DyFADet: Dynamic Feature Aggregation for Temporal Action Detection

Le Yang[1*(✉)], Ziwei Zheng[1*], Yizeng Han[2], Hao Cheng[3], Shiji Song[4], Gao Huang[4], and Fan Li[1]

[1] Xi'an Jiaotong University
{yangle15, lifan}@xjtu.edu.cn, ziwei.zheng@stu.xjtu.edu.cn
[2] DAMO Academy, Alibaba group hanyizeng.hyz@alibaba-inc.com
[3] HKUST(GZ) hcheng046@connect.hkust-gz.edu.cn
[4] Tsinghua University
{shijis, gaohuang}@tsinghua.edu.cn

Abstract. Recent proposed neural network-based Temporal Action Detection (TAD) models are inherently limited to extracting the discriminative representations and modeling action instances with various lengths from complex scenes by shared-weights detection heads. Inspired by the successes in dynamic neural networks, in this paper, we build a novel dynamic feature aggregation (DFA) module that can simultaneously adapt kernel weights and receptive fields at different timestamps. Based on DFA, the proposed dynamic encoder layer aggregates the temporal features within the action time ranges and guarantees the discriminability of the extracted representations. Moreover, using DFA helps to develop a Dynamic TAD head (DyHead), which adaptively aggregates the multi-scale features with adjusted parameters and learned receptive fields better to detect the action instances with diverse ranges from videos. With the proposed encoder layer and DyHead, a new dynamic TAD model, DyFADet, achieves promising performance on a series of challenging TAD benchmarks, including HACS-Segment, THUMOS14, ActivityNet-1.3, Epic-Kitchen 100, Ego4D-Moment QueriesV1.0, and FineAction. Code is released to https://github.com/yangle15/DyFADet-pytorch.

Keywords: Temporal action detection · Dynamic network architectures · Video understanding

1 Introduction

As a challenging and essential task within the field of video understanding, **T**emporal **A**ction **D**etection (TAD) has received widespread attention in recent years. The target of TAD is to simultaneously recognize action categories and localize action temporal boundaries from an untrimmed video. Various methods have been developed to address this task, which can be mainly divided into two categories: 1) Two-stage methods (such as [37, 42]), which first learn to generate

∗ Equal contribution. ✉ Corresponding author.

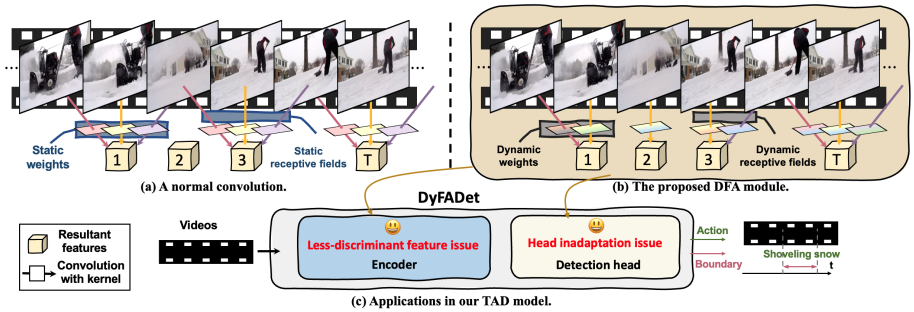


Fig. 1: Differences between convolution and DFA. (a) A normal convolution with static weights and receptive fields. (b) The dynamic formations of DFA at different timestamps. (c) Implementing DFA to build DyFADet can address the two issues in TAD.

the class-agnostic action proposals and then conduct classification and boundary-refinement in proposal-level. 2) One-stage methods [25, 39, 44, 59] classify each frame as well as locate temporal boundaries in an end-to-end manner, achieving better performance and becoming a popular line of TAD research currently.

However, accurately detecting an action from an untrimmed video remains a challenging task. On the one hand, the spatial redundancy in the adjacent frames along with the static feature extraction strategy can result in poor discriminability of learned representations, hindering the detection performance [39, 44]. On the other hand, the head inadaptation issue can happen in common TAD designs [59], where a static shared-weight head is used to detect action instances with diverse temporal lengths, leading to sub-optimal performance. Therefore, it is necessary to find a solution that can simultaneously address these two key issues in modern TAD models.

In this paper, inspired by the recent success of dynamic neural networks [16, 21], we develop a novel **D**ynamic **F**eature **A**ggregation (DFA) module for TAD. As is shown in Fig. 1, the proposed DFA, the kernel shape, and the network parameters can adapt based on input during generating new features, which significantly differs from the static feature learning procedure in most existing TAD models. Such a learning mechanism enables a dynamic feature aggregation procedure, which has the ability to increase the discriminability of the learned representations, and also adapts the feature learning procedure in detection heads based on each level of the feature pyramid during detection.

Based on the DFA, we build a dynamic encoder (DynE) and a Dynamic TAD head (DyHead) to address the two issues in TAD models. On the one hand, the dynamic learning of DynE can facilitate the features of target action to be gathered, and increase the differences between the features of the action and boundaries, resolving the first issue in Fig. 1. On the other hand, the DyHead can adjust detector parameters when it is applied at different pyramid levels for detecting actions with different time ranges. By effectively addressing the two

issues in Fig. 1, the proposed DyFADet with DynE and DyHead can achieve accurate detection results in TAD tasks.

We evaluate the proposed DyFADet on a series of TAD datasets including HACS-Segment [61], THUMOS14 [22], ActivityNet-1.3 [5], Epic-Kitchen 100 [13], Ego4D Moment Queries V1.0 [15] and FineAction [32]. The experimental results show the effectiveness of the proposed DyFADet in TAD tasks.

2 Related Work

Temporal action detection is a challenging video understanding task, which involves localizing and classifying the actions in an untrimmed video. Conventional two-stage approaches [8, 25, 27, 28, 37, 42, 54, 57, 58, 65] usually consist of two steps: proposal generation and classification. Nevertheless, these may suffer from high complexity and end-to-end training could be infeasible. A recent trend is designing one-stage frameworks and training the model in an end-to-end fashion. Some works [31, 40, 43] propose to detect actions with the DETR-like decoders [6], and another line of work [10, 39, 59] builds a multi-scale feature pyramid followed by a detection head. In this work, we mainly follow the mainstream *encoder-pyramid-head* framework. Concretely, we build our model based on the frameworks in the highly competitive methods [39,59], and boost the final detection performance by introducing the *dynamic* mechanism to simultaneously solve the two issues in Fig. 1.

Dynamic neural networks have attracted great research interests in recent years due to their favorable efficiency and representation power [16]. Unlike conventional models that process different inputs with the same computation, dynamic networks can adapt their architectures and parameters conditioned on input samples [20,56] or different spatial [17,18,48] / temporal [36,52] positions.

