IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 15 June 2024, accepted 19 August 2024, date of publication 29 August 2024, date of current version 18 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3451503

RESEARCH ARTICLE

Action Progression Networks for Temporal Action Detection in Videos

CHONG-KAI LU[1], MAN-WAI MAK[1], (Senior Member, IEEE), RUIMIN LI[2], (Member, IEEE), ZHERU CHI[1], (Member, IEEE), AND HONG FU[3]

[1]Department of Electrical and Electronical Engineering, The Hong Kong Polytechnic University, Hong Kong, China
[2]Academy of Advanced Interdisciplinary Research, Xidian University, Xi'an, Shaanxi 710126, China
[3]Department of Mathematics and Information Technology, The Education University of Hong Kong, Hong Kong, China

Corresponding author: Chong-Kai Lu (chong-kai.lu@polyu.edu.hk)

This work was supported in part by The Hong Kong Polytechnic University under Grant R-ZDF9 and in part by the RGC of Hong Kong under Grant 15228223.

ABSTRACT This study introduces an innovative Temporal Action Detection (TAD) model that is distinguished by its lightweight structure and capability for end-to-end training, delivering competitive performance. Traditional TAD approaches often rely on pre-trained models for feature extraction, compromising on end-to-end training for efficiency, yet encounter challenges due to misalignment with tasks and data shifts. Our method addresses these challenges by processing untrimmed videos on a snippet basis, facilitating a snippet-level TAD model that is trained end-to-end. Central to our approach is a novel frame-level label, termed ''action progressions,'' designed to encode temporal localization information. The prediction of action progressions not only enables our snippet-level model to incorporate temporal information effectively but also introduces a granular temporal encoding for the evolution of actions, enhancing the precision of detection. Beyond a streamlined pipeline, our model introduces several novel capabilities: 1) It directly learns from raw videos, unlike prevalent TAD methods that depend on frozen, pre-trained feature extraction models; 2) It is flexible for training with trimmed and untrimmed videos; 3) It is the first TAD model to avoid the detection of incomplete actions; and 4) It can accurately detect long-lasting actions or those with clear evolutionary patterns. Utilizing these advantages, our model achieves commendable performance on benchmark datasets, securing averaged mean Average Precision (mAP) scores of 54.8%, 30.5%, and 78.7% on THUMOS14, ActivityNet-1.3, and DFMAD, respectively.

INDEX TERMS Action recognition, temporal action detection, video analysis.

I. INTRODUCTION

With the advancement of communication technology, video has become the primary medium for consumer Internet traffic and its proportion is continually increasing [1]. The rapid growth of video content has led to a growing demand for powerful AI techniques for automatic video understanding, particularly human action recognition (HAR). The HAR community has primarily focused on the task of action classification, which aims to classify actions in videos that have been trimmed to contain only action content [2], [3], [4]. However, videos are usually unconstrained in practice, containing a significant amount of temporal background content. Consequently, the temporal action detection (TAD) task has drawn increasing attention. TAD requires detecting actions from untrimmed videos in terms of both the categories (classification) and the temporal boundaries (localization). It's challenging because an untrimmed video could contain multiple action instances from different classes and substantial temporal background, and the duration of actions varies significantly.

Deep neural networks (DNN) are known to be powerful feature learners [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]. However, directly inputting a raw untrimmed video into a DNN can result in an excessively large computation and memory footprint for a single pass, potentially leading to issues such as out-of-memory error. To overcome this problem, some studies [16], [17], [18], [19] employ the

The associate editor coordinating the review of this manuscript and approving it for publication was Davide Patti.

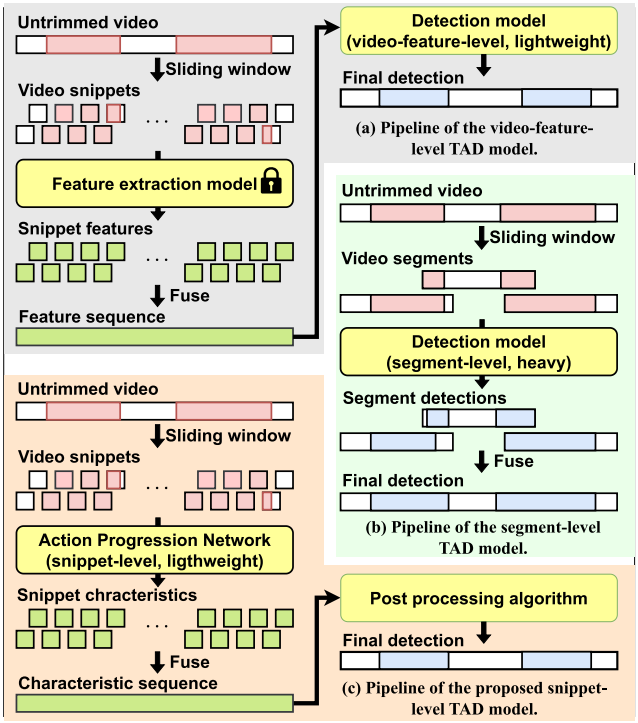

FIGURE 1. Comparison of the proposed and conventional pipelines. (a) The prevalent pipeline utilizes a video feature extraction model, hindering the end-to-end training. (b) The conventional end-to-end TAD pipelines detect actions in video segments independently, suffering from a tension between detection precision and model complexity. (c) The proposed pipeline adopts a lightweight and end-to-end trainable model to solve the TAD problem.

intuitive "divide and conquer" paradigm to crop the lengthy untrimmed videos into shorter video segments, and then perform detection inside each video segments independently and aggregate the results, as shown in Fig. 1(b). However, the *segmenting-based* approach is merely a compromise that may still face out-of-memory issues in extreme scenarios. Moreover, it presents a notable trade-off between computational requirements and high-quality detection [20], [21].

Another prevalent strategy involves compressing untrimmed videos through *pre-extraction* of features [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34]. This kind of method typically employs an off-the-shelf video feature extraction model, such as an I3D pre-trained on Kinetics [35], across the temporal dimension of untrimmed videos for spatial and short-term temporal feature extraction. Subsequently, a TAD neural network is trained on these extracted feature sequences for long-term temporal modeling, as shown in Fig. 1(a). While this approach can mitigate some computational challenges, its effectiveness heavily relies on the quality of the feature extraction model. This reliance limits the flexibility and detection accuracy, due to task misalignment and discrepancy between the pre-training and downstream datasets [36].

Departing from the two paradigms previously discussed, our approach introduces a streamlined TAD framework that processes untrimmed videos on a snippet-by-snippet basis, effectively addressing the inherent complexity of untrimmed videos, as shown in Fig. 1(c). This strategy enables us to harness the benefits of both the segmenting-based and feature pre-extraction methods, facilitating end-to-end training within a lightweight model framework. Noteworthy, several existing TAD techniques [23], [37], [38] employ the snippet-wise classification scores to locate actions. However, these detection results typically serve as preliminary temporal action proposals due to their lack of comprehensive temporal modeling.

To facilitate the learning of temporal information in a snippet-level TAD model, we introduce the action progressions as a snippet-level concept that encodes temporal context information into video snippets. Action progression indicates the progression of an action, enabling a granular understanding of action evolution over time. Action progressions could not only enhance temporal modeling for our snippet-level TAD model but also offer a detailed, continuous encoding of action stages, in contrast to the discrete, three-stage approach typically found in boundary-regression methods [27], [28], [30], [33]. This continuous and fine-grained representation allows for accurate action localization, leveraging the demonstrated effectiveness of encoding action evolution patterns [39], [40], [41]. By adopting action progressions, our model addresses the limitations of previous methods, providing a lightweight end-to-end TAD model for high-precision temporal action detection.

Leveraging the concept of action progressions, we developed the Action Progression Networks (APN), a tailored, snippet-level model that is end-to-end trainable. Fig. 2 illustrates the proposed framework and demonstrates how an APN addresses the TAD problem. Initially, each action frame is assigned a progression label based on its chronological position within the action, as depicted in Fig. 2(a). An APN is then trained to predict the action frame's action progression and category label, as shown in Fig. 2(b). For inference, the framework effectively pinpoints temporal boundaries by identifying action progression sequences that linearly escalate from 0% to 100%, illustrated in Fig. 2(c). Regarding classification, the framework simply averages the predicted class scores across frames within the detected action boundaries.

The detection precision of our framework is intrinsically linked to the precision in predicting the action progressions. To enhance the APN's precision, we employ two key techniques. First, we treat the prediction of action progressions as an *ordinal regression* problem [42], evaluating several advanced methods for this purpose. Among these, we identify the *threshold model* [43] as the superior encoding scheme, primarily because it accommodates the diverse spatial and temporal patterns of actions. Second, we preprocess each input frame into a "local video clip", refer to Fig. 2(b), encompassing the target frame and its immediate neighbors. This adjustment allows for short-term temporal modeling, significantly reducing prediction errors. These strategic

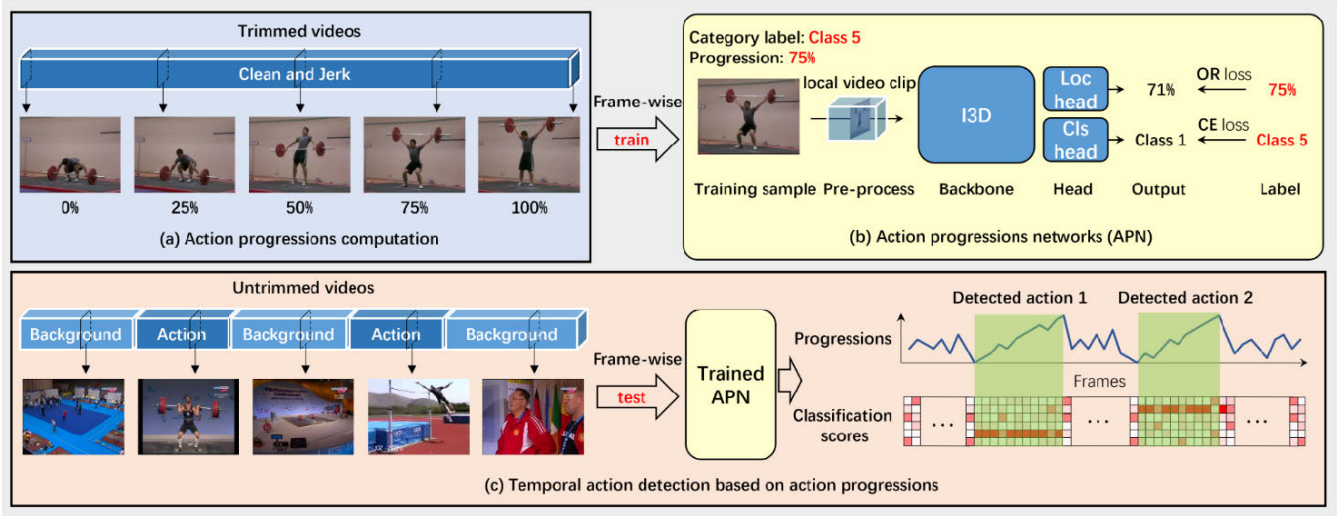

FIGURE 2. Overview of our framework. (a) Action progression labels (in %) are first generated for each action frame, based on the relative temporal location of the frames within the action. (b) An artificial neural network, dubbed action progression network, is then trained to predict the action progressions and categories from action frames. Each action frame is extended to include its neighboring frames before being fed to the model. (c) Using the trained APN, we predict the action progression sequence and classification score matrix of a test video. A profile-matching algorithm is then applied to the predictions to detect the temporal action boundaries and classification scores. OR: Ordinal Regression. CE: Cross Entropy.

improvements, as evidenced in our results in Table 2 and Table 3, collectively contribute to reduced action progression prediction error and heightened detection precision.

