action query in an end-to-end manner. This allows the boundary query to capture abundant boundary features in the cross-attention layer and learn action category information in the self-attention layer, facilitating precise localization of boundaries.

Our contributions are summarized as follows.

- We propose ASQuery, a query-based action segmentation Transformer model that reformulates the action segmentation task from frame-wise classification to a similarity calculation process, thereby generating more accurate frame-wise predictions.
- We further propose a boundary query to efficiently and elegantly generate boundary scores, which can be jointly optimized with the action queries. By utilizing the boundary scores to refine predictions, we significantly alleviate the issue of over-segmentation.
- We validate the effectiveness of proposed modules of ASQuery by extensive ablation studies. Additionally, our model outperforms state-of-the-art by 0.9% and 4.1% in the mean metrics on two public datasets, *i.e.*, Breakfast and Assembly101.

II. Related Work

A. Action Segmentation

Action segmentation models predict the category for each frame of a raw video. MS-TCN [2] first employs multiple stacked temporal convolution networks to efficiently capture temporal dependency, yielding remarkable performance. Subsequently, several approaches, *e.g.*, MS-TCN++ [5] have been proposed to further enhance performance. Additionally, DTGRM [6] utilizes graph convolution networks to model temporal information and relations among actions. ASFormer [3] first introduces Transformer networks into action segmentation. However, these methods commonly utilize a per-frame classification formulation and pay attention to designing powerful networks for aggregating frame features. In contrast, we introduce the action queries for representing action categories, which is a pioneer of query-based model in action segmentation.

B. Over-segmentation

Action segmentation models commonly encounter the issue of over-segmentation, exhibiting fragmented predictions for action segments. Various approaches have been proposed to address this problem, such as the smoothing loss of MS-TCN [2] for reducing over-segmentation errors and the addition of multiple refine layers in ASFormer [3] for enhancing smoothness of predictions. BCN [4] utilizes a module to generate barriers and performs local barrier pooling operation for smoothing predictions, while ASRF [1] adds a boundary regression branch to predict boundaries and refines predictions through these boundaries. We refer to the idea of utilizing boundary information for smoothing predictions. In contrast to existing methods, we abstract boundary information as the boundary query and train it end-to-end along with the action query, which is more accurate and elegant.

C. Query-based Methods

The query-based pipeline was first introduced by DETR [7] in object detection, where queries are utilized to represent objects and the Hungarian matching algorithm is adopted for one-to-one matching between queries and objects within an image, thereby facilitating an end-to-end detection pipeline. MaskFormer [8] and Mask2Former [9] extend this concept to semantic segmentation, hence transforming existing per-pixel classification process into mask classification procedure. However, these methods require predictions of bounding boxes or image masks, which limits their usage in action segmentation. Our model is the first to adopt the query-based pipeline in action segmentation. Different from previous methods, our action queries are category-specific and consist with the number of categories, thereby eliminating the need for Hungarian matching.

III. METHOD

In this section, we firstly present the formulation for query-based action segmentation. Subsequently, we illustrate a comprehensive overview of all modules incorporated in ASQuery. Lastly, we elaborate on the training and inference procedures employed for ASQuery.

A. Query-based Formulation

The task of action segmentation aims to accurately classify each frame in a video into its corresponding category. Given a raw video consisting of L frames, researchers commonly employ a pre-trained model to extract a $1 \times D$ feature vector for every frame. Consequently, the extracted video features can be represented as $X = \{x_1, ..., x_L\} \in \mathbb{R}^{L \times D}$. Existing methods consider this as a frame-wise classification problem and primarily focus on frame-wise features. They utilize networks such as 1D CNN and Transformer to aggregate temporal information from frames in the perceive field. It can be denoted as:

$$F = \Phi(X) = \{f_1, ..., f_L\} \in \mathbb{R}^{L \times C}, \tag{1}$$

where Φ denotes the feature extraction network, f_i represents the feature of i^{th} frame, and F denotes the output video feature. Then they use a classifier Δ with K categories to predict the category of each frame:

$$P = \Delta(W, F) = \{p_1, ..., p_L\} \in \mathbb{R}^{L \times K}, \tag{2}$$

where P represents the category prediction results, while $W \in \mathbb{R}^{\mathbb{C} \times \mathbb{K}}$ denote weights of Δ.

