Classification Matters: Improving Video Action Detection with Class-Specific Attention

Jinsung Lee[1]*, Taeoh Kim[2], Inwoong Lee[2], Minho Shim[2], Dongyoon Wee[2], Minsu Cho[1], and Suha Kwak[1]

[1] Pohang University of Science and Technology (POSTECH), South Korea
[2] NAVER Cloud, South Korea
https://jinsingsangsung.github.io/ClassificationMatters/

Abstract. Video action detection (VAD) aims to detect actors and classify their actions in a video. We figure that VAD suffers more from classification rather than localization of actors. Hence, we analyze how prevailing methods form features for classification and find that they prioritize actor regions, yet often overlooking the essential contextual information necessary for accurate classification. Accordingly, we propose to reduce the bias toward actor and encourage paying attention to the context that is relevant to each action class. By assigning a class-dedicated query to each action class, our model can dynamically determine where to focus for effective classification. The proposed model demonstrates superior performance on three challenging benchmarks with significantly fewer parameters and less computation.

Keywords: Video action detection · Video transformer

1 Introduction

Video action detection (VAD) is the task of identifying actors and categorizing their activities in a video. It has recently attracted increasing attention due to its broad range of applications, such as surveillance video analysis or sports activity recognition. Since a video is a sequence of images, it is not surprising that solutions to video understanding tasks, including VAD, have been developed largely based on image recognition models. In particular, since VAD resembles object detection (OD) in an image, a large number of existing VAD models consider the task as an extension of OD and follow common OD pipelines accordingly [3, 5, 14, 22, 24, 26, 29, 30, 39, 41, 43].

Unfortunately, such straightforward extensions of OD are often not optimal for VAD due to the distinct nature of VAD from OD: *all instances conducting actions in VAD are humans*. Action localization in VAD focuses on identifying human-shaped objects only, making it considerably simpler than localizing arbitrary objects in OD [8]. In contrast, the classification of actions in VAD is significantly more difficult than that of OD. Unlike object classification in OD, which usually

*Work done while doing an internship at NAVER Cloud.

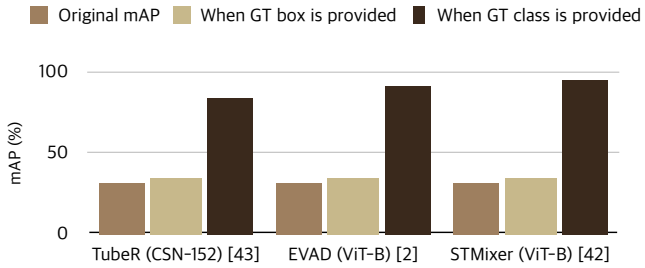


Fig. 1: Detection performance changes of the state-of-the-art methods (i.e., TubeR [43], EVAD [2], STMixer [42]) on AVA [8] when ground-truth boxes or class labels are given.

depends more on general appearances of objects, action classification in VAD requires identifying fine-grained details in both appearance and motion since different action classes are all conducted by humans, and thus, their general appearances are often not distinguishable clearly. We empirically verify the crucial role of classification in improving VAD performance. As shown in Fig. 1, for all the three latest VAD models we tested, providing ground-truth (GT) class labels consistently yields more substantial improvement than providing GT bounding boxes. This result implies that the room for improvement in VAD is mainly occupied by classification rather than localization. However, only a few have paid attention to such inherent challenge in classification for VAD [3, 8, 30].

In this work, we propose a new model architecture that aims to improve the classification performance for VAD. Our model first localizes each actor by attending features globally and then seeks local regions that are informative for identifying its action class. This procedure enables our model to actively focus on local regions that provide greater assistance in classification, such as fine details (*e.g.*, a cigarette for the *'smoke'* action class) or other people interacting with the actor (*e.g.*, a speaker for the *'listen to'* action class). To this end, we introduce *class queries*, each of which separately holds essential information about each action class. These class queries learn to identify if a specific action class takes place within the scene. Specifically, we first construct a feature map that encompasses the interaction between each actor and the global context. Then, each class query is learned to be highly similar with features on a particular region of the feature map relevant to the action class, so that it extracts rich and fine-grained features that are dedicated to each actor and each class. These features help the model identify details at a granular level and context necessary for action classification, while also offering interpretable attention maps for each class that support the model's decision-making concurrently. The consequent attention map illustrates the model's ability to capture details and interactions relevant to the action happening in the scene, and the model seeks over regions that are not bounded by the actor box. We additionally introduce components that notably enhance the utilization of class queries, aiding in capturing details and ensuring specificity to individual actors.