Data-dependent parameters have shown effectiveness in increasing the representation power with minor computational overhead [9, 12, 34, 55]. Existing approaches can generally be divided into two groups: one adopts mechanisms to dynamically re-weight parameter *values* [9, 34, 55], including modern self-attention operators, which can be interpreted as (depth-wise) dynamic convolutions [64]. While the static temporal reception fields of these methods can lead to the less-discriminant feature issue as stated in [39,44]. The other develops deformable convolution to achieve dynamic *reception fields* [12], which have been effectively utilized in different video understanding tasks [23, 24, 35]. However, the kernel weights of these deformable convolutions are static. Compared to the previous methods, our DFA *simultaneously* adapts the convolution weights and temporal reception fields in a *data-dependent* manner, leading to more effective and flexible modeling of temporal information in the TAD task.

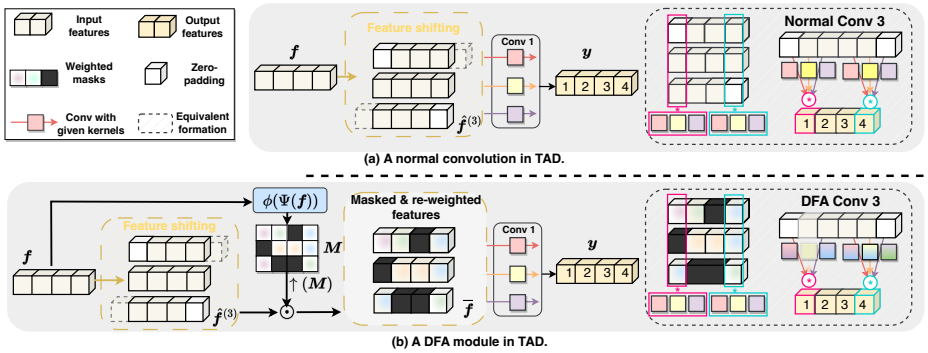


Fig. 2: An illustration of the proposed DFA module. (a) A normal convolution with the kernel size of 3 (Conv 3) and its corresponding formation realized by a shifting module and a point-wise convolution. (b) A DFA. The shifted representations multiplied with the weighted mask will be sent to the point-wise convolution. The DFA module is equivalent to a convolution with adaptive kernel weights and receptive fields.

3 Method

3.1 Dynamic feature aggregation

DFA based on feature shifting. The Dynamic feature aggregation (DFA) needs to effectively adapt the kernel weights and the receptive fields (including the number of sampling positions and the shape of kernels) based on the inputs to improve the feature extraction ability, which is difficult to simultaneously achieve. However, such a procedure can be realized if we separate a normal convolution into a feature-shifting module and a point-wise convolution [50] (shown in Fig. 2): A convolution with the kernel size of k equals that the input features are first shifted according to the k kernel positions, and then processed by a point-wise convolution (Conv 1). Motivated by this, we can learn an input-based mask, which will be used to zero and re-weight the shifted features. Then, using a Conv 1 to process the masked shifted features will equal a fully dynamic convolution shown in Fig 2 (b). A light-weight module, $\Psi(\cdot)$, and a non-linear activation, $\phi(\cdot)$ can be used to generate the mask, where the activation function can be the one whose output is always 0 with any negative inputs (such as ReLU [1] or a general-restricted tanh function [41]).

Formally, suppose we have a standard convolution with the kernel $\mathcal{K} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times k}$, where k is the kernel size and C_{out}, C_{in} are the numbers of input and output channels. Given an input $\boldsymbol{f} \in \mathbb{R}^{C_{\text{in}} \times T}$ (T is the length of temporal dimension and $f_t \in \mathbb{R}^{C_{\text{in}}}$, is the tensor at time t), then we can shift the input by

$$\hat{\boldsymbol{f}}^{(k)} = \text{Shift}(\boldsymbol{f}, k) = \left[\hat{\boldsymbol{f}}^0; \ \ldots \ ; \hat{\boldsymbol{f}}^{k-1}\right] \in \mathbb{R}^{kC_{\text{in}} \times T}, \tag{1}$$
$$\hat{f}_t^s = f_{t-\lfloor k/2 \rfloor + s}, \ s = 0, 1, ..., k-1, \ t = 1, 2, ..., T,$$

where $\hat{\boldsymbol{f}}^s \in \mathbb{R}^{C_{\text{in}} \times T}$, the empty positions of the shifted features will be padded by all-zero tensors. The weighted masks can be calculated by

$$\boldsymbol{M} = \phi(\ \Psi(\boldsymbol{f})\ ), \quad \boldsymbol{M} \in \mathbb{R}^{C_{\text{m}} \times T}, \tag{2}$$

where C_{m} and Ψ can be designed into different formations to achieve different dynamic properties. By repeated unpsampling the $\boldsymbol{M}$ to the dimension of $\hat{\boldsymbol{f}}^{(k)}$ (denoted by $\uparrow(\cdot)$), we have

$$\overline{\boldsymbol{f}}\ = \uparrow(\boldsymbol{M}) \odot \hat{\boldsymbol{f}}^{(k)} = \overline{\boldsymbol{M}} \odot \hat{\boldsymbol{f}}^{(k)}, \tag{3}$$

where we use $\overline{\boldsymbol{M}}$ to represent $\uparrow(\boldsymbol{M})$ for simplicity, and $\odot$ means element-wise multiplication. The final output features can be written as

$$\boldsymbol{y} = \text{DFA}(\boldsymbol{f}) = \sum_{s=0}^{k-1} \mathcal{K}_s(\overline{\boldsymbol{M}}_{[sC_{\text{in}}+1:(s+1)C_{\text{in}}]} \odot \hat{\boldsymbol{f}}^s), \tag{4}$$

where $\mathcal{K}_s \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}$, $\overline{\boldsymbol{M}}_{[(sC_{\text{in}}+1:(s+1)C_{\text{in}}]} \in \mathbb{R}^{C_{\text{in}} \times T}$ means the mask tensor with the index from $sC_{\text{in}}+1$ to $(s+1)C_{\text{in}}$, and $\boldsymbol{y} \in \mathbb{R}^{C_{\text{out}} \times T}$ is the output.