For our query-based model, we adopt action queries to represent cluster centers for corresponding categories, hence transforming the classification problem into comparing similarity(distances) between action queries and frame features. Specifically, for K categories, we initial K learnable action queries $Q_A = \{Q_A^1, ..., Q_A^K\} \in \mathbb{R}^{K \times C}$, where each query represents its corresponding action category. For the i^{th} query $Q_A^i \in \mathbb{R}^{1 \times C}$ and the video feature F, we calculate similarity as the action score $A^i \in \mathbb{R}^{1 \times L}$, which is denoted as:

$$A^i = \Theta(Q_A^i, F) = \{\theta(Q_A^i, f_1), ..., \theta(Q_A^i, f_L)\}, \tag{3}$$

where θ denotes similarity function, here we use dot product. For all action queries Q_A, we obtain the action scores $A = \Theta(Q_A, F) = \{A^1, ..., A^K\} \in \mathbb{R}^{K \times L}$. Though action queries A and weights W of classifier play similar roles, they exhibit several differences. First, weights W are fixed once trained, while queries A are dynamically generated. Second, queries A share the same feature space with frame features by cross-attention layer and interact with each other in the self-attention layer, hence having stronger discriminative ability than weights W. (Demonstrated by the later experiments) Existing action segmentation models generally encounter the issue of over-segmentation, leading to more fragmented predicted outcomes. Existing approaches, *e.g.*, MS-TCN [2], ASFormer [3] use refine networks or ASRF [1] and BCN [4] use action boundaries to alleviate the over-segmentation phenomenon. These methods typically require additional networks, such as refine networks or boundary predict networks to accomplish their targets.

For the over-segmentation problem, different from existing solutions, such as refine networks, boundary branches, query-based formulation facilitates a more elegant way to effectively smooth the prediction results. This paper introduces the boundary query to predict category-agnostic boundaries. Similar to action queries, the boundary query is represented as a learnable embedding $Q_B \in \mathbb{R}^{1 \times C}$. The predicted boundaries $B \in \mathbb{R}^{1 \times L}$ is obtained as following:

$$B = \Theta(Q_B, F) = \{\theta(Q_B, f_1), ..., \theta(Q_B, f_L)\}, \tag{4}$$

where F and Θ are the same as that before. The boundary query can flexibly aggregate features of boundary frames and interact with the action query in the decoder, thereby acquiring abundant information regarding boundaries and action categories. During inference phase, the boundary score can be seamlessly integrated with action scores to refine initial predictions and obtain final smooth results.

B. ASQuery

The ASQuery model, as illustrated in Fig. 2, comprises three modules: 1) a feature extraction module that integrates a backbone and a neck to aggregate and enhance multi-level features; 2) a Transformer decoder module, incorporating multiple Transformer decoder layers, utilized for updating action and boundary queries; and 3) a segmentation module for producing action and boundary scores, then predict the final results during the inference phase.

Feature Extraction Module. The feature extraction module comprises a backbone and a neck, which takes the extracted video features $X \in \mathbb{R}^{L \times D}$ as input and generates multi-level features as output (The raw videos are commonly preprocessed by I3D [10] for extracting features). To construct the backbone, we initially employ a shallow convolutional network due to its strength in incorporating local context [11] and stabilizing training of vision Transformers [12].

We adopt the Transformer network to aggregate features owing to its inherent capability in effectively integrating temporal context across the entire sequence. However, the vanilla self-attention operation has a complexity of $O(T^2)$ in both memory and time, rendering them inefficient for processing long sequences. Hence, we constrain attention to a local window to strike a balance between temporal receptive field and computational efficiency.

In the Transformer layer, we incorporate a 1D depthwise convolution (DConv) [13] prior to multi-head self-attention. To capture actions at various temporal scales, we employ downsampling in the last few Transformer layers by controlling the stride of DConv. In addition, following existing works [2], we enlarge the temporal receptive field by increasing dilation rates of the convolutional networks.