Our model is evaluated on three conventional benchmarks in the field: AVA [8], JHMDB51-21 [11], and UCF101-24 [28]. Among the models with comparable

backbones, our model shows superior performance while being more efficient. Current best performing models [2, 42] predict an instance per frame given a clip of multiple frames, and thus adopt the sliding window strategy to localize an actor in a whole video. In contrast, our model constructs the entire spatio-temporal tube of an actor through a single feed-forward pass, which better aligns with the objective of VAD and also leads to higher computational efficiency, in particular when dealing with longer video clips.

In this paper, we present four key contributions.

- We analyze how existing methods in VAD process features for classification, and perform a detailed investigation into their problematic behaviors.
- We introduce a novel classification module, *Classifying Decoder Layer*, that effectively combines context, actor, and class queries to construct classification features for each action class.
- We provide additional components, *3D Deformable Transformer Encoder* and *Localizing Decoder Layer*, that augment our classification module and significantly boost the model's performance.
- Our model outperforms existing methods on challenging benchmarks with greater efficiency.

2 Related work

Video action detection. Video action detection involves analyzing a video clip and generating actor-specific spatio-temporal tubes while simultaneously predicting the actions being performed by the actors. Earlier studies [14,24,38] try to combine spatial and temporal information to obtain a good actor representation and pass it directly to box regression and classification layers. As advancements in object detection techniques continue, it has become common to employ 2D person detectors to extract actors from the scene and utilize these features for classification [5,6,32,35]. On the other side, a few studies [43] attempt to capture spatio-temporal tubes instead of detecting actors frame-by-frame. Our approach adopts the tube-based process not only to embrace the temporal property of the task but also to enhance the computational efficiency.

Transformer-based architectures. In the field of object detection, DETR [1] has suggested a framework that elevates cross-attention between queries and image features to capture both object location and its class effectively. With its notable performance and convenience, the advent of DETR has sparked the emergence of various adaptations [16, 21, 37, 44]. Among all, DAB-DETR [20] is the most relevant research to our work, as it introduces a modulating function that adjusts the positional information of queries. This approach enhances the cross-attention mechanism, allowing for a more comprehensive representation that encompasses the width and height of the anchor box prior. Our method utilizes this positional prior differently by offering class queries the clues for specifying actor, providing subtle guidance about which instance a class query should be referring to.

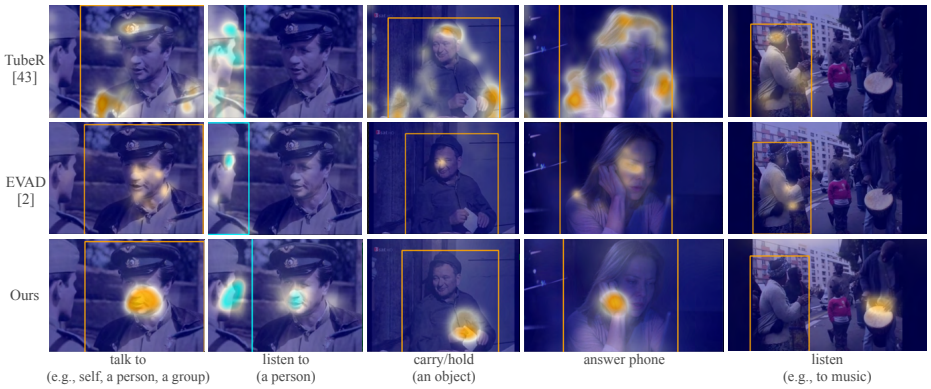


Fig. 2: Sample detection results and classification attention maps of the previous transformer-based model, TubeR [43], EVAD [2], and our model. Each attention map signifies the regions where the model attends to classify the action of the actor marked in the bounding box of the same color. Since our model creates an attention map for each class, we mark the corresponding label under the map. Best viewed in color.