Our evaluation of the APN span three datasets, including THUMOS14 [44], ActivityNet-1.3 [45], and the newly introduced DFMAD.[1] The APN demonstrates competitive performance on three datasets. For example, it achieves competitive accuracy (54.8% vs. 59.6% on THUMOS14) with ten times less GPU memory consumption than Plus-TAD [46], the leading end-to-end trainable TAD model. The performance of APN on DFMAD was notably superior, outpacing all existing TAD models in terms of both detection accuracy and computational efficiency. Specifically, a variant of APN attains an mAP of 85.7% at 6277 FPS, markedly outperforming the ActionFormer [34], which achieves an mAP of 84.0% at 198 FPS. It is noteworthy that the comparison between the APN and other TAD models cannot be exactly fair as the APN is trained with action frames only, which could be categorized as weak supervision according to [47]. Moreover, APN's distinct ability to eschew the detection of incomplete actions, see Fig. 6, represents a novel contribution to temporal action detection, marking it as potentially the first model to exhibit this capability.

Overall, we propose quantifying action evolution for temporal action detection and make *three contributions*: (1) We introduce action progressions as a novel supervision signal, enabling the snippet-level TAD model to learn temporal information and precisely encode action evolution. (2) We design an effective yet straightforward TAD framework characterized by its lightweight structure and capability for end-to-end training exclusively on action frames. (3) The proposed framework demonstrates outstanding performance in detecting actions of long duration or with discernible evolution patterns. (4) To the best of our knowledge, the proposed framework is the first TAD method that could explicitly avoid detecting incomplete actions.

[1]Code will be available at https://github.com/makecent/mmtad

II. RELATED WORK

A. TEMPORAL ACTION DETECTION

Temporal action detection (TAD) is a meaningful yet challenging task. To avoid the extensive cost of annotations, some approaches [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62] tackle this task in a weakly supervised manner by leveraging easily available labels (such as video-level categorical labels). Although weakly supervised methods reduce the burden of labeling temporal annotations, their detection performance is not satisfactory. On the other hand, fully supervised TAD has higher interpretability and performance but requires annotations of both the temporal boundaries and categories of actions in untrimmed videos. Our work, utilizing action-content-only supervision, aligns with the weakly supervised category as discussed by [47]. Nonetheless, we position our approach within the fully supervised domain, emphasizing the necessity for manual annotations of temporal action boundaries. Despite this, our method leverages the benefits of action-content-only supervision: it allows for training on extensive trimmed video datasets [47], [63] and facilitates adaptation to unseen temporal backgrounds compared with models that were biased by the temporal background in the training dataset.

From the perspective of pipeline structures, existing TAD methods commonly adopt multi-stage designs. One prevalent approach, referred to as feature *pre-extraction* [36], involves

leveraging off-the-shelf video feature extraction models, such as sliding a pre-trained I3D model across the video's temporal axis to transform the untrimmed videos into feature sequences that are more complexity manageable. [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34]. Further, stages such as temporal proposal generation [28], action classification [50], and refinement [64] often function as separate components, leading to diverse multi-stage configurations. Though multi-stage designs have their benefits, they can also introduce complexity into the detection workflow and potentially limit precision. Notably, multi-stage TAD models often lack end-to-end training capabilities, involving multiple independently trained DNNs, which may impede the learning of robust representations. For instance, TAD models relying on feature pre-extraction could face substantial performance limitations when there are pronounced domain shifts between the pre-training and target TAD datasets. In contrast, the proposed APN is trained end-to-end, similar to [26], [46], [65], [66], and [67] but using a snippet-wise processing strategy to enable a lightweight TAD model whose complexity is video duration-invariant.

From the perspective of encoding action boundaries, TAD models often draw inspiration from the object detection frameworks, employing anchor-based, anchor-free, or query-based mechanisms. These approaches aim to identify temporal action boundaries by predicting offsets to a predefined set of 1-D anchors (for anchor-based [24], [25], [26], [31], [65], [68], [69], [70], [71], [72] and query-based [67], [73], [74], [75], [76], [77], [78] methods) or points (for anchor-free methods, e.g., [34], [79], [80]). Their effectiveness largely hinges on the initial reference objects' quality, demanding prior knowledge of action distributions in videos. In contrast, bottom-up approaches determine action boundaries through snippet-wise characteristics. A naive bottom-up strategy would be applying a grouping algorithm on snippet-wise classification scores [23], [37], [66], which lacks temporal modeling. Thus, boundary-regression methods [27], [28], [30], [33], [71] opt to predict boundaries scores alongside classification scores, offering fine-grained, snippet-level localization through dense prediction [66], and demonstrating competitive outcomes. The APN introduces a novel bottom-up TAD approach by (a) introducing the action progression as snippet-level characteristics, (b) predicting snippet-wise characteristics independently, and (c) deliberately avoiding learning from temporal backgrounds.

B. ACTION PROGRESSION

The "action progression" or similar concepts have been discussed in previous work [39], [40], [81], [82], [83], [84], [85], [86], [87]. We would like to claim that "action progression" is not a strictly defined terminology [82] and our usage significantly diverges from others. The most related work would be LAP-Net [41], LSTM-TAD [39], and PR-RNN [40], all of which leverage action progression prediction for detection purpose. LAP-Net utilizes action progressions to determine the sampling range of future features for online action detection, LSTM-TAD implicitly encodes them as non-decreasing detection scores for regularization, and PR-RNN treats them as auxiliary clues for linking action boxes. In contrast, our framework uses explicitly encoded action progressions as the primary clue for temporal action localization through a specialized profile-matching algorithm. Moreover, while the cited works rely on long-term temporal memory, our APN uniquely forecasts action progressions for each snippet independently, facilitating an efficient snippet-level TAD model.

C. ORDINAL REGRESSION

In our framework, predicting action progressions from video frames is framed as an ordinal regression problem. A good survey about ordinal regression methods can be found in [42]. According to the proposed taxonomy in [42], there are five main ordinal regression methods. Among them, three are naive methods: *regression*, *nominal classification*, and *cost-sensitive classification*, and two are advanced methods – *ordinal binary decomposition* and *threshold models*. We have investigated all of these methods for the APN. The *regression* and *nominal classification* are implemented by ourselves, whereas the others are implemented with references to the existing work. In particular, we implement *ordinal binary decomposition* by referring to [88], *cost-sensitive classification* by referring to [89], and *threshold models* by referring to [43].

This paper extends our previous work [90] in several aspects. *First,* we use a stack of adjacent frames instead of a single frame as input to the APN, which improves the accuracy of action progression prediction. *Second,* we employ a single model for all action classes rather than training one model for each class. *Third,* We utilize more advanced loss functions and neural network designs. *Fourth,* we conduct a more extensive range of experiments and analyses, such as evaluating the APN on the THUMOS14 dataset to demonstrate its superior performance.

III. ACTION PROGRESSION NETWORKS

In this work, a neural network is trained to tackle a pretext task – *predicting the action evolution process in videos*, rather than detect actions in untrimmed videos directly. The action localization is accomplished then by analyzing the patterns of the predicted action evolution with a simple algorithm. Since the algorithm is independent of the neural networks, its description is left to Section IV. This section describes how we train a network to recognize the action evolution in videos. Specifically, we first explain how the action progressions are quantitatively encoded as frame-level regression labels for action evolution. After that, we present the architecture of the network that predicts action progressions and categories from action frames, *i.e.*, the APN. Finally, we give the details of five ordinal regression methods that have been investigated for encoding the action progressions.

A. ACTION PROGRESSIONS IN VIDEOS

To encode a complete action evolution process into fine-grained stages and take their ordering into account, we quantify a complete temporal action process on a numerical scale. As shown in Fig. 2(a), we label each time point in the action with a real number in the range 0% to 100%. We call these regressive characteristics of action evolution as *action progressions*. The ground truths of action progressions can be derived without extra human labor on the annotation. Specifically, we derive the action progression label for every action frame based on their relative chronological positions in the action instance, which can be easily determined based on the original TAD annotations.

Formally, given an action instance comprising l frames $A = \{f_1, f_2, \ldots, f_l\}$, we assign every frame f_τ with an action progression label p_τ using the following formula:

$$p_\tau = \left\lfloor K \frac{\tau}{l} \right\rceil, \tag{1}$$

where $\lfloor \cdot \rceil$ denotes the rounding-to-the-nearest integer, and K is a constant controlling the number of ranks used to dissect the action evolution. We observed an improvement in detection performance with a larger K. However, the gains diminish after surpassing a certain optimal point, as illustrated in Fig. 4. We set K to 100 by default, resulting in 101 action progressions, namely 0%, 1%, ..., 100%. Partitioning an action instance into 100 intervals is sufficiently fine for capturing most of the temporal structures of actions. As only action frames have action progression labels, the temporal backgrounds in the untrimmed videos are ignored during training.