Regarding the neck network, we utilize a simple 1D feature pyramid network (FPN) [14] to enhance features. It outputs a set of features, which are denoted as $\{X_1, ..., X_M\}$. X_i represents the i^{th} feature, with a size of $(2^{i-M}L \times C)$. The

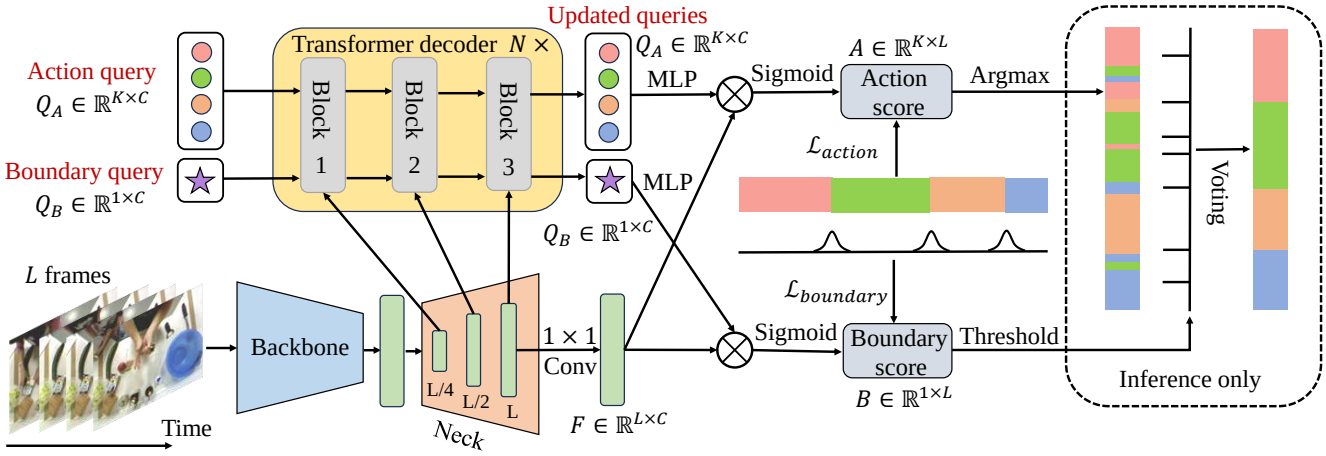

Fig. 2. Overall pipeline of ASQuery. We use a backbone and a neck to process pre-extracted video features and output multi-scale features. The Transformer decoder takes action and boundary queries as the Query, and takes video features as the Key and Value. It outputs the updated action and boundary queries, which contain action category and boundary information, respectively. Next, ASQuery generates the action and boundary scores by multiplying updated queries with the video embedding. During inference, we first obtain predicted boundaries according to the boundary score, then combine the action score and boundaries through majority voting to get the final predictions.

temporal scale of the longest feature X_M is equivalent to the number of frames in original video. Finally, we employ a 1×1 convolution to transform the video feature X_M into the feature representation $F \in \mathbb{R}^{L\times C}$.

Transformer Decoder Module. The Transformer decoder module process initial action and boundary queries, as well as multi-level video features, then yield updated queries that contain comprehensive category and boundary information. This paper adopt the standard Transformer decoder, which consists of multiple Transformer decoder layers, with each layer containing several Transformer blocks. These blocks are aligned with the outputs of FPN, which means that the i^{th} block takes the action and boundary queries as **Q**uery and takes video feature X_i as **K**ey and **V**alue. Within the block, there is a self-attention layer followed by a cross-attention layer. Specifically, for the i^{th} Transformer block $\mathrm{Tb_i}$, the process can be denoted as:

$$\begin{aligned} Q_A^{\text{out}}, Q_B^{\text{out}} &= \mathrm{Tb_i}(Q_A^{\text{in}}, Q_B^{\text{in}}, X_i) \\ &= \mathrm{CAttn}(\mathrm{SAttn}(Q_A^{\text{in}}, Q_B^{\text{in}}), X_i), \end{aligned} \tag{5}$$

where Q^{in} and Q^{out} denote the input and updated queries, respectively. SAttn and CAttn denote the self-attention and cross-attention layers, respectively. For clarity, we omit the fully connected layer, layernorm, and position encoding. The self-attention layer facilitates the perception of action categories and boundary information via the interaction between queries, thereby enhancing the discriminative ability. While action-specific and boundary-specific features are extracted with the guide of corresponding queries in the cross-attention layer. Hence, the output queries from the Transformer decoder effectively encode action and boundary characteristics.