Methods that tackle the classification problem. To achieve better action classification score, extensive research has been conducted on methods that consistently explore the relationship between the actor and the contextual information. In previous studies, this context has been categorized into two types: actor-actor relationship and actor-context relationship, and the methods used to establish these relationships often exhibit slight variations. In order to obtain the relationship between the two, either detected instances and the global features are concatenated [22, 30, 41] or fed to a transformer module [2, 10, 12, 31, 43]. While the transformer module is an advanced design to take instances' relations into account, previous methods based on transformers suffer from the problem where the classification process attends to the regions that are prone to becoming biased toward near the actor regions. In the subsequent section, we delve into a detailed discussion of this issue describing its reason and implications.

3 Background

Following the popular OD models [1, 16, 44], it has become a de-facto standard in VAD to employ the transformer architecture [34] in which query that represents an actor (referred to as the actor feature in VAD) gathers information from the video feature map (referred to as the context feature in VAD). In particular, recent VAD models with this architecture [2, 43] utilize transformers to search regions that are relevant to each actor, and use the resulting attention map to create their classification features.

However, this structure makes the classification of prevailing methods become biased towards the context features near actor's location. Since their output classification features are derived from a single attention map, all action classes share the same information of the context feature. Thus, instead of understand-

ing class-specific knowledge, the transformer weights are often trained only to embed commonly shared semantics across different classes. As mentioned in Sec. 1, action classes of VAD share an obvious semantic element, *the actor*, and this characteristic inevitably forces their models to include more information related to actor itself, leading to higher attention values near actor regions. Such classification feature may limit the model's observation scope to actor regions, which results in the model missing important regions that aid in classification.

The attention maps of TubeR [43] depicted in Fig. 2 clearly illustrate the identified issue: the attention is primarily concentrated on the actor's boundary regardless of the action the actor performs. Similarly, in the case of EVAD [2], the attention is mainly distributed over actor's face and body parts, yet critical regions essential for classification are overlooked. Furthermore, both prior methods fail to extend their attention beyond the actor's bounding box, where crucial contexts that provide clues to distinguish action classes are located. The activated regions may indicate the commonly shared semantics across different classes, but their lack of class-specificity leads to the model missing important clues for classification even when such clues are on the actor's body. Therefore, we aim to increase the class specificity of the classification feature and improve the classification performance.

4 Proposed method

In this section, we introduce our model that addresses the aforementioned challenges. Fig. 3 provides an overview of the model's architecture, consisting mainly of a backbone, transformer encoder, and transformer decoder. Given an input video clip $X \in \mathbb{R}^{T\times H_0\times W_0\times 3}$ of RGB frames and N_c action classes, our model operates as a function that takes X as input and outputs $\hat{Y} = \left\{\left(\hat{\mathbf{B}}_i, \hat{\mathbf{C}}_i\right) | \hat{\mathbf{B}}_i \in \mathbb{R}^{T\times 4}, \hat{\mathbf{C}}_i \in [0,1]^{T\times N_c},\ i \in [1, N_X]\right\}$, where (T, H_0, W_0) is the temporal length, height and width of the input video clip, $\left(\hat{\mathbf{B}}_i, \hat{\mathbf{C}}_i\right)$ represents a spatio-temporal tube and per-frame action class prediction for the i-th actor, and N_X is the total number of actors appearing in the video X.

The key component of the model lies in the classification module, which we denote as *Classifying Decoder Layer* (CDL). It distinguishes itself from other approaches through the utilization of *class queries*, which are learnable embeddings designed to encapsulate information specific to each class label. Class queries help the model's classification in two aspects. First, they allow transformer architecture to create more variation between features that represent different action classes and mitigates the issue of being biased towards common semantics (*i.e.*, the actor) of multiple classes. Thus, class queries provide opportunities to explore beyond actor locations (*e.g.*, *'listen to (a person)'* and *'listen (e.g., to music)'* in the third row of Fig. 2). Second, they acknowledge diverse characteristics of each class and grant the model more opportunities to browse over regions that are particularly conditioned to individual classes. More details into the use of class queries are presented in Sec. 4.2.