Noteworthy, for the action progression derived by Eq. (1) to be reasonable and informative, some conditions must be met. That is, the action should have an exact, rich, and non-repetitive evolution pattern, which we call *progressive actions*. In other words, our method is not suitable for detecting non-progressive actions, e.g., the indeterminate actions like *painting*, the ephemeral actions like *hugging*, and the repetitive actions like *walking*. The reward is that our work can detect progressive actions precisely, as we will see in Fig. 5.

B. NEURAL NETWORK ARCHITECTURE

We train a DNN named APN to predict the action progressions and categories from video frames. Overall, an APN takes as input a short-term video snippet, which could be as short as a single video frame, and outputs a predicted action progression and a vector of predicted classification scores, as shown in Fig. 2(b).

1) INPUT

Although the APN aims to predict action progressions of each frame, the prediction will be more accurate if the APN receives contextual frames (instead of a single frame) as input. Formally, the input volume I_τ of a frame $f_\tau \in \mathbb{R}^{D\times W\times H}$ at time τ is constructed as follows:

$$I_\tau = \{f_{\tau-(l-1)d}, \ldots, f_{\tau-d}, f_\tau, f_{\tau+d}, \ldots, f_{\tau+ld}\}, \tag{2}$$

where d is the temporal stride (in frames) between the adjacent frames of the $L = 2l$ frames contained in the volume, and D, W, and H are the dimensions of the channels, width, and height of the videos, respectively. We observed that increasing the duration (L) and resolution (d) of the input context window always benefits the detection precision for all action types. However, setting them larger than their sweet spots cannot improve the detection precision further. Also, the "sweet spots" of different actions vary. We notice two simple rules: (1) for long actions, a long context window (L) is required; (2) for actions with complex temporal patterns, a high temporal resolution is required (smaller d). We find that $L = 32$ and $d = 4$ are suitable for almost all types of actions in the three datasets, as we will discuss in Section V-C.

2) BACKBONE

We study the I3D [35] and the ResNet-50 [11] as the APN backbone to extract features. The ResNet-50 is used when the input is a single 2-D image, i.e., $I_\tau = f_\tau$; otherwise the I3D is used. The output features of the I3D and the ResNet-50 backbone after average pooling are of size 1024 and 2048, respectively.

It is worth noting that the choice of backbones for the APN is flexible. Thus, it is possible to incorporate the APN with any other advanced models that are good at capturing video dynamics, e.g., we find that using SlowFast [91] and TimeSformer [92] as the backbone of APN result in higher detection precision than I3D. Here, we only investigate the commonly used backbones because our innovation lies in the workflow rather than the network architecture.

3) SIBLING HEADS

The APN has two heads, the classification head and the localization head, referring to as *Cls* and *Loc* heads in Fig. 2(b). The classification head is a fully connected layer outputting the classification scores trained with cross-entropy loss. The localization head is designed to output the action progressions. We investigated five different ordinal regression methods for encoding the action progressions, resulting in five types of localization heads. The details of them are described in Section III-C. Formally, the sibling heads output the predicted classification scores, $\hat{\boldsymbol{s}} \in \mathbb{R}^C \cap (0, 1)$, and a predicted action progression $\hat{p} \in \mathbb{R} \cap [0, K]$, where C and K are the number and action classes and progression ranks, respectively.

C. ORDINAL REGRESSION HEADS

The action progressions are represented as a number of ordered ranks, as derived by Eq. (1), for a fine-grained encoding of the action evolution. An intuitive scheme would be taking the prediction of action progressions as a regression problem. Despite the simplicity of the regression encoding,

it may offer a sub-optimal performance [42], [43], [88], [89], [93] as the regression values set a fixed and equal distance between the adjacent progression ranks. The regression encoding could work well on simple and short-lasting actions, *e.g.*, the model may somehow connect the progressions values with the height of the barbell for action "Clean and Jerk", see *e.g.*, Fig. 2 (a). However, it could encounter issues when comes to complex activities that consist of multiple distinct phrases, known as the *non-stationary* problem [88]. For example, the action "High Jump" has a "run-jump-land" temporal procedure. Thus it evolves in terms of different characteristics at different phases, which could not be well-represented by a homogeneous linear regression range.

Since the prediction precision of action progressions significantly affects the proposed APN's detection performance, to find a proper setting, we investigate five representative ordinal regression methods for encoding action progressions and building the localization head of APN, including regression, nominal classification, binary decomposition [88], cost-sensitive classification [89], and the threshold model [43]. In the following, we suppose that an action frame f_τ with the ground truth action progression label $p \in \mathbb{Z} \cap [0, K]$ is input to the APN.

1) REGRESSION

The regression head is an intuitive baseline that outputs an action progression without encoding except for a normalization operation. The mean absolute error is used for computing the regression loss:

$$\mathcal{L}_{\text{reg}}(o, p) = \left| \mathcal{G}(o) - \frac{p}{K} \right|, \tag{3}$$

where $o \in \mathbb{R}$ is the head's output and $\mathcal{G}$ is a clamp function that normalizes the output value to the range [0, 1]. During inferencing, the predicted rank $\hat{p}$ is decoded from the inference output $\hat{o}$ based on the following formula:

$$\hat{p}_{\text{reg}} = K\mathcal{G}(\hat{o}). \tag{4}$$

2) NOMINAL CLASSIFICATION

Strictly speaking, nominal classification is not an ordinal regression method because it ignores the continuity between the ranks. We put it here as another baseline for consistency. The head simply predicts the action progression as solving a classification problem. Because there are $K + 1$ ranks, the output of this head is a vector, $\boldsymbol{o} \in \mathbb{R}^{K+1}$. The widely used classification loss, cross-entropy (CE), is adopted:

$$\mathcal{L}_{\text{ncl}}(\boldsymbol{o}, p) = -\log(\text{softmax}(\boldsymbol{o})_p) = -\log \frac{\exp(o_p)}{\sum_{j=0}^{K} \exp(o_j)}. \tag{5}$$

As for inference, instead of using the conventional argmax function, we compute the expected values to decode the prediction result from the head's outputs:

$$\hat{p}_{\text{ncl}} = \sum_{j=0}^{K} j \times \text{softmax}(\hat{\boldsymbol{o}})_j. \tag{6}$$

Although the nominal classification head does not have the *non-stationary* problem, it ignores the rank order. For example, the classification losses of predicting rank-1 as rank-2 and predicting rank-1 as rank-99 have the same magnitude, which is not desirable.

3) COST-SENSITIVE CLASSIFICATION

This head shares the same output format and inference formula as the nominal classification head, *i.e.*, $\boldsymbol{o} \in \mathbb{R}^{K+1}$, Eq. (6). The difference is that it computes loss with soft labels, instead of the one-hot labels, to encode the ordinal relationship among adjacent classes. In our experiment, we used the following formula to convert the original regression label p to its soft label $\boldsymbol{q} \in \mathbb{R}^{K+1}$:

$$\boldsymbol{q} = [q_j]_{j=0}^{K}, \text{ where } q_j = \frac{\exp(-\sqrt{|j-p|})}{\sum_{k=0}^{K} \exp(-\sqrt{|k-p|})}. \tag{7}$$

The above equation Eq. (7) assigns lower probabilities to ranks as the distance between them to the target rank increases. Following [89], we used the Kullback–Leibler divergence to compute the loss of an output $\boldsymbol{o}$ when the soft label is $\boldsymbol{q}$:

$$\mathcal{L}_{\text{cst}}(\boldsymbol{o}, \boldsymbol{q}) = \sum_{j=0}^{K} q_j \log \frac{q_j}{\text{softmax}(\boldsymbol{o})_j}. \tag{8}$$

The derivation of soft labels is tricky and hard to determine without prior knowledge of the datasets. In other words, using a simple and unified formula Eq. (7) to encode the ordering among action progressions is not optimal.

4) BINARY DECOMPOSITION

This head decomposes the ordinal regression problem into multiple binary classification problems – "Is the progression p greater than j?", where j takes value from all possible ranks $[0, 1, \ldots, K]$. Concretely, the output of the binary decomposition head is $\boldsymbol{o} \in \mathbb{R}^{K\times 2}$. The CE loss on each binary classifier is computed, and their mean is taken as the final loss:

$$\mathcal{L}_{\text{bdc}}(\boldsymbol{o}, p) = -\sum_{j=1}^{K} [\![p \geq j]\!] \mathcal{F}(\boldsymbol{o}_j)_1 + [\![p < j]\!] \mathcal{F}(\boldsymbol{o}_j)_2, \tag{9}$$

where the subscripts 1 and 2 are the indexes on the nodes of each binary classifier. $[\![\cdot]\!]$ is a Boolean test, which is equal to 1 if the inner condition is true, and 0 otherwise. $\mathcal{F}$ is the log-softmax activation function. The inference formula is:

$$\hat{p}_{\text{bdc}} = \sum_{j=1}^{K} [\![\text{softmax}(\hat{\boldsymbol{o}}_j)_1 \geq 0.5]\!]. \tag{10}$$

Binary decomposition converts a regression problem into a series of binary classification sub-problems. This design can help alleviate the *non-stationary* problem. However, the decomposition head does not have any limit on the outputs of different ranks, resulting in the *inconsistency* problem [43]. For example, it may answer *Yes* to the sub-problem "Is $\hat{p} \geq 20$?", and meanwhile answer *No* to the sub-problem "Is $\hat{p} \geq 10$?".

5) THRESHOLD MODEL

This head is created by simply adding K learnable bias terms to the conventional regression head. Each biased result is then transformed, by a sigmoid function, to the probability of one binary classification problem (just like the binary decomposition). Notably, unlike binary decomposition, the threshold model performs binary classification by simply outputting the probability of *Yes* to the question "Is $\hat{p} \geq$ rank j?". In this way, the binary cross-entropy (BCE) (rather than CE) is applied to each element of the output $\boldsymbol{o} \in \mathbb{R}^K$ to compute the loss:

$$\mathcal{L}_{\text{thr}}(\boldsymbol{o}, p) = -\sum_{j=1}^{K} [\![p \geq j]\!] \log \mathcal{A}(o_j) + [\![p < j]\!] \log(1 - \mathcal{A}(o_j)), \tag{11}$$

where $[\![\cdot]\!]$ is the Boolean test and $\mathcal{A}$ is the sigmoid activation function. The inference formula is:

$$\hat{p}_{\text{thr}} = \sum_{j=1}^{K} [\![\mathcal{A}(\hat{o}_j) \geq 0.5]\!]. \tag{12}$$

The threshold modeling uses different thresholds (bias terms) to encode different degrees of dissimilarity between the samples of successive ranks, thus helping to solve the non-stationary problem. Besides, the threshold model can naturally learn bias that decreases monotonically from lower ranks to higher ranks, solving the *inconsistency* problem mentioned in the binary decomposition head.