Different formations of DFA. Using different $\Psi(\cdot)$ will result in the different formations of DFA. By implementing $\Psi(\cdot)$ as a convolution, the DFA can be built into a convolution-based DFA module (DFA_Conv). Moreover, changing the C_{m} leads to the different dynamic properties. Take DFA_Conv as an example. K-formation: using $C_{\text{m}} = k$, the $\uparrow(\cdot)$ will repeat the mask in the interleaved manner at the channel dimensions with C_{in} times, which makes the DFA share the same dynamic receptive field among the channel dimension. C-formation: For $C_{\text{m}} = C_{\text{in}}$, the $\uparrow(\cdot)$ will repeat the mask k times at the channel dimensions, which makes the DFA a temporal dynamic channel pruning convolution. Also, using $C_{\text{m}} = kC_{\text{in}}$ (CK), the DFA will adapt the receptive fields at different timestamps and channels.

Moreover, inspired by the success of Transformer-based architecture, we further implemented the DFA in as the formation in Fig. 3 (b). The generated mask will result in the zeros in the attention matrix at the unimportant timestamps, and then the masked attentions will be used to re-weight the shifted features.

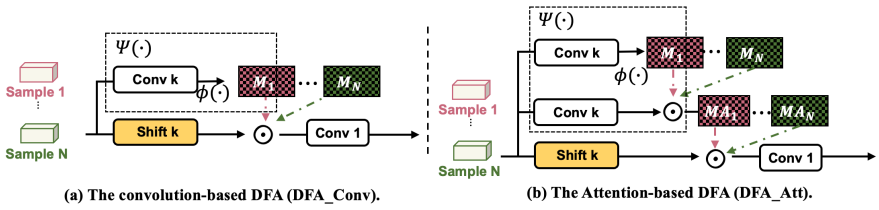


Fig. 3: Two different formations of the proposed DFA module.

Differences to existing works. If we restrict that $\boldsymbol{M} \in \{0, 1\}$ has the same number of masked positions, then DFA will equal to 1-d deformable convolution [12, 23], which adapts the shape of the kernels at different timestamp. If we remove the $\phi(\cdot)$, the DFA will equal to the dynamic convolution in [9, 55] which only adapts the kernel weights-based inputs. Moreover, TD3DConv in [24] first uses temporal attention weights to adjust the features and then uses DC for feature learning, where these weights(W) are input-dependent yet temporal static during generating new features. While these weights are fully dynamic in our DFA. The DFA provides an effective way for dynamic feature aggregation, which addresses the aforementioned issues in modern TAD models. Therefore, based on it, we further develop two important components Dynamic encoder and Dynamic TAD head for the proposed TAD model, DyFADet.

3.2 DyFADet

Overview Based on the dynamic property of DFA, we can build a TAD model that can effectively extract the discriminative representations in the encoder and adapt the detection heads for the action instances with different ranges. Following the common design in [39, 59], the proposed DyFADet consists of a video feature backbone, an encoder, and two detection heads for action classification and localization. Concretely, we first extract the video features using a pre-trained backbone. Then the extracted representations will be sent to the encoder to generate the features pyramid, where the former features will be down-sampled by the DynE layer with the scale of 2 to obtain $\boldsymbol{f}^l, l = 1, ..., L$ (L is the number of total levels). These features will be then used by DyHead for action detection. The architecture of DyFADet is shown in Fig. 4 (a), where the DynE layer and DyHead are two components built based on DFA applying the dynamic feature learning strategy.

Feature encoder with DynE. As illustrated in Fig. 4 (b), the DynE layer is built by substituting the SA module with the proposed DynE module, which follows the marco-architecture of transformer layers [59]. The DynE module has two branches: A instance-dynamic branch based on DFA_Conv with kernel size of 1 generates the global temporal weighted mask to help aggregate global the action information. For another branch, we propose to use the DFA_Conv with convolutions implemented by different kernel sizes to better generate the weighted mask tenors. To further improve the efficiency during feature learning, all convolution modules are implemented by depth-wise convolutions (D Conv) with the corresponding dimensionality. The two branches can be written as

$$\boldsymbol{f}_{\text{ins}} = \text{DFA_Conv}_1\left(\text{Squeeze}(\text{LN}(\text{DS}(\boldsymbol{f})))\right) \tag{5}$$

$$\boldsymbol{f}_{\text{k}} = \text{DFA_Conv}_{k,w}\left(\text{LN}(\text{DS}(\boldsymbol{f}))\right), \tag{6}$$

where the DS is the down-sampling achieved by max-pooling with the scale of 2. LN is the Layer normalization [3]. The Squeeze is achieved by average pooling at the channel dimension. w is the factor to expand the window size of the convolution to $w * (k + 1)$, which enables the module can learn a the feature

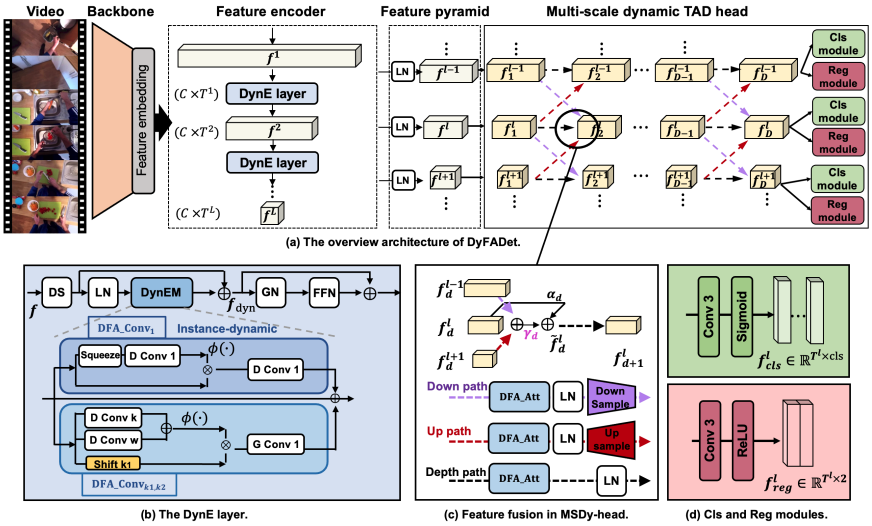


Fig. 4: (a) Overview of DyFADet. (b) The DynE layer consisting of the feature encoder. GN is Group-normalization [51]. (c) The multi-scale feature fusion in DyHead. (d) The classification and regression module obtains the classification and boundary results.

with long-term information. Then the output features of the DynE module can be represented by

$$\boldsymbol{f}_{\text{dyn}} = \boldsymbol{f}_{\text{w}} + \boldsymbol{f}_{\text{ins}} + \text{DS}(\boldsymbol{f}). \tag{7}$$

Overall, in the feature encoder, each DynE layer will down-sample the feature with a scale of 2 to generate the representations with different temporal resolutions. The dynamic feature selection ability of DynE layers will guarantee the discriminative information of the obtained representations and alleviate the less-discriminant feature problem, which leads to better TAD performance.