Segmentation Module. The segmentation module is designed to compute action and boundary scores based on updated queries. To obtain these scores, two Multi-Layer Perceptron (MLP) with two hidden layers are used to convert the action and boundary queries into corresponding embeddings. Finally, we obtain action scores A by taking the dot product between action queries Q_A and video features F. Here F is generated by the feature extraction module. A sigmoid activation is applied to normalize the scale to $(0, 1)$. The boundary score B is obtained in a similar way.

C. Training and Inference

Training. During the training phase, we employ frame-wise and action-wise losses to optimize action scores. Specifically, for the frame-wise loss, we adopt the standard focal loss [15] to measure the discrepancy between ground truth and predicted action scores. For the action-wise loss, given a video containing actions of N categories, we derive the class mask label $M = \{M_1, ..., M_N\} \in \mathbb{R}^{N\times L}$, here M_i is generated as:

$$M_i = \{y_1, ..., y_L\}, \quad y_j = \begin{cases} 1 & \text{if } j^{gt} == i, \\ 0 & \text{else}, \end{cases} \tag{6}$$

where j^{gt} denotes the ground truth category of the j^{th} frame. We extract the corresponding action score $A_M \in \mathbb{R}^{N\times L}$ from A for the selected N classes. The action-wise loss is computed using dice loss [16]. The combined loss for action scores is calculated as $\mathcal{L}_{\text{action}} = \lambda_{\text{focal}}\mathcal{L}_{\text{focal}}(A, Y) + \lambda_{\text{dice}}\mathcal{L}_{\text{dice}}(A_M, M)$. To compute the boundary loss, we obtain the boundary supervision signal $B^{gt} \in \mathbb{R}^{1\times L}$ as follows:

$$B^{gt} = g(\hat{B}), \hat{B} = \{b_1, ..., b_L\}, b_i = \begin{cases} 0 & \text{if } i^{gt} == (i+1)^{gt}, \\ 1 & else, \end{cases} \tag{7}$$

where $g(\cdot)$ represents a Gaussian filter utilized for smoothing boundaries. The binary cross-entropy loss is employed for boundary loss. Auxiliary losses are incorporated to account

for the action and boundary scores generated by intermediate Transformer blocks, which are denoted as $\mathcal{L}_{\text{aux}}^{\text{action}}$ and $\mathcal{L}_{\text{aux}}^{\text{boundary}}$. Hence, the total loss is obtained as follows:

$$\begin{aligned}\mathcal{L}_{\text{total}} &= \mathcal{L}_{\text{action}} + \mathcal{L}_{\text{boundary}} + \mathcal{L}_{\text{aux}}^{\text{action}} + \mathcal{L}_{\text{aux}}^{\text{boundary}} \\ &= \lambda_{\text{focal}}\mathcal{L}_{\text{focal}}(A, Y) + \lambda_{\text{dice}}\mathcal{L}_{\text{dice}}(A_M, M) \\ &+ \lambda_{\text{cross}}\mathcal{L}_{\text{cross}}(B, B^{gt}) + \lambda_{\text{aux}}\mathcal{L}_{\text{aux}},\end{aligned} \tag{8}$$

where $\mathcal{L}_{\text{aux}}$ represents $\mathcal{L}_{\text{aux}}^{\text{action}} + \mathcal{L}_{\text{aux}}^{\text{boundary}}$ for simplification. In experiments, we set the hyper-parameters as λ_{focal} = λ_{dice} = λ_{cross} = λ_{aux} = 1.0.

Inference. We design a specialized process that integrates action and boundary scores to generate final predictions in the inference phase. Firstly, the category of each frame is determined by its highest action score. Subsequently, action boundaries are derived from B. The frame exhibiting a boundary score that achieves a local maximum and surpasses a predefined threshold ρ is considered the boundary frame. Assuming the presence of only one action within each segment, the final prediction of a segment is determined through majority voting on the categories of frames within it. Consequently, we partition all video frames into distinct action segments via the process.