IV. TEMPORAL ACTION DETECTION WITH APN

With the APN framework, the action progressions of video frames can be learned and predicted. However, there remains the question of how to use the predicted action progressions to derive the temporal boundaries of actions in the videos. In this section, we describe how to apply a trained APN for temporal action detection. An overview of the detection workflow is displayed in Fig. 2(c).

First, given an untrimmed test video with T frames, we present the frames to the trained APN sequentially to predict a sequence of action progressions $\hat{\boldsymbol{p}} \in \mathbb{R}^T$ and a class score matrix $\hat{\boldsymbol{S}} \in \mathbb{R}^{C \times T}$:

$$\begin{aligned} \hat{\boldsymbol{p}} &= \begin{bmatrix} \hat{p}_1 & \hat{p}_2 & \dots & \hat{p}_\tau & \dots & \hat{p}_T \end{bmatrix} \in \mathbb{R}^T, \\ \hat{\boldsymbol{S}} &= \begin{bmatrix} \hat{\boldsymbol{s}}_1 & \hat{\boldsymbol{s}}_2 & \dots & \hat{\boldsymbol{s}}_\tau & \dots & \hat{\boldsymbol{s}}_T \end{bmatrix} \in \mathbb{R}^{C \times T}, \end{aligned} \tag{13}$$

where C is the number of action classes, $\hat{p}_\tau \in \mathbb{R} \cap [0, K]$ and $\hat{\boldsymbol{s}}_\tau \in \mathbb{R}^C \cap (0, 1)$ are the predicted action progression and class scores of the τ-th frame, respectively. After that, we applied a *profile-matching* algorithm to the progression sequence $\hat{\boldsymbol{p}}$ and class score matrix $\hat{\boldsymbol{S}}$.

According to the definition of action progression, if a subsequence in the predicted progression sequence $\hat{\boldsymbol{p}}$ close to an arithmetic sequence starting with 0 and ending with K, it is likely to cover an action instance. Algorithm 1 is designed based on this principle.

Algorithm 1 Action Detection on Progression Sequence

Input:
Predicted action progression sequence: $\hat{\boldsymbol{p}} \in \mathbb{R}^T$;
Predicted classification scores: $\hat{\boldsymbol{S}} \in \mathbb{R}^{C \times T}$;
Parameters:
Minimum action length: T_{len}
Maximum progression value for start frame: P_{start}
Minimum progression value for end frame: P_{end}
Minimum IoU threshold in NMS: η_{IoU}
Initialization:
Number of frames: T
Number of action classes: C
Detected actions: $\Psi = \{\Psi_1 = \varnothing, \dots, \Psi_C = \varnothing\}$
$\boldsymbol{starts} = \{\tau \mid \hat{p}_\tau < \mathrm{P}_{start}\}$
$\boldsymbol{ends} = \{\tau \mid \hat{p}_\tau > \mathrm{P}_{end}\}$

```
for start in starts do
    for end in ends do
        actionLen = end − start + 1
        if actionLen > T_len then
            candidate = [p̂_τ]_{τ=start}^{end}
            oracle = [Kτ/actionLen]_{τ=0}^{actionLen}
            loc_score = E(candidate, oracle)
            if loc_score > 0 then
                cls_scores = Σ_{i=start}^{end} ŝ_i/actionLen
                for j in {1, 2, ..., C} do
                    cls_score = cls_scores_j
                    confidence = loc_score × cls_score
                    Ψ_i ∪ {(start, end, confidence)}
for i in {1, 2, ..., C} do
    Ψ_j = NMS(Ψ_j, η_IoU)
```

Output:
Detected actions: Ψ

In Algorithm 1, NMS stands for the non-maximum-suppression [94], which is used to reduce redundant detections; $\mathcal{E}(\cdot)$ reflects the similarity between two profiles of the same length, and it is used for evaluating the localization scores (confidences). In practice, we compute $\mathcal{E}(\cdot)$ based on the Euclidean distance as follow:

$$\mathcal{E}(\boldsymbol{v_1}, \boldsymbol{v_2}) = 1 - \frac{\text{MSE}(\boldsymbol{v_1}, \boldsymbol{v_2})}{1666.66}, \tag{14}$$

where 1666.66 is the mean square error (MSE) of random pairs when $K = 100$ (See Appendix). Ψ_j stands for the detected results of action class j, which consists of tuples (start, end, confidence) representing one detected action

instance in terms of starting time, ending time, and the confidence score, respectively.

The current version of APN does not have a pre-processing step to filter out or ignore non-progressive actions. It simply takes all types of actions as progressive, and therefore, it may produce incorrect detections for non-progressive actions. However, for videos containing mainly progressive actions, such as the video in the DFMAD dataset, the APN can detect progressive actions precisely and quickly, as demonstrated in Table 6. In short, we recommend using APNs to detect progressive actions.

V. EXPERIMENTS

In this section, we first introduce the datasets utilized for evaluation and the metrics used to assess performance. Following this, we detail the implementation specifics. Subsequently, we present an empirical analysis of the APN's performance, including comparisons with state-of-the-art methods, and conclude with visualizations of detection outcomes.

A. DATASETS AND PERFORMANCE METRICS

1) DATASETS

THUMOS14 serves as a benchmark dataset for temporal action detection (TAD), featuring videos across 20 human action classes against temporal backgrounds. The dataset is comprised of 2,756 trimmed videos for training, 200 untrimmed videos for validation, and 212 untrimmed videos for testing, totaling 2,756, 3,007, and 3,358 annotated action instances across the training, validation, and test splits, respectively. Unlike conventional TAD methods, which are typically trained only on the validation split due to their dependency on learning temporal backgrounds, our APN utilizes action frames exclusively. This enables the use of both training and validation splits for model training, leveraging the comprehensive annotations available.

ActivityNet-1.3 consists of 10,024 training videos (15,410 instances) and 4,926 validation videos (7,654 instances), spanning over 200 action classes. In contrast to THUMOS14, ActivityNet-1.3 often features videos with a single, extensively annotated action instance, presenting a different set of challenges for TAD.

DFMAD is a part of the DCD dataset [95] aimed at children behavior research. It includes 63 untrimmed, long-duration videos from a first-person perspective. Of these, 50 videos are designated for training and 13 for testing. Each video encompasses eight complete (4+2+2) and five incomplete (2+2+1) action instances across three action classes. DFMAD is characterized by (1) precise annotations, exclusively marking complete action instances, distinguishing it from the more common practice in THUMOS14 and ActivityNet of annotating incomplete actions or sequences of multiple action instances, and (2) the extended duration of both videos and actions, averaging 676 seconds and 31 seconds, respectively. These features make DFMAD particularly suited for evaluating the efficacy of TAD methodologies in complex, research scenarios.

In Fig. 3, sample images from the three datasets are shown. It is evident that the videos have significant variety in terms of camera settings and content, making temporal action detection challenging.

2) METRICS

The evaluation of the APN framework's performance leverages several key metrics, detailed as follows:

Mean Absolute Error (MAE): is employed to assess the APN's accuracy in predicting action progressions. The MAE values are normalized to a [0, 100] range to ensure consistency with the action progression labels' scale. For instance, if the progression labels are divided into $K = 50$ categories, as specified in Eq. (1), the MAE is then scaled by a factor of 2. The normalization facilitates direct comparison and the baseline MAE is 25% of random predictions.

Top-1 Accuracy (Acc.) is used to measure the classification efficacy of the APN when the localization is correct. This measure provides insight into the APN's capability to correctly classify action instances within the detected temporal boundaries.

Mean Average Precision (mAP) is used to gauge the overarching performance of temporal action detection, incorporating both the aspects of correct classification and precise localization. mAP under different thresholds of the intersection of union (IoU) and their mean are measured and denoted as **mAP@IoU** and **Avg.**, respectively. Following the common practice, the IoU thresholds [0.3:0.1:0.9] are used for THUMOS14, [0.5:0.05:0.9] for ActivityNet-1.3, and [0.5:0.1:0.9] for DFMAD.

B. IMPLEMENTATION DETAILS

Because one of the innovations of this work is attributed to the novel TAD pipeline, we intentionally make a concise and essential framework with few hyper-parameters to tune.

Procedures. In each training iteration, an action frame is selected randomly as input to the APN, which then predicted the frame's action progression and category. For inference, 1,000 frames are uniformly sampled from the test video and handled by the trained APN in sequence, and the resulting predictions are processed by Algorithm 1 to determine the final action detections.

Pre-processing. Data augmentation techniques are not adopted. The input images are simply resized to a fixed resolution of 224×224 via interpolation. To incorporate more temporal information, each input frame to the APN is extended to cover its neighboring $L \times d$ frames. The boundary frames are repeated when $L \times d$ was beyond the temporal range of actions.

Pre-training. Parameters of the I3D and ResNet50 backbones are initialized using weights pre-trained on Kinetics-400 [35] and ImageNet [14], respectively, facilitating a robust feature extraction foundation.

FIGURE 3. Some randomly sampled images of THUMOS14, AcitivyNet-1.3, and DFMAD.

Two-streams. The APN is trained separately on RGB images and TVL1 optical flow data [96], [97], implementing a late fusion strategy to average classification scores and action progression predictions from both streams.

Optimization. Network parameters are optimized using the Adam optimizer [98] and a simple learning rate schedule is adopted: A fixed learning rate 1e-4 for training the APN of 10 epochs.

Profile-matching. In Algorithm 1, P_{start} and P_{end} were set to 20 and 80, respectively, and η_{IoU} was set at 0.4. T_{len} was configured to 60 frames for THUMOS14 and ActivityNet-1.3, and 600 frames for DFMAD, according to the distribution of action lengths.

All hyper-parameters are determined by conducting a grid search on a separate validation set for each dataset individually. We find that the optimal hyper-parameters for different datasets are similar, with the exception of T_{len}.

C. STUDY ON ACTION PROGRESSION NETWORKS

We carried out a series of experiments to identify an optimal configuration for the APN, validate the efficacy of our proposed techniques, and gain deeper insights into temporal action detection.