Dynamic TAD head. The shared-weight static detection heads in common TAD models can have the head inadaptation issue, which means that the optimal weights for detecting long-range and short-range action instances can be different, resulting in the sub-optimal detection performance. Even when implemented with the cross-level feature fusion, the heads only show limited improvement. Intuitively, fusing the features from a higher scale helps the head to explore more global long-term information. Exploring the information from a lower scale can enable the head to find more boundary details. We infer that such a multi-scale fusion can benefit the detection performance only if these intrinsic representations are properly selected from the adjacent scale. These motivate us to build a novel DFA, which can dynamically adjust the shared parameters based on inputs and selectively fuse the cross-level features for better performance.

The architecture of a DyHead is illustrated in Fig. 4 (a). Both the features from the corresponding and the adjacent levels in the pyramid are sent to the

DyHead. By implementing the dynamic feature learning mechanism with the proposed DFA, the head parameters can adjust based on inputs, and important representations will be selectively fused. Specifically, suppose that the depth of the head is D, the output features $\boldsymbol{f}_{d+1}^{l}$ at the l-th level can be calculated as the accumulation of $\boldsymbol{f}_{d}^{l-1}$, $\boldsymbol{f}_{d}^{l}$ and $\boldsymbol{f}_{d}^{l+1}$, which are processed by three different paths (shown in Fig. 4 (c)). The down path and up path are built based on the DFA with an additional LN and down-sampling (DS) or up-sampling (US) interpolation module with a scale of 2. The features will be first fused by

$$\begin{aligned}\tilde{\boldsymbol{f}}_{d}^{l} = \gamma_d \cdot \Big(& \mathrm{DS}(\ \mathrm{DFA_Att}(\boldsymbol{f}_{d}^{l-1})\ ) + \\ & \mathrm{US}(\ \mathrm{DFA_Att}(\boldsymbol{f}_{d}^{l+1})\ )\Big) + \alpha_d \cdot \boldsymbol{f}_{d}^{l},\end{aligned} \tag{8}$$

where γ_d and α_d are two learnable factors. Then the resultant feature will be processed by the depth path by

$$\boldsymbol{f}_{d+1}^{l} = \mathrm{DFA_Att}(\ \tilde{\boldsymbol{f}}_{d}^{l}\ ). \tag{9}$$

The proposed DyHead With the multi-scale dynamic feature fusion procedure, the final features at depth D, namely, $\boldsymbol{f}_{D}^{l}$, $l = 1, ..., L$, will be used to detect the action instances with different temporal ranges.

Classification and regression modules. The final classification (Cls) and regression (Reg) modules are designed to process final features across all levels. The cls module is realized by using a 1D convolution with a Sigmoid function to predict the probability of action categories at each time stamp. The Reg module is implemented using a 1D convolution with a ReLU to estimate the distances $\{d_m^s, d_m^e\}$ from the timestamp t to the action start and end $\{t_m^s, t_m^e\}$ for m-th action instance, which can be obtained by $t_m^s = t - d_m^s$ and $t_m^e = t + d_m^e$.

3.3 Training and Inference

Training. Follow [59], the center sampling strategy [59] is used during training, which means that the instants around the center of an action instance are labeled as positive during training. The loss function of the proposed DyFADet follow the simple design in [59], which has two different terms: (1) $\mathcal{L}_{\text{cls}}$ is a focal loss [29] for classification; (2) $\mathcal{L}_{\text{reg}}$ is a DIoU loss [63] for distance regression. The loss function is then defined as

$$\mathcal{L} = \sum_{t,l} \big(\mathcal{L}_{cls} + \lambda_{\text{reg}} \mathbb{I}_{ct}\ \mathcal{L}_{reg}\big)/T_{pos}, \tag{10}$$

where T_{pos} is the number of positive timestamps and $\mathbb{I}_{ct}$ is an indicator function that denotes if a time step t is within the sampling center of an action instance.

Inference. During inference, if a classification score is higher than the detection threshold, the instant will be kept. Then, Soft-NMS [4] will be further applied to remove the repeated detected instances.

4 Experiments

4.1 Experimental settings

Datasets. Six TAD datasets, including HACS-Segment [61], THUMOS14 [22], ActivityNet-1.3 [5], Epic-Kitchen 100 [13], Ego4D-Moment Queries v1.0 (Ego4D-MQ1.0) [15] and FineAction [32], are used in our experiments. ActivityNet-1.3 and HACS are two large-scale datasets with 200 classes of action, containing 10,024 and 37,613 videos for training, 4,926 and 5,981 videos for tests. THUMOS14 consists of 20 sport action classes and contains 200 and 213 untrimmed videos with 3,007 and 3,358 action instances on the training and test set, respectively. The Epic-Kitchen 100 and Ego4D-MQ1.0 are two datasets in first-person vision. Epic-Kitchen 100 has two sub-tasks: noun and verb localization, containing 495 and 138 videos with 67,217 and 9,668 action instances for training and testing, respectively. Ego4D-MQ1.0 has 2,488 video clips and 22.2K action instances from 110 pre-defined action categories, which are densely labeled. FineAction contains 103K temporal instances of 106 fine-grained action categories, annotated in 17K untrimmed videos.

Table 1: Results on HACS-segment.

Method	Backbone	0.5	0.75	0.95	Avg.
SSN [62]	I3D	28.8	18.8	5.3	19.0
LoFi [53]	TSM	37.8	24.4	7.3	24.6
G-TAD [54]	I3D	41.1	27.6	8.3	27.5
TadTR [31]	I3D	47.1	32.1	10.9	32.1
BMN [27]	SlowFast	52.5	36.4	10.4	35.8
ActionFormer [59]	SlowFast	54.9	36.9	9.5	36.4
TALLFormer [10]	Swin	55.0	36.1	11.8	36.5
TCANet [37]	SlowFast	54.1	37.2	11.3	36.8
TriDet [39]	SlowFast	56.7	39.3	11.7	38.6
TriDet [39]	VM2-g	62.4	44.1	13.1	43.1
Ours	SlowFast	57.8	39.8	11.8	39.2
Ours	VM2-g	**64.0**	**44.8**	**14.1**	**44.3**

Evaluation metric and experimental implementation. The standard mean average precision (mAP) at different temporal intersection over union (tIoU) thresholds will be reported as evaluation metric in the experiments. We follow the practice in [39,59] that uses off-the-shelf pre-extracted features as input. Our method is trained with AdamW [33] with warming-up. More training details are provided in the supplementary materials.

Detailed architecture of DyFADet. We used 2 convolutions for feature embedding, 7 DynE layers as the encoder, and separate DyHeads for classification and regression as the detectors. In our experiments, we report the best results of DyFADet with different architectures. A comprehensive ablation study about the architecture is also provided in Section 4.3.