IV. Experiments

A. Experiment Settings

Dataset. We perform extensive experiments on two largest datasets: the Breakfast [17] and Assembly101 datasets [18].

Breakfast consists of 1,712 third-person view videos depicting 48 action classes related to preparing breakfast. Following [2], we adopt the video features extracted by I3D [10] as inputs and perform four-fold cross-validation and report the average performances.

Assembly101 is a large-scale procedural activity dataset comprising 202 action classes that involve people assembling and disassembling 101 toy vehicles. We utilize the TSM [19] features provided by assembly101. 4,694 videos are utilized for training, while 1,424 videos are reserved for evaluation.

Metrics. In line with previous works [2], we employ frame-wise accuracy (Acc), segmental edit distance (Edit), and segmental F1 scores at overlap threshold 10%, 25%, and 50% (F1@{10, 25, 50}) as the evaluation metrics. Notably, the Edit and F1 scores are largely influenced by the over-segmentation issue, fragmented predictions would lead to low Edit and F1 scores metrics.

Implement Details. We employ the AdamW [20] optimizer and the cosine learning rate schedule with an initial learning rate of 1×10^{-4}. Our model is trained for 25 epochs, including a linear warmup of 5 epochs. The batch size is set to 4.

B. Comparison and Result Analysis

Quantitative Comparison. As presented in Tab. I, we compare ASQuery with state-of-the-art action segmentation models on Breakfast and Assembly101 datasets. It can be seen that our method exceeds other algorithms on almost metrics. Notably, ASQuery achieves a significant improvement over DiffAct [27] by +1.9% on F1@50 and +1.5% on accuracy on the Breakfast dataset. In addtion, ASQuery exhibits a significant advantage over other approaches in terms of F1 scores on the Assembly101 dataset, which demonstrates its powerful capability of alleviating the over-segmentation problem. These findings demonstrate the superiority of the query-based action segmentation pipeline over the traditional frame-wise classification framework.

TABLE I
COMPARSION WITH STATE-OF-THE-ART METHODS. OUR METHOD ACHIEVES SUPERIOR RESULTS ON ALMOST METRICS ON BOTH BREAKFAST [17] AND ASSEMBLY101 [18] DATASETS. *Mean* DENOTES THE MEAN NUMBER OF FIVE EVALUATION METRICS. GRAY DENOTES THAT THE RESULTS ARE OBTAINED BY OUR IMPLEMENTATION. THE DATA PRE-PROCESSING AND AUGMENTATION ARE IN LINE WITH ASSEMBLY101 TO ENSURE FAIRNESS.

Method	Breakfast						Assembly101					
	F1@{10,25,50}			Edit	Acc	Mean	F1@{10,25,50}			Edit	Acc	Mean
MS-TCN'19, CVPR [2]	52.6	48.1	37.9	61.7	66.3	53.3	31.6	28.0	21.0	30.7	38.	29.9
MS-TCN++, PAMI'20 [5]	64.1	58.6	45.9	65.6	67.6	60.4	31.6	27.8	20.6	30.7	37.1	29.7
BCN, ECCV'20 [4]	68.7	65.5	55.0	66.2	70.4	65.2	29.6	27.1	21.2	28.4	34.5	28.2
DTGRM, AAAI'21 [6]	68.7	61.9	46.6	68.9	68.3	62.9	25.7	22.5	16.8	26.2	37.0	25.6
C2F-TCN, arXiv'21 [21]	72.2	68.7	57.6	69.6	76.0	68.8	33.3	29.0	21.3	32.4	39.2	31.0
G2L, CVPR'21 [22]	74.9	69.0	55.2	73.3	70.7	68.6	34.0	30.3	22.7	33.6	38.1	31.7
HASR, ICCV'21 [23]	74.7	69.5	57.0	71.9	69.4	68.5	30.9	27.4	20.4	30.6	37.5	29.4
ASRF, WACV'21 [1]	74.3	68.9	56.1	72.4	67.6	67.9	26.3	23.4	17.0	27.2	30.6	24.9
ASFormer, BMVC'21 [3]	76.0	70.6	57.4	75.0	73.5	70.5	32.1	28.6	21.8	31.8	38.6	30.6
DTL, NeurIPS'22 [24]	78.8	74.5	62.9	77.7	75.8	73.9	-	-	-	-	-	-
UVAST, ECCV'22 [25]	76.9	71.5	58.0	77.1	69.7	70.6	26.3	22.7	16.3	27.3	34.9	25.5
LTContext, ICCV'23 [26]	77.6	72.6	60.1	77.0	74.2	72.3	33.9	30.0	22.6	30.4	**41.2**	31.6
DiffAct, ICCV'23 [27]	80.3	75.9	64.6	**78.4**	76.4	75.1	-	-	-	-	-	-
ASQuery (Ours)	**80.7**	**76.5**	**66.5**	**78.4**	**77.9**	**76.0**	**37.8**	**35.6**	**29.4**	**35.3**	40.4	**35.7**