1) IMPACT OF END-TO-END TRAINING

We investigated the influence of end-to-end training. The outcomes, detailed in Table 1, reveal that fine-tuning the backbone initialized with weights from prior training significantly enhances detection capabilities across all evaluated methods. Besides, the improved performance when trained with additional trimmed videos validates the APN's unique advantage of being trainable on trimmed videos. This distinction also highlights the potential for further improving the APN by training on large-scale action classification datasets [99], [100].

TABLE 1. Impact of end-to-end training. FT: Fine-tuning the backbone, which is initialized with the weights pre-trained on Kinetics400.

Model	Metric	THUMOS14		DFMAD	
		w/ FT	w/o FT	w/ FT	w/o FT
APN	MAE	16.2	18.6	8.7	14.5
	Acc.	65.6%	59.1%	92.7%	82.2%
	mAP	40.8%	27.3%	72.5%	55.0%
APN (w/o trimmed)	mAP	38.4%	25.9%	-	-
PlusTAD	mAP	50.2%	25.8%	69.4%	43.1%
E2E-TadTR	mAP	45.3%	34.0%	62.9%	36.8%

2) COMPARISON OF ORDINAL REGRESSION METHODS

Five ordinal regression methods for encoding action progressions were investigated, refer to Section III-C for details, and the experimental results are summarized in Table 2. We see that (1) The detection performance (mAP) is sensitive to the precision of action progression prediction (MAE).

TABLE 2. Comparison of ordinal regression methods (Section III-C) for APN on the THUMOS14 Dataset.

Ordinal regression method	MAE	mAP
Nominal classification	17.88	36.8%
Regression	17.35	38.9%
Cost-sensitive classification	16.72	38.5%
Binary decomposition	16.46	39.9%
Threshold model	16.23	40.8%

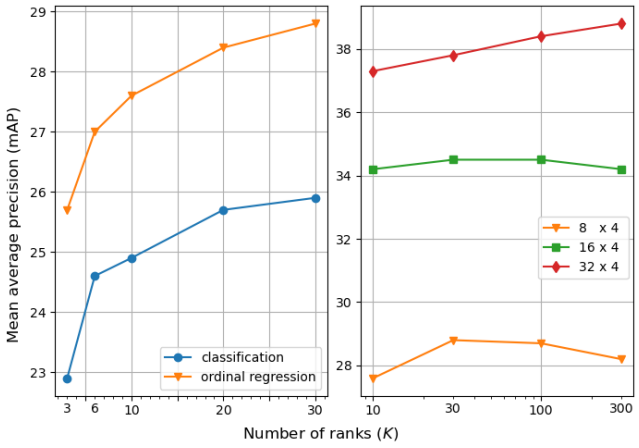

FIGURE 4. The mean Average Precision (mAP) on THUMOS14 against the number of ranks (K) in the action progressions using the threshold model as the regression method. *Left:* Increasing the number of ranks results in higher mAPs, and encoding with the ordinal regression outperforms the nominal classification. *Right:* The longer duration of the input ($L \times d$ in the legend) the better the performance. The mAP drops as the number of ranks rises to a certain value, which increases with the duration of the input.

(2) All methods outperform the *nominal classification*, demonstrating the importance of ordered encoding for action evolution phases. (3) The *threshold model* encoding achieved the best detection precision, and outperformed the naive regression by 1.9% (40.8% vs 38.9%). Since similar experimental results were obtained with ActivityNet-1.3 and DFMAD, the threshold model is concluded to be a reliable choice and was therefore used by default in other experiments.

3) IMPACT OF THE NUMBER OF RANKS

In addition to considering the order of evolution stages, another distinct characteristic of action progressions is that they are obtained by dissecting the action evolution into many stages, in contrast to the three stages used in the conventional approaches. We investigated the impact of the number of ranks, i.e., the value of K in Eq. (1). The experimental results are summarized in Fig. 4. We found that increasing the number ranks for the action progressions leads to higher mAP on temporal action detection. This result further validates the effectiveness of the proposed action progressions. Because of the diminishing return from increasing K, we set 100 as the default value of K in the subsequent experiments.

TABLE 3. Impact of the temporal context of the APN's input on performance. L: Number of frames. d: Stride. Loc: Localization. Cls: Classification.

Row	Input volume I_τ $L \times d$	Loc MAE	Cls Acc.	Detection mAP@0.5
0	1×1	19.9	54.9%	26.2%
1	16×1	16.2	65.6%	40.8%
2	16×2	14.6	71.3%	48.4%
3	16×4	14.2	76.3%	49.8%
4	16×8	13.6	77.6%	51.5%
5	32×4	13.1	77.2%	54.7%
6	64×2	12.8	76.9%	55.0%
7	128×1	14.0	69.5%	47.4%

4) IMPACT OF TEMPORAL CONTEXT OF THE INPUT

We tested the APNs at different temporal durations and resolutions. Specifically, we investigated various settings of the number of frames and strides (L and d in Eq. (2)) for the APNs' input. The experimental results are summarized in Table 3. We have several observations: (1) Comparing the performance of the same number of frames but different strides, thereby different temporal duration, e.g., Rows 1 to 4, we found that longer temporal duration improves the performance in terms of MAE, Acc., and mAP. This is expected because a longer temporal duration contains more information for action detection. (2) Comparing the performances of the same temporal duration but different strides, i.e., Rows 4 to 7, we found that the input volume 32×4 holds the best balance between performance and the computational cost. The low performance of 128×1 reflects that there is a lot of redundant information between adjacent frames in the video, and the effectiveness of the pre-trained weights is reduced when the downstream task uses a different temporal stride. We set the default value of L to 32 and d to 4 (i.e., one out of four from 128 consecutive frames) in the subsequent experiments unless stated otherwise.

5) PERFORMANCE ON DIFFERENT ACTIONS

As introduced in Section III-A, the APN is especially suitable for detecting *progressive* actions. To further investigate this, we present the APN's performance on different action classes in Fig. 5. There are several observations: (1) The APN outperforms SS-TAD [24] and SCNN [19], and all three methods perform very differently on detecting different actions. (2) The APN achieves high average precision when detecting actions that have clear evolution patterns, e.g., the APN achieves up to 85% precision on detecting action "High Jump", whereas the other two methods only achieve about 20%. Similar situations also occurred in action "Diving", "Golf Swing", and "Throw Discus" and all these actions have distinct evolution patterns in our manual observations. (3) On the contrary, the APN achieves a poor performance on indeterminate actions, e.g., "Billiards" (13.8%), and ephemeral actions, e.g., "Cricket Shot" (15.4%) and "Tennis Swing" (19.5%). Despite of superior performance

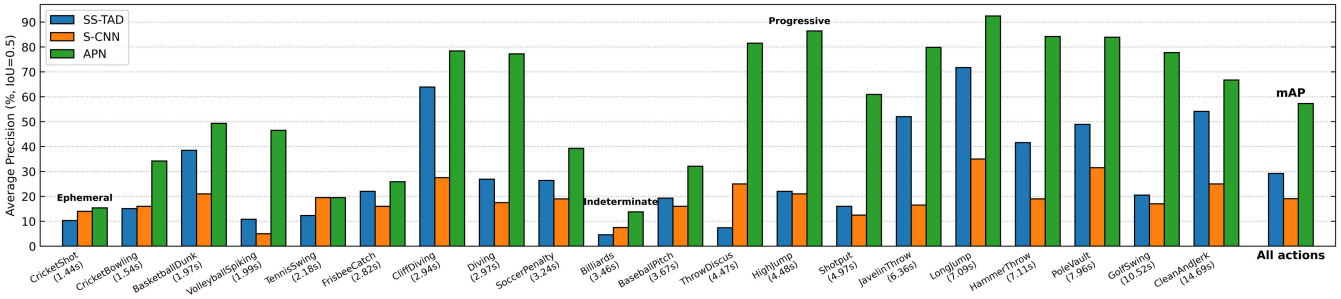

FIGURE 5. Average precision across various action classes in THUMOS14, presented in ascending order of the average durations. The proposed APN demonstrates high precision in progressive actions that have clear evolutionary patterns (e.g., *High Jump*), yet shows reduced effectiveness in actions with indeterminate characteristics (e.g., *Billiards*) and those that are brief in duration (e.g., *Cricket Shot*).

TABLE 4. Comparison with state-of-the-art methods on THUMOS14.

Methods	Feature	0.3	0.4	0.5	0.6	0.7	Avg.
Multi-Stage Training							
ABytes [47]	I3D	26.1	20.3	15.5	-	3.7	-
BMN [28]	TSN	56.0	47.4	38.8	29.7	20.5	38.5
G-TAD [101]	TSN	54.5	47.6	40.3	30.8	23.4	39.3
DBG [30]	TSN	57.8	49.4	39.8	30.2	21.7	39.8
P-GCN [102]	I3D	63.6	57.8	49.1	-	-	-
RTD-Net [74]	I3D	68.3	62.3	51.9	38.8	23.7	49.0
TadTR [75]	I3D	74.8	69.1	60.1	46.6	32.8	56.7
AFormer [34]	I3D	82.1	77.8	71.0	59.4	43.9	66.8
DiffTAD [78]	I3D	74.9	72.8	71.2	**62.9**	**58.5**	68.0
TriDet [103]	I3D	**83.6**	**80.1**	**72.9**	62.4	47.4	**69.3**
Self-DETR [77]	I3D	74.6	69.5	60.0	47.6	31.8	56.7
End-to-End training							
R-C3D [65]	C3D	54.5	51.5	44.8	35.6	28.9	43.1
PBRNet [71]	I3D	58.5	54.6	51.3	41.8	29.5	47.1
DaoTAD [72]	I3D	62.8	59.5	53.8	43.6	30.1	50.0
AFSD [80]	I3D	67.3	62.4	55.5	43.7	31.1	52.0
E2E-TadTR [67]	SF50	69.4	64.3	56.0	46.4	34.9	54.2
PlusTAD [46]	S50	**75.5**	**70.8**	**63.5**	**50.9**	**37.4**	**59.6**
APN (ours)	I3D	68.0	62.5	54.7	44.1	30.3	52.0
APN (ours)	S50	72.1	66.4	57.9	47.1	30.4	54.8

TABLE 5. Comparison with state-of-the-art methods on ActivityNet.