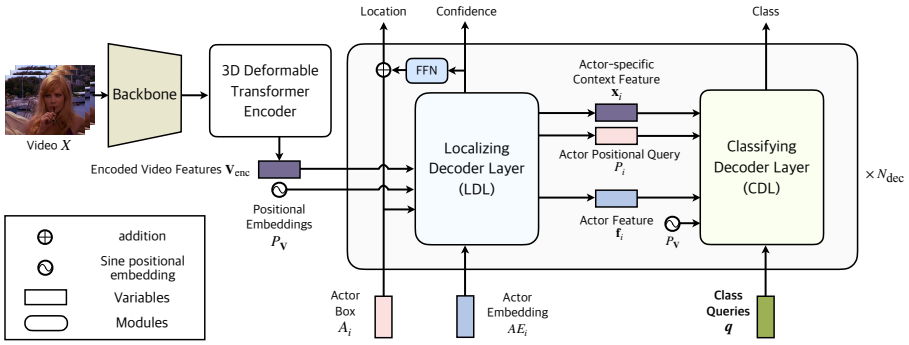


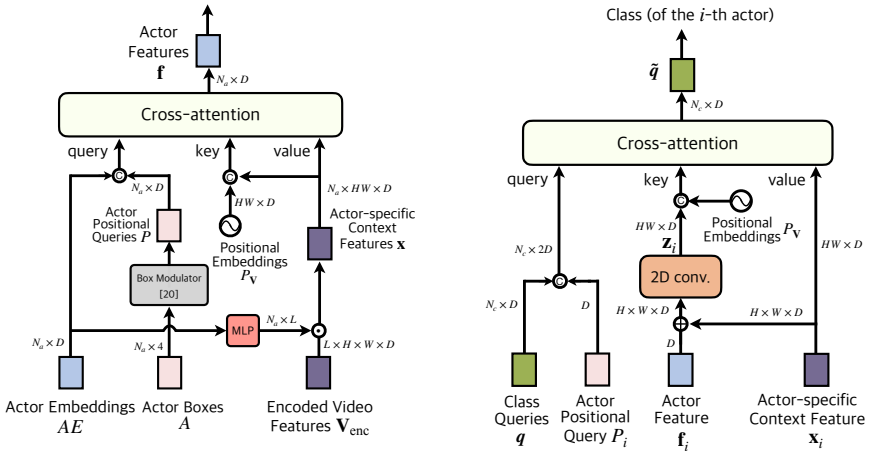
Fig. 3: Overview of the proposed model

CDL is strategically placed within the transformer decoder layers to enable access to features crucial for classification while allowing the acquisition of enriched information as layers are stacked. To ensure that CDL receives informative features, we introduce essential modifications to the conventional transformer encoder and decoder layers, hereby referred to as *3D Deformable Transformer Encoder* and *Localizing Decoder Layer* (LDL), respectively. In the following sections, we will delve into how each module operates, providing insights into their functions and contributions to the overall framework.

4.1 Backbone and transformer encoder

The input X is passed through the backbone network and 1D convolutions, and we integrate the outputs from intermediate layers to produce multi-scale feature maps $\mathbf{V} = \left\{ \boldsymbol{v}^l \in \mathbb{R}^{T_l \times H_l \times W_l \times D} \middle| \; l \in [1, L] \right\}$, where L is the number of feature map scales and D is the output channel dimension. Given the established benefits of utilizing multi-scale feature maps for effectively capturing various levels of semantics and details [9, 18, 23, 36], we leverage the multi-scale feature maps to achieve more precise classification. Accordingly, we modify the ordinary transformer encoder to efficiently handle these feature maps, which we dubbed *3D Deformable Transformer Encoder*. It processes the feature maps $\mathbf{V}$ and produces encoded feature maps $\mathbf{V}'_{\text{enc}}$ of the same shapes. To mitigate the computational memory demands associated with multi-scale spatio-temporal features, we draw inspiration from Deformable DETR [44], where a query gets encoded with the features gathered from distant points that are determined by offsets and weights generated by the query itself. Since the original work utilizes $(\Delta h, \Delta w)$-shaped 2-dimensional offsets for its encoder, we extend this offset to $(\Delta t, \Delta h, \Delta w)$ so that the query can be encoded with temporally distant features. A detailed logic of the encoder is described in Sec. C of the supplementary materials.