TABLE II
ABLATION STUDY ON THE PROPOSED COMPONENTS. A PRESENTS THE ACTION QUERY, B DENOTES THE BOUNDARY QUERY.

A	B	F1@{10,25,50}			Edit	Acc	Mean
		61.4	56.4	45.8	63.3	73.0	60.0
✓		58.4	54.2	45.1	59.8	76.9	58.9
	✓	78.3	73.6	62.8	76.0	75.1	73.1
✓	✓	**80.7**	**76.5**	**66.5**	**78.4**	**77.9**	**76.0**

C. Ablation study

We conduct comprehensive ablation experiments on the Breakfast dataset [17].

Analysis on Proposed Components. Tab. II presents the quantitative analysis of action and boundary queries. In the baseline setting (Row 1 of Tab. II), frame-wise categories are predicted by classifier solely based on frame features. It can be seen that the action query demonstrates a 3.9% increase in frame-wise accuracy, while the boundary query effectively alleviates the issue of over-segmentation, thereby yielding higher F1 scores and Edit metrics. Besides, the two types of queries exhibit a mutually reinforcing effect, and our model attains optimal performance through their integration.

We visualize the refining process in Fig. 3. Row 2 depicts the boundary score curve. X-axis denotes indexes of frames in the video, Y-axis represents the boundary scores. The green dotted line denotes the boundary threshold, we set it at 0.1. The blue lines represent the predicted boundaries according to the boundary score, while the red lines signify the actual

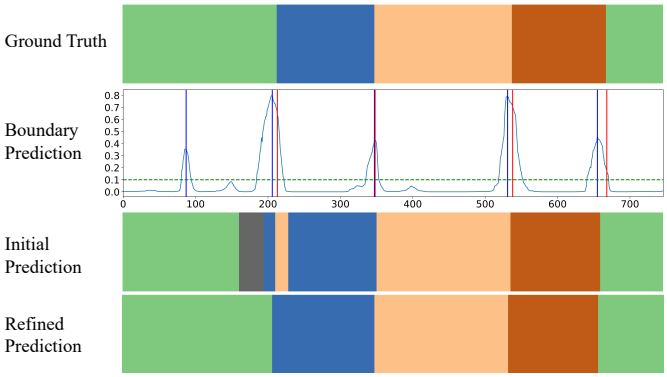

Fig. 3. Visualization results of refining predictions via boundaries on the "P04 webcam01 P04 milk" video from Breakfast dataset.

action boundaries. It can be seen that the real boundaries are all well predicted, and the fragmented results are corrected.

TABLE III
COMPARISON OF METHODS FOR GENERATING ACTION BOUNDARY. "FEATURE" DENOTES REGRESSING BOUNDARIES BY VIDEO FEATURES, "QUERY" REPRESENTS GENERATING THE BOUNDARY QUERY BY EMPLOYING AN MLP TO ACTION QUERIES. † INDICATES THAT THE BOUNDARY QUERY DOES NOT INTERACT WITH ACTION QUERIES IN THE SELF-ATTENTION LAYER.