4.2 Main results

HACS. The performance of different TAD methods on the HACS dataset is provided in Table 1 and the best and the second best performance are denoted

Table 2: Comparison with the SOTA methods on THUMOS14 and ActivityNet-1.3. TSN [46], I3D [7], Swin Transformer (Swin) [10] and TSP (R(2+1)D) [2] features are used. The best and the second best performance are denoted by **bold** and blue.

Method	THUMOS14							ActivityNet-1.3				
	Features	0.3	0.4	0.5	0.6	0.7	Avg.	Features	0.5	0.75	0.95	Avg.
BMN [27]	TSN	56.0	47.4	38.8	29.7	20.5	38.5	TSN	50.1	34.8	8.3	33.9
G-TAD [54]	TSN	54.5	47.6	40.3	30.8	23.4	39.3	TSN	50.4	34.6	9.0	34.1
A2Net [57]	I3D	58.6	54.1	45.5	32.5	17.2	41.6	I3D	43.6	28.7	3.7	27.8
PBRNet [30]	I3D	58.5	54.6	51.3	41.8	29.5	-	I3D	54.0	35.0	9.0	35.0
TCANet [37]	TSN	60.6	53.2	44.6	36.8	26.7	44.3	TSN	52.3	36.7	6.9	35.5
RTD-Net [43]	I3D	68.3	62.3	51.9	38.8	23.7	49.0	I3D	47.2	30.7	8.6	30.8
VSGN [60]	TSN	66.7	60.4	52.4	41.0	30.4	50.2	I3D	52.3	35.2	8.3	34.7
AFSD [25]	I3D	67.3	62.4	55.5	43.7	31.1	52.0	I3D	52.4	35.2	6.5	34.3
ReAct [40]	TSN	69.2	65.0	57.1	47.8	35.6	55.0	TSN	49.6	33.0	8.6	32.6
TadTR [31]	I3D	74.8	69.1	60.1	46.6	32.8	56.7	TSN	51.3	35.0	9.5	34.6
TALLFormer [10]	Swin	76.0	-	63.2	-	34.5	59.2	Swin	54.1	36.2	7.9	35.6
ActionFormer [59]	I3D	82.1	77.8	71.0	59.4	43.9	66.8	R(2+1)D	54.7	37.8	**8.4**	36.6
ASL [38]	I3D	83.1	79.0	71.7	59.7	45.8	67.9	I3D	54.1	37.4	8.0	36.2
TriDet [39]	I3D	83.6	**80.1**	**72.9**	**62.4**	47.4	**69.3**	R(2+1)D	54.7	38.0	**8.4**	36.8
Ours	I3D	**84.0**	**80.1**	72.7	61.1	**47.9**	69.2	R(2+1)D	**58.1**	**39.6**	**8.4**	**38.5**

by bold and blue. In our experiments, the SlowFast [14] features are used for the proposed DyFADet in TAD tasks on HACS.

From the results, we see that our method with SlowFast features outperforms all other evaluated methods in terms of average mAP (39.2%), and also achieves the best performance across all tIoUs thresholds. Notably, with tIoU = 0.5, the DyFADet surpasses the Tridet by 1.1%. As a TAD model can generally benefit from a more advanced backbone, we further implement VideoMAE V2-Gaint [45] (VM2-g) to conduct TAD on HACS. Remarkably, our method achieves the highest performance with VideoMAE V2-Gaint and beats the previous SOTA TriDet (VM2-g) with the new SOTA result on HACS, 44.3%.

THUMOS14 and ActivityNet-1.3. The experimental results which compare the performance of ours and other TAD models are shown in Table 2. Average mAP in [0.3:0.1:0.7] and [0.5:0.05:0.95] are reported on THUMOS14 and ActivityNet-1.3, respectively. In the experiments, our method conducts the TAD task based on I3D and R(2+1)D features for the THUMOS14 and ActivityNet-1.3 datasets. We see that our method with I3D features achieves the mAP of 69.2%, which is competitive to TriDet [39], while significantly outperforming all other related TAD methods on THUMOS14. On ActivityNet-1.3, the proposed DyFADet significantly surpasses the TriDet by about 1.7%, and achieves the mAP of 38.5%. The high performance of the proposed DyFADet indicates the effectiveness of dynamic feature learning mechanisms in modern TAD methods.

Table 3: Results on THUMOS14 using VM2-g.

Method	Backbone	0.3	0.55	0.7	Avg.
ActionFormer [59]	VM2-g	84.0	73.0	47.7	69.6
TriDet [39]	VM2-g	84.8	73.3	48.8	70.1
Ours	VM2-g	84.3	73.7	**50.2**	70.5
Ours	VM2-g+F	**85.4**	**74.0**	**50.2**	**71.1**

Table 4: Results on FineAction.

Method	Backbone	0.5	0.75	0.95	Avg.
BMN	I3D	14.4	8.9	3.1	9.3
G-TAD	I3D	13.7	8.8	3.1	9.1
ActionFormer	InternVideo	-	-	-	17.6
ActionFormer	VM2-g	29.1	17.7	5.1	18.2
Ours	VM2-g	**37.1**	**23.7**	**5.9**	**23.8**

Generally, a TAD model can benefit from a more advanced backbone. Therefore, we further implement VideoMAE V2-Gaint [45] (VM2-g) on THUMOS14. We see that all TAD methods achieve significant improvements with the advanced backbone, VM2-g. Our method can achieve the mAP of 70.5%, which is superior to TriDet and ActionFormer. Also, the detection performance can be boosted to 71.1% if we additionally use the optic flow (F) features.

FineAction. We report the performance of the different popular TAD methods including BMN [27], G-TAD [54], ActionFormer [59], and our method on TAD task. Moreover, I3D [46], InternVideo [47] and and VM2-g [45] are used to extract the off-line features and the average mAP in [0.50:0.05:0.95] is reported for all methods. The experimental results are provided in Tab.4. The experimental results show that our method outperforms other TAD models and achieve a new SOTA result on FineAction, 23.8%, with the VideoMAEv2-giant.

Epic-Kitchen 100 and Ego4D-MQ1.0. The evaluations are also conducted on two large-scale egocentric datasets, which are shown in Table 5 and Table 6, respectively. For Epic-Kitchen 100, the average mAP in [0.1:0.1:0.5] is reported and all methods use the SlowFast features. We see that our method has the best performance across all tIoU thresholds on both subsets and achieves an average mAP of 25.0% and 23.4% for verb and noun subsets, respectively, which are significantly superior to the strong performance of the recent TAD methods, including Actionformer [59], ASL [38]. For Ego4D-MQ1.0, two types of features, including SlowFast (SF) and EgoVLP (EV) features are used in the experiments. With SlowFast features, the proposed method achieves the mAP of 15.3%, which significantly outperforms the Actionformer. Moreover, we see that using or combining the features with advanced backbone models, such as EgoVLP [26], can further boost the performance of our method by a large margin. SF and EV denote Slowfast [14] and EgoVLP [26] features.