After $\mathbf{V}$ go through the encoder layers, the output feature maps $\mathbf{V}'_{\text{enc}}$ are scaled by interpolation to match the same spatio-temporal dimension across different levels: $\mathbf{V}_{\text{enc}} = \left\{ \boldsymbol{v}^l_{\text{enc}} \in \mathbb{R}^{T \times H \times W \times D} \middle| \; l \in [1, L] \right\}$. Note that the temporal dimension is recovered to the original length T in order to output boxes and class labels for each timestep.



(a) Localizing Decoder Layer (LDL) **(b)** Classifying Decoder Layer (CDL)

Fig. 4: Structure of transformer decoder layers of our model. We use ⓒ, $\odot$, and $\oplus$ to indicate concatenation, multiplication, and summation. In (b), we denote variables with the actor index i to describe the process simpler.

4.2 Transformer decoder

The decoder architecture serves as the fundamental design that embodies our objective of achieving improved classification. It is divided into two modules: localizing decoder layer (LDL) and classifying decoder layer (CDL). LDL specializes in gathering actor-related features from the encoded feature maps $\mathbf{V}_{\text{enc}}$ and constructs localization features. In contrast, CDL leverages the intermediate outputs of LDL, alongside class queries, to generate classification features. The subsequent sections elaborate on each module. Note that we simplify the notation by omitting the time index t and focus on how an individual frame is processed within the decoder layer.

Localizing decoder layer (LDL). LDL aims to construct features containing the information related to actors and provides informative features to CDL. In a nutshell, LDL resembles the decoder layer of DETR [1]: it consists of a cross-attention layer that takes learnable queries and the encoder output $\mathbf{V}_{\text{enc}}$ as its inputs to embed the actor information. Though, it differs from the original DETR decoder in two aspects. First, it constructs query and key by concatenating the *content* and *spatial* parts [21], where each part plays a different role in embedding actor's appearance feature and actor's positional feature, respectively. Such a design is essential in our model since class queries utilize actor-specific positional features for accurate classification, which we will describe in the next section. Second, it aggregates multi-scale feature maps to create feature maps that are specific to each actor. Due to the space limit, we briefly explain the roles and implications of LDL's input and output, and encourage readers to refer to Sec. D of the supplementary material for details.

We describe the structure of LDL in Fig. 4(a). Let N_a denote the number of actor candidates, then LDL takes as its query input $A \in \mathbb{R}^{N_a \times 4}$ and $AE \in \mathbb{R}^{N_a \times D}$, which we each denote as *actor box* and *actor embedding*. A and AE are utilized as the spatial part and the content part of the input query, and learn to embed appearance and positional information of actors, respectively. Thus, the output of LDL, which we denote as *actor features* $\mathbf{f} \in \mathbb{R}^{N_a \times D}$, results in containing the information related to each actor.

During its process, we transform A from coordinate space to D-dimensional space so that we can construct the *actor positional queries* P, which act as the spatial part of the input query while containing spatial information of the actor. At the same time, the multi-scale feature map $\mathbf{V}_{\text{enc}}$ is aggregated to obtain single-scale feature map for each actor. The weight utilized for the aggregation is acquired by applying a linear layer to AE, adding actor-specificity to key and value. We denote the aggregated output as *actor-specific context feature* $\mathbf{x}$.

Classifying decoder layer (CDL). CDL is designed to enable the model to selectively attend to class-specific information. As mentioned in Sec. 3, the process for constructing classification features in the contemporary approaches [2, 43] introduces bias towards the actor's body parts. By introducing class queries, our decoder determines on which context the model should focus for each action class, thus allowing for an expanded scope of observation that may extend beyond the bounding box of the actor. However, adopting class queries in a naïve manner leads to the problem of *actor-agnostic activation*; class queries may activate the class-specific context belonging to wrong actors, thereby gathering clues that are not relevant to the target actor the model aims to classify (Fig. 6(a)). Therefore, the objective of CDL is to enhance specificity of both actor and class simultaneously. As class queries improve the class-specificity, we additionally incorporate actor positional queries P and actor-specific context features $\mathbf{x}$ to address the actor-specificity.