Methods	Feature	0.5	0.75	0.95	Avg.
Multi-Stage Training					
BMN [28]	TSN	50.1	34.7	8.3	33.9
G-TAD [101]	TSN	50.4	34.6	9.0	34.1
P-GCN [102]	I3D	48.3	33.2	3.3	31.1
RTD-Net [74]	I3D	47.2	30.7	8.6	30.8
TadTR [75]	I3D	52.8	37.1	**10.8**	36.1
AFormer [34]	I3D	53.5	36.2	8.2	35.6
DiffTAD [78]	I3D	**56.1**	36.9	9.0	36.1
TriDet [103]	R(2+1)D	54.7	**38.0**	8.4	**36.8**
Self-DETR [77]	I3D	52.3	33.7	8.4	33.8
End-to-End training					
R-C3D	C3D	26.8	-	-	-
PBRNet	I3D	**54.0**	35.0	9.0	35.0
AFSD	I3D	52.4	35.3	6.5	34.4
E2E-TadTR	SF50	50.5	**36.0**	**10.8**	**35.1**
PlusTAD	S50	51.2	33.4	7.6	33.1
APN (ours)	I3D	49.3	31.7	6.7	30.5

of the APN over the other two methods, the performance gap is narrowed considerably on these non-progressive actions.

D. COMPARISON WITH STATE-OF-THE-ART

Our initial comparisons of the APN on the THUMOS14 dataset are summarized in Table 4. The APN demonstrated competitive performance, notably when trained end-to-end[2] and under weak supervision [47]. Specifically, the APN achieved an average mAP of 54.8%, closely approaching the 59.6% achieved by PlusTAD [46]. Remarkably, the APN significantly surpassed ActionBytes [47] (30.3% vs. 3.7%), which was also trained on action frames without incorporating temporal background knowledge.

On ActivityNet-1.3, as shown in Table 5, the APN also achieved competitive detection accuracy, with narrower performance gaps between different TAD methods than observed in THUMOS14. Here, the APN recorded an average mAP of 30.5%, holding its ground against PlusTAD's 33.1% and ActionFormer's 35.6% [34].

The detection precision of end-to-end training and multi-stage training TAD methods are close except for the outstanding ActionFormer [34], likely due to its incorporation of attention mechanism [12] and modern techniques like exponential Moving Average (EMA), score voting [104], distribution-aware initialization [105] and so on. Our approach, focusing on a novel pipeline and action progressions, intentionally avoids these complex implementations.

[2]End-to-end training refers to the simultaneous training of all model parameters.

TABLE 6. Comparison of our method with other state-of-the-art approaches on the DFMAD dataset.

Method	Modality	mAP@ 0.5	0.7	0.9	FPS
Multi-Stage Training					
BMN + P-GCN	RGB-Flow	69.1	67.8	17.7	176
G-TAD + SSN	RGB-Flow	69.2	59.5	17.4	443
G-TAD + P-GCN	RGB-Flow	77.9	58.1	11.8	175
ActionFormer	RGB-Flow	86.1	84.0	20.7	198
End-to-End training					
PlusTAD	RGB	89.8	88.3	34.9	375
APN	RGB-Flow	**98.9**	**92.9**	44.2	456
APN-strict	RGB-Flow	87.3	82.4	**48.5**	460
APN-2D	RGB	96.0	85.7	24.1	**6277**

Nevertheless, the advantages of APN have not been fully exhibited based on the comparison of THUMOS14 and ActivityNet. The experimental results on DFMAD are summarized in Table 6. In this dataset, the APN outperforms all the other investigated methods, including the ActionFormer and PlusTAD in terms of both detection precision and computational efficiency. Concretely, APN obtained a mAP@IoU = 0.9 of 44.2%, significantly outperforming the 20.7% and 34.9% achieved by the ActionFormer and PlusTAD. Moreover, by implementing more stringent parameter adjustments ($P_{start} = 10$, $P_{end} = 90$ of APN-strict), we achieved an enhanced precision of mAP@IoU = 0.9 of 48.5% at a high-quality detection threshold, at the expense of reduced precision in lower-quality detection thresholds (mAP@IoU = 0.5). Impressively, even with a lighter 2D backbone (ResNet-50), the APN maintained superior detection accuracy while operating up to **30** times faster (6277 vs. 175 FPS).[3]

We attribute the exceptional performance on the DFMAD dataset to the APN framework's effectiveness and efficiency. The APN, along with other end-to-end trained TAD methods, surpasses the ActionFormer – the leading method on THUMOS14 and ActivityNet – mainly due to the stark domain shift between the Kinetic400 pre-training dataset and the DFMAD downstream dataset. This substantial shift underscores the critical need for fine-tuning the feature extraction backbone. Additionally, the APN's success can be linked to the nature of the DFMAD actions, which are prolonged and exhibit distinct evolution patterns. These properties allow the APN to fully leverage its capability to intricately dissect action evolution, contributing to its superior performance.

E. QUALITATIVE RESULTS

To have clear observation of the performance of APN on temporal action detection, we visualize the detection results and ground truths of some example videos in Fig. 6. Although the test videos are complicated, our APN can detect the temporal boundaries and categories of actions precisely. Moreover, as shown in Fig. 6 (b), the APN ignored the incomplete action instances in the test videos.

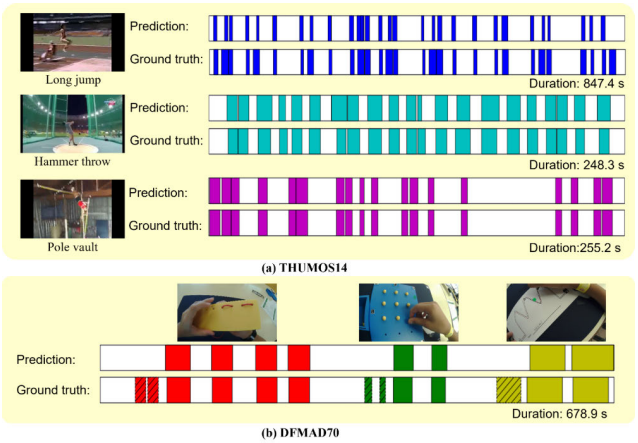

FIGURE 6. Qualitative results of the APN on (a) THUMOS14 and (b) DFMAD. Each rectangular block represents one action instance, different colors represent different action classes, and the blocks with hatch stand for incomplete action instances.

VI. LIMITATIONS

The proposed APN framework exhibits several limitations, detailed below:

Difficulty in detecting non-progressive actions. As outlined in Section III-A, the APN could struggle with identifying actions lacking distinct evolution patterns, such as indeterminate actions (e.g., *painting*), ephemeral actions (e.g., *clicking*), and repetitive actions (e.g., *walking*).

Lack of long-term temporal modeling. Given that the APN processes short-term video snippets, it inherently lacks the capacity for long-term temporal analysis. The long-term modeling is partially accomplished in the profile-matching algorithm, which models progression sequences but may fall short in scenarios requiring intricate long-term temporal relationships.

Computational efficiency under large prediction errors. Although the computational cost of Algorithm 1 is low for most cases, it becomes non-negligible when the APN's prediction error (MAE) is large, *e.g.*, MAE = 18. This algorithm should still have a lot of room for optimization.

Inability to detect incomplete actions. This characteristic can be seen both as a strength and a limitation of the APN, depending on the research objective or application's needs. The framework consistently omits incomplete action instances, which might be not desirable for certain analyses when incomplete actions need to be detected.

VII. CONCLUSION

In this paper, we introduce a pioneering approach to temporal action detection, featuring a novel pipeline that enables end-to-end training of a snippet-level model, the Action Progression Network (APN). Central to our framework is

[3]FPS performance evaluated with a 2080Ti NVIDIA GPU. Comparison in FLOP is omitted due to variable video processing cycles for different TAD methods.

the concept of action progressions that quantitatively encode the action evolution process into ordered ranks. Action progressions enrich our model with temporal context and allow for the construction of detailed temporal models for actions that could facilitate precise action localization. Through rigorous evaluation of three benchmark datasets, the APN has demonstrated competitive performance against existing methodologies. Notably, the APN shows exceptional capability in identifying actions characterized by prolonged durations or distinct evolutionary patterns, achieving substantially higher precision at speeds up to 30 times faster than competing models.

Two distinctive features of the APN further underscore its practical utility: its adaptability to training with trimmed videos and its designed avoidance of incomplete action detections. These aspects significantly broaden the applicability of our model in practical settings. We envisage that the introduction of action progressions will spur advancements in temporal action detection, steering future models towards simplicity and greater real-world applicability. It is our hope that this work will inspire the action detection community to explore new directions and further leverage the concept of action progressions.

APPENDIX
THE MINIMUM ERROR OF THE RANK PREDICTION WITHOUT PRIOR KNOWLEDGE

Given a set of ranks that uniformly spread across a limited range, what is the minimum prediction error we can obtain without any prior knowledge except for the range? Assume that the range is $[0, K]$ and the set's size is infinite. The mean absolute error (MAE) and mean squared error (MSE) can be derived as follows.

Without knowledge of the ground truth value, one may *randomly* take values from the range $[0, K]$ as the predicted ranks. In this case, the errors are:

$$E_r^{MAE} = \frac{1}{K}\int_0^K \frac{1}{K}\left[\int_0^{\hat{p}}(\hat{p}-p)\mathrm{d}p + \int_{\hat{p}}^K (p-\hat{p})\mathrm{d}p\right]\mathrm{d}\hat{p} = \frac{K}{3}, \tag{15}$$

$$E_r^{MSE} = \frac{1}{K}\int_0^K \frac{1}{K}\int_0^K (\hat{p}-p)^2\mathrm{d}p\mathrm{d}\hat{p} = \frac{K^2}{6}, \tag{16}$$

where p and $\hat{p}$ are the ground truth and predicted rank, respectively.

In addition to random prediction, one may use a fixed value as the predicted rank. In this case, the errors are:

$$\begin{aligned} E_f^{MAE} &= \frac{1}{K}\left[\int_0^{\hat{p}}(\hat{p}-p)\mathrm{d}p + \int_{\hat{p}}^K (p-\hat{p})\mathrm{d}p\right] \\ &= \frac{\hat{p}^2}{K} - \hat{p} + \frac{K}{2}, \end{aligned} \tag{17}$$

$$\begin{aligned} E_f^{MSE} &= \frac{1}{K}\int_0^K (\hat{p}-p)^2\mathrm{d}p \\ &= K\hat{p}^2 - K^2\hat{p} + \frac{K^3}{3}. \end{aligned} \tag{18}$$

It is easy to derive their minimum with respect to $\hat{p}$:

$$\min E_f^{MAE} = \frac{K}{4}, \text{ when } \hat{p} = \frac{K}{2}, \tag{19}$$

$$\min E_f^{MSE} = \frac{K^3}{12}, \text{ when } \hat{p} = \frac{K}{2}. \tag{20}$$

We set $K = 100$ in our experiments; therefore, the minimum MAE should be the $\min E_f^{MAE} = 100/4 = 25$, and the minimum MSE should be the $E_r^{MSE} = 100^2/6 = 1666.66$. The former can be used as a baseline to compare the MAE obtained by the APN, i.e., MAE in Tables 1, 2, and 3. The latter is where the 1666.66 in Eq. (14) comes from.