4.3 Ablation study

DFA modules in TAD. The experiments investigate the effectiveness of different dynamic feature aggregation modules in TAD tasks. The results are shown in Table 7, where different implementations of the encoder and the detection head are evaluated.

Table 5: Results on Epic-Kitchen 100.

	Method	0.1	0.2	0.3	0.4	0.5	Avg.
V.	G-TAD [54]	12.1	11.0	9.4	8.1	6.5	9.4
	ActionFormer [59]	26.6	25.4	24.2	22.3	19.1	23.5
	ASL [38]	27.9	-	25.5	-	19.8	24.6
	Ours	**28.0**	**27.0**	**25.6**	**23.5**	**20.8**	**25.0**
N.	G-TAD [54]	11.0	10.0	8.6	7.0	5.4	8.4
	ActionFormer [59]	25.2	24.1	22.7	20.5	17.0	21.9
	ASL [38]	26.0	-	23.4	-	17.7	22.6
	Ours	**26.8**	**26.0**	**24.1**	**21.9**	**18.5**	**23.4**

Table 6: Results on Ego4D-MQ1.0.

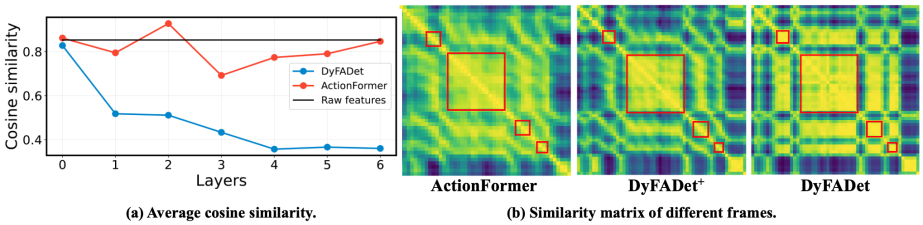
Method	Features	0.1	0.3	0.5	Avg.
VSGN [60]	SF	9.1	5.8	3.4	6.0
VSGN [26]	EV	16.6	11.5	6.6	11.4
ActionFormer [59]	SF	20.1	14.4	10.0	14.9
ActionFormer [59]	EV	26.9	20.2	13.7	20.3
ActionFormer [59]	EV+SF	28.0	21.2	15.6	21.6
Ours	SF	19.0	15.0	11.2	15.3
Ours	EV	28.4	22.1	16.2	22.2
Ours	EV+SF	**28.8**	**22.6**	**16.9**	**22.8**

The baseline model is realized by an all-convolution TAD model, which achieved the mAP of 62.1%. We further use different dynamic modules to substitute or improve the convolutions in the encoder as the comparison, such as the deformable 1-d convolution [23], squeeze-and-excitation module [19], Dynamic convolution [64] and temporal deformable 3d convolution module [24]. All dynamic modules achieve better performance than the baseline model with convolution, indicating the strong ability of dynamic modules in TAD tasks. While, due to the stronger adaptation ability of DFA, the DyFADet* substituting the convolutions with DFA can achieve the performance equaling to the recent strong TAD model, Actionformer. Moreover, using the proposed DynE layer (DyFADet†) further increases the final performance by 1.0%. For the detection head, we see that applying the multi-scale connection in the TAD head can improve the final detection performance. However, naively using the convolution to connect different scales (DyFADet‡) only results in limited improvements. While, after being equipped with DyHead, the DyFADet can achieve a performance of 69.2%, outperforming all other models in the experiments. A more comprehensive study about the architecture for DyFADet is provided in supplementary materials.

Table 7: Results with different modules.

Method	Encoder	MS-head	Avg.
Baseline	Conv	✖	62.1
Baseline	DeformConv [23]	✖	66.1
Baseline	SE [19]	✖	63.4
Baseline	Dyn Conv [64]	✖	66.7
Baseline	TD3d Conv [24]	✖	66.5
DyFADet*	DFA_Conv	✖	66.8
ActionFormer	SA	✖	66.8
DyFADet†	DynE	✖	67.8
DyFADet‡	Conv	Dyn	67.9
DyFADet‡	DynE	Conv	68.0
DyFADet	DynE	Dyn	**69.2**

The discriminability of the learned features. As shown in [39, 44], the features obtained by recent TAD methods [49, 59] tend to exhibit high similarities between snippets, which leads to the less-discriminant feature problem and be harmful to TAD performance. In Fig. 5(a), we perform statistics of the average cosine similarity between each feature at different timestamps. We observe that the features in Actionformer exhibit high similarity, indicating poor discriminability. Using the DynE based encoder can address the issue and therefore improve the detection performance. Moreover, we further provide the feature



(a) Average cosine similarity. (b) Similarity matrix of different frames.

Fig. 5: (a) The average cosine similarity between features at different timestamps in the same level among each encoder layer. (b) Similarity matrix of the extracted features among timestamps.

similarity matrix between the different timestamps in Fig.5(b), where the red boxes exhibit the action intervals. The darker color means the features are more discriminant and share less similarity. From the result, we see that the inter-class features from our method within the red boxes show strong similarities, resulting in that the boundary features are distinctive and can be easily extracted. While Actionformer fails to explore the discriminant features during feature learning. From the results, our method addresses the mentioned first issue in Fig.1 based on the more discriminant learned features in DyFADet†, which can be further enhanced by using the proposed DyFADet as TAD model.

Visualization of DFA in DyHead. In Fig. 6 (a), we provide the visualization results of the proposed DFA. For better visualization, we use the DyFADet with $C_m = k$ in Eq.(2), meaning that the same receptive fields will be shared among channels while varying at different timestamps. From the results we see that the DFA can adapt the re-weight masks based on inputs, leading to different formations at different timestamps. Noting that in classical TAD models, such as Actionformer [59], the parameters of the detection head are shared among different levels, which might be harmful in detection. While, the dynamic properties and multi-scale fusion of the DyFADet specialize the detection head for inputs and target action instances, leading to a fine-grained dynamic action detection manner which can achieve better TAD performance.

Moreover, we visualize the average activated rate at each path for the whole video in DyHead (shown in Fig. 6 (b)). The results show that the proposed DyHead achieves the dynamic routing-like feature learning mechanism similar to [41]. However, our method can simultaneously adapt the receptive fields and the kernel weights, which improves the model ability in TAD tasks. More visualization results can be found in supplementary materials.