Methods	F1@{10,25,50}			Edit	Acc	Mean
Feature	80.1	76.0	65.8	78.0	77.1	75.4
Query	80.4	76.1	65.7	**78.4**	76.5	75.4
Ours†	80.2	75.8	65.5	78.0	77.3	75.3
Ours	**80.7**	**76.5**	**66.5**	**78.4**	**77.9**	**76.0**

Analysis on Boundary Query. We perform comparative experiments to analyze the boundary query and report results in Tab. III. Row 1 shows the results by directly regressing boundaries based on frame features. Row 2 reports the results of predicting boundaries via a boundary query, which is obtained by employing a 3-layer MLP on action queries. In contrast to ASQuery, row 3 differs in that the boundary query does not interact with action queries within the self-attention layer of the Transformer decoder. Compared to other methods, the boundary query in ASQuery is jointly optimized with action queries. It can be seen that ASQuery outperforms other methods, highlighting the effectiveness of our query-based Transformer structure and the interaction between boundary and action queries.

V. CONCLUSION

This paper proposes a novel query-based method named ASQuery that employs the action query for representing action categories, thereby transforming the traditional frame-wise classification problem into the similarity calculation between queries and frame features. Additionally, the boundary query is introduced to precisely locate action boundaries and refine predictions. Moreover, ASQuery integrates them for joint optimization and achieves state-of-the-art performance on two challenging datasets, *i.e.*, Breakfast and Assembly101.

REFERENCES

[1] Yuchi Ishikawa, Seito Kasai, Yoshimitsu Aoki, and Hirokatsu Kataoka. Alleviating over-segmentation errors by detecting action boundaries. In *WACV*, 2021.

[2] Yazan Abu Farha and Jurgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *CVPR*, 2019.

[3] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. Asformer: Transformer for action segmentation. In *BMVC*, 2021.

[4] Zhenzhi Wang, Ziteng Gao, Limin Wang, Zhifeng Li, and Gangshan Wu. Boundary-aware cascade networks for temporal action segmentation. In *ECCV*, 2020.

[5] Shi-Jie Li, Yazan AbuFarha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. Ms-tcn++: Multi-stage temporal convolutional network for action segmentation. *TPAMI*, 2020.

[6] Dong Wang, Di Hu, Xingjian Li, and Dejing Dou. Temporal relational modeling with self-supervision for action segmentation. In *AAAI*, 2021.

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[8] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *NeurIPS*, 2021.

[9] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022.

[10] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[11] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *NeurIPS*, 2019.

[12] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. *NeurIPS*, 2021.

[13] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[16] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016.

[17] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *CVPR*, 2014.

[18] Fadime Sener, Dibyadip Chatterjee, Daniel Shelepov, Kun He, Dipika Singhania, Robert Wang, and Angela Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. In *CVPR*, 2022.

[19] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019.

[20] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019.

[21] Dipika Singhania, Rahul Rahaman, and Angela Yao. Coarse to fine multi-resolution temporal convolutional network. *arXiv*, 2021.

[22] Shang-Hua Gao, Qi Han, Zhong-Yu Li, Pai Peng, Liang Wang, and Ming-Ming Cheng. Global2local: Efficient structure search for video action segmentation. In *CVPR*, 2021.

[23] Hyemin Ahn and Dongheui Lee. Refining action segmentation with hierarchical video representations. In *ICCV*, 2021.

[24] Ziwei Xu, Yogesh Rawat, Yongkang Wong, Mohan S Kankanhalli, and Mubarak Shah. Don't pour cereal into coffee: Differentiable temporal logic for temporal action segmentation. *NeurIPS*, 2022.

[25] Nadine Behrmann, S Alireza Golestaneh, Zico Kolter, Juergen Gall, and Mehdi Noroozi. Unified fully and timestamp supervised temporal action segmentation via sequence to sequence translation. In *ECCV*, 2022.

[26] Emad Bahrami, Gianpiero Francesca, and Juergen Gall. How much temporal long-term context is needed for action segmentation? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10351–10361, 2023.

[27] Daochang Liu, Qiyue Li, AnhDung Dinh, Tingting Jiang, Mubarak Shah, and Chang Xu. Diffusion action segmentation. *arXiv*, 2023.