REFERENCES

[1] T. Barnett, S. Jain, U. Andra, and T. Khurana, "CISCO visual networking index (VNI) complete forecast update," Americas/EMEAR CISCO Knowl. Netw. (CKN) Presentation, Tech. Rep., 2017, pp. 1–30.

[2] M. Rahevar, A. Ganatra, T. Saba, A. Rehman, and S. A. Bahaj, "Spatial–temporal dynamic graph attention network for skeleton-based action recognition," *IEEE Access*, vol. 11, pp. 21546–21553, 2023.

[3] M. Rahevar and A. Ganatra, "Spatial–temporal gated graph attention network for skeleton-based action recognition," *Pattern Anal. Appl.*, vol. 26, no. 3, pp. 929–939, Aug. 2023.

[4] X. Gao, Z. Chang, X. Ran, and Y. Lu, "CANet: Comprehensive attention network for video-based action recognition," *Knowledge-Based Syst.*, vol. 296, Jul. 2024, Art. no. 111852.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015, *arXiv:1409.1556*.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[13] OpenAI et al., "GPT-4 technical report," 2024, *arXiv:2303.08774*.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2009, pp. 248–255.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28.

[16] D. Oneata, J. Verbeek, and C. Schmid, "Action and event recognition with Fisher vectors on a compact feature set," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1817–1824. [Online]. Available: https://www.cv-foundation.org/openaccess/content_iccv_2013/papers/Oneata_Action_and_Event_2013_ICCV_paper.pdf

[17] D. Oneata, "The LEAR submission at Thumos 2014," HAL, Tech. Rep., 2014.

[18] L. Wang, Y. Qiao, and X. Tang, "Action recognition and detection by combining motion and appearance features," *THUMOS14 Action Recognit. Challenge*, vol. 1, no. 2, p. 2, 2014. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.718.527&rep=rep1&type=pdf

[19] Z. Shou, D. Wang, and S.-F. Chang, "Temporal action localization in untrimmed videos via multi-stage CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1049–1058.
[20] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 4, pp. 814–830, Apr. 2016.
[21] J. Gao, K. Chen, and R. Nevatia, "CTAP: Complementary temporal action proposal generation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 68–83.
[22] F. C. Heilbron, J. C. Niebles, and B. Ghanem, "Fast temporal activity proposals for efficient detection of human actions in untrimmed videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1914–1923.
[23] G. Singh and F. Cuzzolin, "Untrimmed video classification for activity detection: Submission to ActivityNet challenge," 2016, *arXiv:1607.01979*.
[24] S. Buch, V. Escorcia, B. Ghanem, L. Fei-Fei, and J. C. Niebles, "End-to-end, single-stream temporal action detection in untrimmed videos," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2017. [Online]. Available: http://www.bmva.org/bmvc/2017/papers/paper093/paper093.pdf
[25] T. Lin, X. Zhao, and Z. Shou, "Single shot temporal action detection," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 988–996.
[26] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar, "Rethinking the faster R-CNN architecture for temporal action localization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1130–1139.
[27] T. Lin, X. Zhao, H. Su, C. Wang, and M. Yang, "BSN: Boundary sensitive network for temporal action proposal generation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018.
[28] T. Lin, X. Liu, X. Li, E. Ding, and S. Wen, "BMN: Boundary-matching network for temporal action proposal generation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3888–3897.
[29] F. Long, T. Yao, Z. Qiu, X. Tian, J. Luo, and T. Mei, "Gaussian temporal awareness networks for action localization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 344–353.
[30] C. Lin, J. Li, Y. Wang, Y. Tai, D. Luo, Z. Cui, C. Wang, J. Li, F. Huang, and R. Ji, "Fast learning of temporal action proposal via dense boundary generator," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 7, pp. 11499–11506. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6815
[31] X. Wang, C. Gao, S. Zhang, and N. Sang, "Multi-level temporal pyramid network for action detection," in *Proc. Chin. Conf. Pattern Recognit. Comput. Vis. (PRCV)*. Nanjing, China: Springer, 2020, pp. 41–54.
[32] L. Yang, H. Peng, D. Zhang, J. Fu, and J. Han, "Revisiting anchor mechanisms for temporal action localization," *IEEE Trans. Image Process.*, vol. 29, pp. 8535–8548, 2020.
[33] P. Zhao, L. Xie, C. Ju, Y. Zhang, Y. Wang, and Q. Tian, "Bottom-up temporal action localization with mutual regularization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Springer, 2020, pp. 539–555.
[34] C. Zhang, J. Wu, and Y. Li, "ActionFormer: Localizing moments of actions with transformers," 2022, *arXiv:2202.07925*.
[35] J. Carreira and A. Zisserman, "Quo Vadis, action recognition? A new model and the kinetics dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4724–4733.
[36] B. Wang, Y. Zhao, L. Yang, T. Long, and X. Li, "Temporal action localization in the deep learning era: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 4, pp. 2171–2190, Apr. 2024.
[37] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin, "Temporal action detection with structured segment networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2933–2942.
[38] H. Qiu, Y. Zheng, H. Ye, Y. Lu, F. Wang, and L. He, "Precise temporal action localization by evolving temporal proposals," in *Proc. ACM Int. Conf. Multimedia Retr.*, Jun. 2018, pp. 388–396.
[39] S. Ma, L. Sigal, and S. Sclaroff, "Learning activity progression in LSTMs for activity detection and early detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1942–1950.
[40] B. Hu, J. Cai, T.-J. Cham, and J. Yuan, "Progress regression RNN for online spatial–temporal action localization in unconstrained videos," 2019, *arXiv:1903.00304*.
[41] S. Qu, G. Chen, D. Xu, J. Dong, F. Lu, and A. Knoll, "LAP-Net: Adaptive features sampling via learning action progression for online action detection," 2020, *arXiv:2011.07915*.
[42] P. A. Gutiérrez, M. Pérez-Ortiz, J. Sánchez-Monedero, F. Fernández-Navarro, and C. Hervás-Martínez, "Ordinal regression methods: Survey and experimental study," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 1, pp. 127–146, Jan. 2016.
[43] W. Cao, V. Mirjalili, and S. Raschka, "Rank-consistent ordinal regression for neural networks," 2019, *arXiv:1901.07884*.
[44] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. *THUMOS Challenge: Action Recognition With a Large Number of Classes*. Accessed: Sep. 1, 2024. [Online]. Available: http://crcv.ucf.edu/THUMOS14/
[45] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles, "ActivityNet: A large-scale video benchmark for human activity understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 961–970.
[46] M. Yang, G. Chen, Y.-D. Zheng, T. Lu, and L. Wang, "BasicTAD: An astounding RGB-only baseline for temporal action detection," *Comput. Vis. Image Understand.*, vol. 232, Jul. 2023, Art. no. 103692.
[47] M. Jain, A. Ghodrati, and C. G. M. Snoek, "ActionBytes: Learning from trimmed videos to localize actions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1168–1177.
[48] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia, "Temporal localization of fine-grained actions in videos by domain transfer from web images," in *Proc. 23rd ACM Int. Conf. Multimedia*, Oct. 2015, pp. 371–380.
[49] K. K. Singh and Y. J. Lee, "Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3544–3553.
[50] L. Wang, Y. Xiong, D. Lin, and L. Van Gool, "UntrimmedNets for weakly supervised action recognition and detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6402–6411.
[51] Z. Shou, H. Gao, L. Zhang, K. Miyazawa, and S.-F. Chang, "AutoLoc: Weakly-supervised temporal action localization in untrimmed videos," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 154–171.
[52] P. Nguyen, B. Han, T. Liu, and G. Prasad, "Weakly supervised action localization by sparse temporal pooling network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6752–6761.
[53] Z. Liu, L. Wang, Q. Zhang, Z. Gao, Z. Niu, N. Zheng, and G. Hua, "Weakly supervised temporal action localization through contrast based evaluation networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3898–3907.
[54] S. Narayan, H. Cholakkal, F. S. Khan, and L. Shao, "3C-net: Category count and center loss for weakly-supervised action localization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8678–8686. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2019/papers/Narayan_3C-Net_Category_Count_and_Center_Loss_for_Weakly
[55] K. Min and J. J. Corso, "Adversarial background-aware loss for weakly-supervised temporal activity localization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Seoul, South Korea: Springer, 2020, pp. 283–299.
[56] B. Shi, Q. Dai, Y. Mu, and J. Wang, "Weakly-supervised action localization by generative attention modeling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1006–1016.
[57] Z. Luo, D. Guillory, B. Shi, W. Ke, F. Wan, T. Darrell, and H. Xu, "Weakly-supervised action localization with expectation-maximization multi-instance learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Springer, 2020, pp. 729–745.
[58] Y. Zhai, L. Wang, W. Tang, Q. Zhang, J. Yuan, and G. Hua, "Two-tream consensus network for weakly-supervised temporal action localization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Springer, 2020, pp. 37–54. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-58539-6_3
[59] S. Qu, G. Chen, Z. Li, L. Zhang, F. Lu, and A. Knoll, "ACM-Net: Action context modeling network for weakly-supervised temporal action localization," 2021, *arXiv:2104.02967*.
[60] P. Lee, J. Wang, Y. Lu, and H. Byun, "Weakly-supervised temporal action localization by uncertainty modeling," 2020, *arXiv:2006.07006v3*.
[61] G. Singh, V. Choutas, S. Saha, F. Yu, and L. Van Gool, "Spatio-temporal action detection under large motion," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2023, pp. 5998–6007.
[62] Y. Tang, Y. Zheng, C. Wei, K. Guo, H. Hu, and J. Liang, "Video representation learning for temporal action detection using global-local attention," *Pattern Recognit.*, vol. 134, Feb. 2023, Art. no. 109135.
[63] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, *arXiv:1212.0402*.