Hyper-parameters. In Table 8 , we further evaluate the performance of DyFADet on THUMOS14 with different hyper-parameters, including the expanded factor, w, the layer number of the detection head, and the dynamic type of the proposed module. We observe the model works best with $D = 3$ and the w should

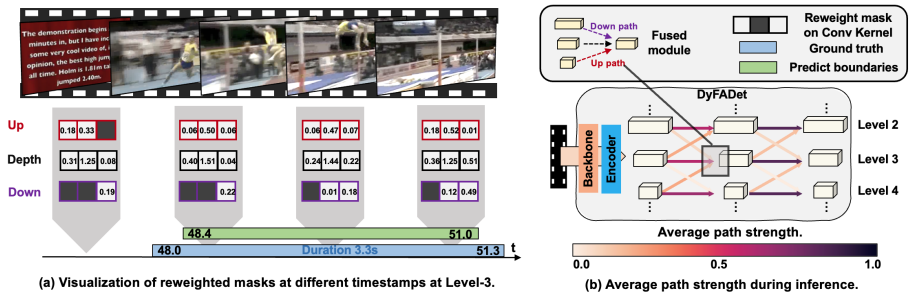


Fig. 6: Visualization when DyFADet detecting action High Jump. (a) the DFA reweight masks at different timestamps at Level-3 in DyHead. (b) Average path strength during inference of the whole video.

Table 8: Different hyper-parameters.

w	mAP	D	mAP	DynType	mAP
3	68.2	2	67.4	C	68.4
5	69.2	3	69.2	K	69.2
6	67.0	4	67.1	CK	68.4

Table 9: Average latency on THUMOS14.

Method	AP		Latency		Params
	0.5	Avg.	CPU(s)	GPU(ms)	(M)
ActionFormer	71.0	66.8	0.57	68.4	29.2
DyFADet†	71.7	67.8	**0.16**	**44.7**	12.7
DyFADet	**72.7**	**69.2**	0.49	58.7	18.8

be selected for different datasets. Moreover, although the dynamic type can affect the final performance, DyFADet with each dynamic type achieves higher performance compared to most TAD methods in Table 2.

Latency. We test the average latency for single video inference on GeForce RTX 4090 GPU on THUMOS14. As shown in Table 9, DyFADet† is faster than ActionFormer while has better detection performance. Moreover, DyFADet can be improved by the DyHead which further brings 1.4% average mAP improvement and the latency is still comparable to Actionformer.

5 Conclusion

In this paper, we introduced a novel DFA module simultaneously adapting its kernel weights and receptive fields, to address the less-discriminant feature and head inadaptation issues in TAD models. The proposed DyFADet based on DFA achieves high performance on a series of TAD benchmarks, which indicates that an input-based fine-grained feature extraction mechanism should be considered for building high-performance TAD models. For future works, we believe that the efficiency of DyFADet can be further improved by combining the sparse convolution [11] and adding additional constraints to encourage each DFA to mask as many features as possible with a minor performance penalty. The applications of DFA in more video-understanding tasks will be further investigated.

Acknowledgement. This work is supported in part by National Natural Science Foundation of China under Grants 62206215, China Postdoctoral Science Foundation under Grants 2022M712537, China National Postdoctoral Program for Innovative Talents BX2021241, and CCF-BAIDUOF 2021024.

References

1. Agarap, A.F.: Deep learning using rectified linear units (ReLU). arXiv preprint arXiv:1803.08375 (2018)
2. Alwassel, H., Giancola, S., Ghanem, B.: Tsp: Temporally-sensitive pretraining of video encoders for localization tasks. In: ICCV (2021)
3. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
4. Bodla, N., Singh, B., Chellappa, R., Davis, L.S.: Soft-nms–improving object detection with one line of code. In: ICCV (2017)
5. Caba Heilbron, F., Escorcia, V., Ghanem, B., Carlos Niebles, J.: Activitynet: A large-scale video benchmark for human activity understanding. In: CVPR (2015)
6. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (2020)
7. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: CVPR (2017)
8. Chen, G., Zheng, Y.D., Wang, L., Lu, T.: Dcan: improving temporal action detection via dual context aggregation. In: AAAI (2022)
9. Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., Liu, Z.: Dynamic convolution: Attention over convolution kernels. In: CVPR (2020)
10. Cheng, F., Bertasius, G.: Tallformer: Temporal action localization with a long-memory transformer. In: ECCV (2022)
11. Contributors, S.: Spconv: Spatially sparse convolution library. `https://github.com/traveller59/spconv` (2022)
12. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: ICCV (2017)
13. Damen, D., Doughty, H., Farinella, G.M., Furnari, A., Kazakos, E., Ma, J., Moltisanti, D., Munro, J., Perrett, T., Price, W., et al.: Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. IJCV (2022)
14. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: ICCV (2019)
15. Grauman, K., Westbury, A., Byrne, E., Chavis, Z., Furnari, A., Girdhar, R., Hamburger, J., Jiang, H., Liu, M., Liu, X., et al.: Ego4d: Around the world in 3,000 hours of egocentric video. In: CVPR (2022)
16. Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y.: Dynamic neural networks: A survey. IEEE TPAMI (2021)
17. Han, Y., Liu, Z., Yuan, Z., Pu, Y., Wang, C., Song, S., Huang, G.: Latency-aware unified dynamic networks for efficient image recognition. IEEE TPAMI (2024)
18. Han, Y., Yuan, Z., Pu, Y., Xue, C., Song, S., Sun, G., Huang, G.: Latency-aware spatial-wise dynamic networks. NeurIPS (2022)
19. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR (2018)
20. Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K.: Multi-scale dense networks for resource efficient image classification. In: ICLR (2018)