We present the structure of CDL in Fig. 4(b). As CDL creates classification features for each actor in a parallel manner, we index features that correspond to each actor with $i \in [1, N_a]$ and describe how the module operates for the i-th actor. CDL is a cross attention layer whose input query is comprised with class queries $\boldsymbol{q} \in \mathbb{R}^{N_c \times D}$ and the actor positional query $P_i \in \mathbb{R}^D$. Since the input query is a combination of both class-specific and actor-specific features, the output feature of CDL grants information dedicated to each class and the actor concurrently. To further enhance the actor-specificity of the output, we utilize the actor-specific context feature $\mathbf{x}_i$ as the input key and value of CDL. We first broadcast the actor feature $\mathbf{f}_i \in \mathbb{R}^D$ to match the spatial dimension of $\mathbf{x}_i \in \mathbb{R}^{H \times W \times D}$, and take sum of these two features to pass it through subsequent convolutional layers. Hence, the resulting feature map, which we denote as $\mathbf{z}_i \in \mathbb{R}^{HW \times D}$, represents the interaction that happens between the i-th actor and the context. Finally, $\mathbf{z}_i$ is paired with positional embeddings $P_{\mathbf{V}} \in \mathbb{R}^{HW \times D}$ to form the input key's spatial part, and enter the following cross-attention layer.

In the cross-attention layer, the i-th actor's classification attention map $\mathcal{A}_i \in \mathbb{R}^{N_c \times HW}$ is constructed as follows:

$$\mathcal{A}_i \propto \text{softmax}\left(\left(\left[\begin{array}{c|c} \boldsymbol{q} & \begin{array}{c} — P_i^T — \\ \vdots \\ — P_i^T — \end{array} \end{array}\right] W_Q\right)\left(\left[\mathbf{z}_i | P_{\mathbf{V}}\right] W_K\right)^T\right), \tag{1}$$

where W_Q and W_K are query and key projection matrices of the cross attention, respectively. Since W_Q and W_K directly transform the features specific to both actor and class, the attention weight's contributions to each class logit become significantly more diverse than prior methods [2, 43] where such contributions of attention weights only differ by a scale within different classes.[1] To be specific,

$$(\mathcal{A}_i)_{(c,m)} \propto \text{softmax}\left(\left(\nu_{(c,i)}^T W_Q\right)\left(\eta_{(i,m)}^T W_K\right)^T\right), \tag{2}$$

where $m \in [1, HW]$ indexes a spatial region, $\nu_{(c,i)}$ is a vector that is conditioned to class and actor, and $\eta_{(i,m)}$ is a vector that is conditioned to actor and region. As the class index c and the actor index i changes in Eq. (2), the following variation of the attention map can be trained with the transformer weights dynamically, and thus, we achieve the goal of obtaining both class-specificity and actor-specificity.

The output of CDL $\tilde{\boldsymbol{q}} \in \mathbb{R}^{N_c \times D}$ is subsequently passed to the next layer to serve as class queries again. To derive the probability for each class, $\tilde{\boldsymbol{q}}$ from the last layer is mean pooled across the channel dimension and then processed through a sigmoid layer. Thus, the final output $\tilde{\boldsymbol{q}} \in [0, 1]^{N_c}$ becomes the classification score for the i-th actor.

4.3 Training objective

The outputs of the decoder, $\tilde{\boldsymbol{q}}$, A_i, and $\mathbf{f}_i$ all contribute to the calculation of the loss. The confidence score $\hat{\boldsymbol{p}} \in [0, 1]^{N_a \times 1}$ for each actor is derived from $\mathbf{f}_i$ to determine the validity of the i-th actor's classification and box regression results. The final box and class outputs $\hat{Y} = \left\{\left(\hat{\mathbf{B}}_i, \hat{\mathbf{C}}_i\right) | \hat{\mathbf{B}}_i \in \mathbb{R}^{T\times 4}, \hat{\mathbf{C}}_i \in [0, 1]^{T \times N_c},\ i \in [1, N_a]\right\}$ are first processed with Hungarian algorithm [15] to ensure the model to output a single optimal detection result per actor. We describe the details of the matching procedure in Sec. F of the supplementary materials. The prediction results that are matched with ground truth boxes and classes receive loss signals to output correct answers, while the remaining predictions are trained to output zero probabilities. Let Y be a padded ground truth labels of size N_a, i.e., $Y = \left\{\left(\mathbf{B}_i, \mathbf{C}_i\right) | i \in [1, N_a]\right\}$ where $\mathbf{B}_i = \mathbf{0}_{T\times 4}$ and $\mathbf{C}_i = \mathbf{0}_{T\times N_c}$ for $i \in [N_X + 1, N_a]$. Additionally assume that the Hungarian matcher assigns the i-th ground truth label of Y to the index $\omega(i)$ of $\hat{Y}$, then each prediction $\hat{Y}_{\omega(i)}$ is passed to the following loss function:

$$\begin{aligned} \mathcal{L}(Y_i, \hat{Y}_{\hat{\omega}(i)}) = {} & \lambda_{\text{class}} \mathcal{L}_{\text{class}}(\mathbf{C}_i, \hat{\mathbf{C}}_{\hat{\omega}(i)}) \\ & + \mathbb{1}_{i \le N_X} \lambda_{\text{box}} \mathcal{L}_{\text{box}}(\mathbf{B}_i, \hat{\mathbf{B}}_{\hat{\omega}(i)}) + \mathbb{1}_{i \le N_X} \lambda_{\text{giou}} \mathcal{L}_{\text{giou}}(\mathbf{B}_i, \hat{\mathbf{B}}_{\hat{\omega}(i)}) \\ & + \mathbb{1}_{i \le N_X} \lambda_{\text{conf}} \mathcal{L}_{\text{conf}}(\mathbf{1}, \hat{\boldsymbol{p}}_{\hat{\omega}(i)}) + \mathbb{1}_{i > N_X} \lambda_{\text{conf}} \mathcal{L}_{\text{conf}}(\mathbf{0}, \hat{\boldsymbol{p}}_{\omega(i)}), \end{aligned} \tag{3}$$

[1] See Sec. A. of the supplementary material for details.

where

$$\begin{aligned} &\mathcal{L}_{\text{class}}(\mathbf{C}_i, \hat{\mathbf{C}}_{\hat{\omega}(i)}) = \text{BFLoss}(\mathbf{C}_i, \hat{\mathbf{C}}_{\hat{\omega}(i)}),\ \mathcal{L}_{\text{box}}(\mathbf{B}_i, \hat{\mathbf{B}}_{\hat{\omega}(i)}) = \|\mathbf{B}_i - \hat{\mathbf{B}}_{\hat{\omega}(i)}\|_1, \\ &\mathcal{L}_{\text{giou}}(\mathbf{B}_i, \hat{\mathbf{B}}_{\hat{\omega}(i)}) = -\text{GIoU}(\mathbf{B}_i, \hat{\mathbf{B}}_{\hat{\omega}(i)}\big),\ \mathcal{L}_{\text{conf}}(\cdot, \hat{\boldsymbol{p}}_{\hat{\omega}(i)}) = \text{BCELoss}(\cdot, \hat{\boldsymbol{p}}_{\omega(i)}). \end{aligned} \tag{4}$$

Notice that BFLoss, BCELoss, and GIoU are a binary focal loss [19], binary cross-entropy loss, and generalized IoU [25], respectively, and the corresponding lambdas are hyperparameters to balance the loss term. Since $\mathcal{L}(Y_i, \hat{Y}_{\omega(i)}) \in \mathbb{R}^T$, the final loss $\mathcal{L}$ is represented as:

$$\mathcal{L} = \frac{1}{|T||N_a|} \sum_{t=1}^{T} \sum_{i=1}^{N_a} \mathcal{L}(Y_i, \hat{Y}_{\hat{\omega}(i)})_t. \tag{5}$$

5 Experiments

5.1 Experimental setup

Datasets. The model's performance is evaluated on three conventional public benchmarks: AVA, JHMDB51-21, and UCF101-24. AVA [8] is a large scale dataset of 430, 15-minute film/TV show clips. To be specific, it consists of 211K frames for training and 57K frames for validating. Due to the large scale of the AVA dataset, it is sparsely annotated at a rate of 1 frame per second (FPS). Following a standard evaluation protocol, we evaluate the model only at the frame level. Furthermore, we report the performance based on the refined annotation AVA v2.2. JHMDB51-21 [11] provides 928 short video clips from YouTube and movie clips. All videos of JHMDB51-21 are fully annotated with one of 21 actions, and actor's bounding box. Additionally, videos are trimmed to only contain frames where actions occur, removing the need for temporal localization of the task. UCF101-24 [28] provides 3,207 untrimmed YouTube videos, which means it contains frames where no action is taking place. Therefore, the dataset requires model to be able to distinguish between frames with and without actions. Following the customary convention, we utilize the corrected annotations [27].
Evaluation criteria. The model is evaluated with mean Average Precision (mAP) under IoU threshold of 0.5. Following the convention, mAP is applied on either a frame-level (f-mAP) or a video-level (v-mAP).