[64] Y. Feng, Z. Zhang, R. Quan, L. Wang, and J. Qin, "RefineTAD: Learning proposal-free refinement for temporal action detection," in *Proc. 31st ACM Int. Conf. Multimedia*, 2023, pp. 135–143.
[65] H. Xu, A. Das, and K. Saenko, "R-C3D: Region convolutional 3D network for temporal activity detection," 2017, *arXiv:1703.07814*.
[66] K. Yang, P. Qiao, D. Li, S. Lv, and Y. Dou, "Exploring temporal preservation networks for precise temporal action localization," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 7477–7484.
[67] X. Liu, S. Bai, and X. Bai, "An empirical study of end-to-end temporal action detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 19978–19987.
[68] J. Gao, Z. Yang, C. Sun, K. Chen, and R. Nevatia, "TURN TAP: Temporal unit regression network for temporal action proposals," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3648–3656. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2017/papers/Gao_TURN_TAP_Temporal_ICCV_2017_paper.pdf
[69] D. Zhang, X. Dai, X. Wang, and Y.-F. Wang, "S3D: Single shot multi-span detector via fully 3D convolutional networks," 2018, *arXiv:1807.08069*.
[70] J. Li, X. Liu, Z. Zong, W. Zhao, M. Zhang, and J. Song, "Graph attention based proposal 3D ConvNets for action detection," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 4, pp. 4626–4633.
[71] Q. Liu and Z. Wang, "Progressive boundary refinement network for temporal action detection," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 11612–11619.
[72] C. Wang, H. Cai, Y. Zou, and Y. Xiong, "RGB stream is enough for temporal action detection," 2021, *arXiv:2107.04362*.
[73] J. Wu, P. Sun, S. Chen, J. Yang, Z. Qi, L. Ma, and P. Luo, "Towards high-quality temporal action detection with sparse proposals," 2021, *arXiv:2109.08847*.
[74] J. Tan, J. Tang, L. Wang, and G. Wu, "Relaxed transformer decoders for direct action proposal generation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2021, pp. 13526–13535.
[75] X. Liu, Q. Wang, Y. Hu, X. Tang, S. Zhang, S. Bai, and X. Bai, "End-to-end temporal action detection with transformer," *IEEE Trans. Image Process.*, vol. 31, pp. 5427–5441, 2022, doi: 10.1109/TIP.2022.3195321.
[76] D. Shi, Y. Zhong, Q. Cao, J. Zhang, L. Ma, J. Li, and D. Tao, "ReAct: Temporal action detection with relational queries," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Tel Aviv-Yafo, Israel: Springer, 2022, pp. 105–121.
[77] J. Kim, M. Lee, and J.-P. Heo, "Self-feedback DETR for temporal action detection," 2023, *arXiv:2308.10570*.
[78] S. Nag, X. Zhu, J. Deng, Y.-Z. Song, and T. Xiang, "DiffTAD: Temporal action detection with proposal denoising diffusion," 2023, *arXiv:2303.14863*.
[79] Y. Tang, C. Niu, M. Dong, S. Ren, and J. Liang, "AFO-TAD: Anchor-free one-stage detector for temporal action detection," 2019, *arXiv:1910.08250*.
[80] C. Lin, C. Xu, D. Luo, Y. Wang, Y. Tai, C. Wang, J. Li, F. Huang, and Y. Fu, "Learning salient boundary feature for anchor-free temporal action localization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 3320–3329.
[81] T. Han, J. Wang, A. Cherian, and S. Gould, "Human action forecasting by learning task grammars," 2017, *arXiv:1709.06391*.
[82] X. Li, Y. Zhang, J. Zhang, M. Zhou, S. Chen, Y. Gu, Y. Chen, I. Marsic, R. A. Farneth, and R. S. Burd, "Progress estimation and phase detection for sequential processes," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 1, no. 3, pp. 1–20, Sep. 2017.
[83] A. Kukleva, H. Kuehne, F. Sener, and J. Gall, "Unsupervised learning of action classes with continuous temporal embedding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12066–12074.
[84] F. Becattini, T. Uricchio, L. Seidenari, L. Ballan, and A. D. Bimbo, "Am i done? Predicting action progress in videos," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 16, no. 4, pp. 1–24, Nov. 2020.
[85] R. G. VidalMata, W. J. Scheirer, A. Kukleva, D. Cox, and H. Kuehne, "Joint visual-temporal embedding for unsupervised learning of actions in untrimmed sequences," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 1238–1247.
[86] D. Pucci, F. Becattini, and A. Del Bimbo, "Joint-based action progress prediction," *Sensors*, vol. 23, no. 1, p. 520, Jan. 2023.
[87] J. Dijkstra and S. L. Pintea, "Is there progress in activity progress prediction?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV) Workshops*, Oct. 2023, pp. 2958–2966.
[88] Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua, "Ordinal regression with multiple output CNN for age estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4920–4928.
[89] R. Díaz and A. Marathe, "Soft labels for ordinal regression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4733–4742.
[90] C. Lu, R. Li, H. Fu, B. Fu, Y. Wang, W.-L. Lo, and Z. Chi, "Precise temporal localization for complete actions with quantified temporal structure," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 4781–4788.
[91] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast networks for video recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6201–6210.
[92] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?" in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 2, 2021, p. 4.
[93] E. F. Harrington, "Online ranking/collaborative filtering using the perceptron algorithm," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 250–257.
[94] A. Rosenfeld and M. Thurston, "Edge and curve detection for visual scene analysis," *IEEE Trans. Comput.*, vol. C-20, no. 5, pp. 562–569, May 1971.
[95] R. Li, H. Fu, Y. Zheng, W.-L. Lo, J. J. Yu, C. H. P. Sit, Z. Chi, Z. Song, and D. Wen, "Automated fine motor evaluation for developmental coordination disorder," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 27, no. 5, pp. 963–973, May 2019.
[96] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime TV-L^1 optical flow," in *Proc. 29th DAGM Symp.* Heidelberg, Germany: Springer, Sep. 2007, pp. 214–223.
[97] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 568–576.
[98] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[99] J. Carreira, E. Noland, C. Hillier, and A. Zisserman, "A short note on the kinetics-700 human action dataset," 2019, *arXiv:1907.06987*.
[100] R. Goyal, S. Ebrahimi Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, and M. Mueller-Freitag, "The 'something something' video database for learning and evaluating visual common sense," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 5842–5850.
[101] M. Xu, C. Zhao, D. S. Rojas, A. Thabet, and B. Ghanem, "G-TAD: Sub-graph localization for temporal action detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10153–10162.
[102] R. Zeng, W. Huang, M. Tan, Y. Rong, P. Zhao, J. Huang, and C. Gan, "Graph convolutional networks for temporal action localization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 1–10.
[103] D. Shi, Y. Zhong, Q. Cao, L. Ma, J. Li, and D. Tao, "TriDet: Temporal action detection with relative boundary modeling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 18857–18866.
[104] K. Kim and H. S. Lee, "Probabilistic anchor assignment with iou prediction for object detection," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K., Springer, Aug. 2020, pp. 355–371.
[105] A. Karpathy. (2019). *A Recipe for Training Neural Networks*. [Online]. Available: https://Karpathy.github.io

CHONG-KAI LU received the B.Sc. degree from the University of Electronic Science and Technology of Chia, in 2017, and the M.Sc. degree from The Hong Kong Polytechnic University, Hong Kong, in 2019, where he is currently pursuing the Ph.D. degree. His research interests include computer vision, deep learning, and video content analysis.

MAN-WAI MAK (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from the University of Northumbria, in 1993. He joined the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, in 1993, where he is currently a Professor and the Interim Head. He has authored more than 200 technical articles in speaker recognition, machine learning, and bioinformatics. He also co-authored postgraduate textbooks *Biometric Authentication: A Machine Learning Approach* (Prentice-Hall, 2005) and *Machine Learning for Speaker Recognition* (Cambridge University Press, 2020). His research interests include speaker recognition, machine learning, and bioinformatics. He served as a member for the IEEE Machine Learning for Signal Processing Technical Committee, from 2005 to 2007. He also served as a Technical Committee Member for several international conferences, including ICASSP and Interspeech. He gave a tutorial on machine learning for speaker recognition in Interspeech'2016. He has served as an Associate Editor for IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH AND LANGUAGE PROCESSING. He is currently an Associate Editor of *Journal of Signal Processing Systems* and IEEE BIOMETRICS COMPENDIUM.

RUIMIN LI (Member, IEEE) received the B.S. degree from Shanxi University, in 2015, and the Ph.D. degree from Xi'an Institute of Optics and Precision Mechanics, CAS, in January 2021. She is currently a Lecturer with the Academy of Advanced Interdisciplinary Research, Xidian University. Her research interests include human action recognition, temporal action localization, human action evaluation, intelligent diagnostic systems development, and remote sensing image processing.

ZHERU CHI (Member, IEEE) received the B.Eng. and M.Eng. degrees from Zhejiang University, in 1982 and 1985, respectively, and the Ph.D. degree from The University of Sydney, in March 1994. From 1985 to 1989, he was on the Faculty of the Department of Scientific Instruments, Zhejiang University. He was a Senior Research Assistant/a Research Fellow with the Laboratory for Imaging Science and Engineering, The University of Sydney, from April 1993 to January 1995. In February 1995, he joined The Hong Kong Polytechnic University, where he was an Associate Professor with the Department of Electronic and Information Engineering, before he retired from the post, in April 2021. He has co-authored one book and 12 book chapters, and published more than 240 technical papers. His research interests include pattern recognition, machine learning, and computational intelligence techniques. Since 1997, he has been served as a coorganizer for a special session/session chair/area moderator/program committee member for a number of international conferences. He was a Contributor to Comprehensive Dictionary of Electrical Engineering (CRC Press and IEEE Press, 1999). He was an Associate Editor of IEEE TRANSACTIONS ON FUZZY SYSTEMS, from 2008 to 2010; and a Technical Editor of *International Journal of Information Acquisition*.

HONG FU received the B.S. and M.S. degrees from Xi'an Jiaotong University, in 2000 and 2003, respectively, and the Ph.D. degree from The Hong Kong Polytechnic University, in 2007. She is currently an Assistant Professor with the Department of Mathematics and Information Technology, The Education University of Hong Kong.