21. Huang, G., Wang, Y., Lv, K., Jiang, H., Huang, W., Qi, P., Song, S.: Glance and focus networks for dynamic visual recognition. IEEE TPAMI (2023)
22. Jiang, Y.G., Liu, J., Roshan Zamir, A., Toderici, G., Laptev, I., Shah, M., Sukthankar, R.: THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/THUMOS14/` (2014)
23. Lei, P., Todorovic, S.: Temporal deformable residual networks for action segmentation in videos. In: CVPR (2018)
24. Li, J., Liu, X., Zhang, M., Wang, D.: Spatio-temporal deformable 3d convnets with attention for action recognition. PR **98**, 107037 (2020)
25. Lin, C., Xu, C., Luo, D., Wang, Y., Tai, Y., Wang, C., Li, J., Huang, F., Fu, Y.: Learning salient boundary feature for anchor-free temporal action localization. In: CVPR (2021)
26. Lin, K.Q., Wang, J., Soldan, M., Wray, M., Yan, R., XU, E.Z., Gao, D., Tu, R.C., Zhao, W., Kong, W., et al.: Egocentric video-language pretraining. NeurIPS (2022)
27. Lin, T., Liu, X., Li, X., Ding, E., Wen, S.: Bmn: Boundary-matching network for temporal action proposal generation. In: ICCV (2019)
28. Lin, T., Zhao, X., Su, H., Wang, C., Yang, M.: Bsn: Boundary sensitive network for temporal action proposal generation. In: ECCV (2018)
29. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017)
30. Liu, Q., Wang, Z.: Progressive boundary refinement network for temporal action detection. In: AAAI (2020)
31. Liu, X., Wang, Q., Hu, Y., Tang, X., Zhang, S., Bai, S., Bai, X.: End-to-end temporal action detection with transformer. IEEE TIP (2022)
32. Liu, Y., Wang, L., Wang, Y., Ma, X., Qiao, Y.: Fineaction: A fine-grained video dataset for temporal action localization. IEEE TIP (2022)
33. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
34. Ma, N., Zhang, X., Huang, J., Sun, J.: Weightnet: Revisiting the design space of weight networks. In: ECCV (2020)
35. Mac, K.N.C., Joshi, D., Yeh, R.A., Xiong, J., Feris, R.S., Do, M.N.: Learning motion in feature space: Locally-consistent deformable convolution networks for fine-grained action detection. In: ICCV (2019)
36. Meng, Y., Lin, C.C., Panda, R., Sattigeri, P., Karlinsky, L., Oliva, A., Saenko, K., Feris, R.: Ar-net: Adaptive frame resolution for efficient action recognition. In: ECCV (2020)
37. Qing, Z., Su, H., Gan, W., Wang, D., Wu, W., Wang, X., Qiao, Y., Yan, J., Gao, C., Sang, N.: Temporal context aggregation network for temporal action proposal refinement. In: CVPR (2021)
38. Shao, J., Wang, X., Quan, R., Zheng, J., Yang, J., Yang, Y.: Action sensitivity learning for temporal action localization. In: ICCV (2023)
39. Shi, D., Zhong, Y., Cao, Q., Ma, L., Li, J., Tao, D.: Tridet: Temporal action detection with relative boundary modeling. In: CVPR (2023)
40. Shi, D., Zhong, Y., Cao, Q., Zhang, J., Ma, L., Li, J., Tao, D.: React: Temporal action detection with relational queries. In: ECCV (2022)
41. Song, L., Li, Y., Jiang, Z., Li, Z., Sun, H., Sun, J., Zheng, N.: Fine-grained dynamic head for object detection. NeurIPS (2020)
42. Sridhar, D., Quader, N., Muralidharan, S., Li, Y., Dai, P., Lu, J.: Class semantics-based attention for action detection. In: ICCV (2021)
43. Tan, J., Tang, J., Wang, L., Wu, G.: Relaxed transformer decoders for direct action proposal generation. In: ICCV (2021)

44. Tang, T.N., Kim, K., Sohn, K.: Temporalmaxer: Maximize temporal context with only max pooling for temporal action localization. arXiv preprint arXiv:2303.09055 (2023)
45. Wang, L., Huang, B., Zhao, Z., Tong, Z., He, Y., Wang, Y., Wang, Y., Qiao, Y.: Videomae v2: Scaling video masked autoencoders with dual masking. In: CVPR (2023)
46. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: ECCV (2016)
47. Wang, Y., Li, K., Li, Y., He, Y., Huang, B., Zhao, Z., Zhang, H., Xu, J., Liu, Y., Wang, Z., Xing, S., Chen, G., Pan, J., Yu, J., Wang, Y., Wang, L., Qiao, Y.: Internvideo: General video foundation models via generative and discriminative learning. arXiv preprint arXiv:2212.03191 (2022)
48. Wang, Y., Chen, Z., Jiang, H., Song, S., Han, Y., Huang, G.: Adaptive focus for efficient video recognition. In: ICCV (2021)
49. Weng, Y., Pan, Z., Han, M., Chang, X., Zhuang, B.: An efficient spatio-temporal pyramid transformer for action detection. In: ECCV (2022)
50. Wu, B., Wan, A., Yue, X., Jin, P., Zhao, S., Golmant, N., Gholaminejad, A., Gonzalez, J., Keutzer, K.: Shift: A zero flop, zero parameter alternative to spatial convolutions. In: CVPR (2018)
51. Wu, Y., He, K.: Group normalization. In: ECCV (2018)
52. Wu, Z., Xiong, C., Ma, C.Y., Socher, R., Davis, L.S.: Adaframe: Adaptive frame selection for fast video recognition. In: CVPR (2019)
53. Xu, M., Perez Rua, J.M., Zhu, X., Ghanem, B., Martinez, B.: Low-fidelity video encoder optimization for temporal action localization. In: NeurIPS (2021)
54. Xu, M., Zhao, C., Rojas, D.S., Thabet, A., Ghanem, B.: G-tad: Sub-graph localization for temporal action detection. In: CVPR (2020)
55. Yang, B., Bender, G., Le, Q.V., Ngiam, J.: Condconv: Conditionally parameterized convolutions for efficient inference. NeurIPS (2019)
56. Yang, L., Han, Y., Chen, X., Song, S., Dai, J., Huang, G.: Resolution adaptive networks for efficient inference. In: CVPR (2020)
57. Yang, L., Peng, H., Zhang, D., Fu, J., Han, J.: Revisiting anchor mechanisms for temporal action localization. IEEE TIP (2020)
58. Yang, M., Chen, G., Zheng, Y.D., Lu, T., Wang, L.: Basictad: an astounding rgb-only baseline for temporal action detection. Computer Vision and Image Understanding (2023)
59. Zhang, C.L., Wu, J., Li, Y.: Actionformer: Localizing moments of actions with transformers. In: ECCV (2022)
60. Zhao, C., Thabet, A.K., Ghanem, B.: Video self-stitching graph network for temporal action localization. In: ICCV (2021)
61. Zhao, H., Torralba, A., Torresani, L., Yan, Z.: Hacs: Human action clips and segments dataset for recognition and temporal localization. In: ICCV (2019)
62. Zhao, Y., Xiong, Y., Wang, L., Wu, Z., Tang, X., Lin, D.: Temporal action detection with structured segment networks. In: ICCV (2017)
63. Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., Ren, D.: Distance-iou loss: Faster and better learning for bounding box regression. In: AAAI (2020)
64. Zhou, C., Loy, C.C., Dai, B.: Interpret vision transformers as convnets with dynamic convolutions. arXiv preprint arXiv:2309.10713 (2023)
65. Zhu, Z., Tang, W., Wang, L., Zheng, N., Hua, G.: Enriching local and global contexts for temporal action localization. In: ICCV (2021)