5.2 Ablation studies

We carry out ablation studies to justify our choice of design. A CSN-152 [33] backbone pretrained on Kinetics-400 [13] and Instagram65M [7] is used for the experiments on AVA, and a ViT-B [4] backbone pretrained on Kinetics-400 is used for the experiments on UCF101-24.
Effectiveness of model components. We demonstrate the efficacy of each proposed component of our model. The baseline configuration of the vanilla model is set to include an ordinary transformer encoder [34] and a DETR decoder [1],

Table 1: Ablation experiments on each module of the model.

Encoder	Decoder	mAP
Transformer	DETR	28.6
Transformer	LDL	29.1
Transformer	LDL + CDL	31.4
3D Deformable Transformer	LDL	31.3
3D Deformable Transformer	LDL + CDL	33.5

Table 2: Ablation experiments on ways to aggregate multi-scale feature maps.

Method	AVA	UCF
actor-specific	33.5	85.9
weighted sum	32.9	82.0
mean pooling	32.0	82.8

Table 3: Effect of attaching the actor positional queries to class queries.

Method	AVA	UCF
w/ actor positional queries	33.5	85.9
w/o actor positional queries	31.7	82.9

Table 4: Ablation experiments on ways to combine actor and context features.

Method	AVA	UCF
summation	33.5	85.9
concat + 1d conv [30]	31.8	84.7
cross-attention [43]	31.3	81.3
self-attention [2]	30.8	81.0

Fig. 5: Detection results from the model that uses a single-scale feature map (left) and multi-scale feature maps (right). Detection threshold is set to 0.5.

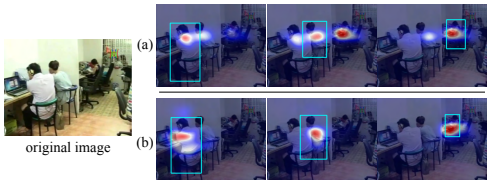


Fig. 6: Attention visualizations of the class, *'sit'*. (a) shows the model without the attachment of the actor positional queries to class queries, and (b) depicts the model with this attachment.

which are commonly observed baseline structures in recent transformer-based VAD models [2,43]. The impact of each module is assessed through a comparative analysis of the rows in Table 1 where the respective modules have been ablated. As the CDL module is dependent to LDL, we add CDL on top of LDL to measure its effectiveness. The most prominent improvement comes from the use of 3D Deformable Transformer Encoder and CDL. Our encoder module takes multi-scale feature maps and enables the model to capture fine details. Fig. 5 illustrates the detection results of the third and fifth row of Table 1, demonstrating the effectiveness of utilizing multi-scale feature maps. Still, without the aid of CDL, the advantage of utilizing multi-scale feature maps is not fully maximized: the fourth row shows significantly lower performance than the current model. In fact, the current state of the art STMixer [42] also utilizes multi-scale feature maps, but it underperforms compared to our model (Table 5).

Different ways to aggregate multi-scale feature maps. In LDL, we aggregate multi-scale feature maps into single-scale feature maps, while conditioning them to each actor to provide actor-specific context to CDL. Actor-conditioned aggregation helps identifying context features that are relevant to the actor of interest and mitigates the issue of mistakenly gathering context that is relevant

Table 5: Performance comparison on three benchmarks [8, 11, 28]. The column labeled "D" signifies whether a model utilizes an off-the-shelf detector or not.

Model	D	Input	Backbone	Pre-train	mAP
SlowFast [6]	✓	32 × 2	SF-R101-NL	K600	29.0
ACAR [22]	✓	32 × 2	SF-R101-NL	K600	31.4
AIA [31]	✓	32 × 2	SF-R101	K700	32.3
VideoMAE [32]	✓	16 × 4	ViT-B	K400	31.8
MeMViT [40]	✓	32 × 3	MViTv2-B	K600	32.8
WOO [3]	✗	32 × 2	SF-R101-NL	K600	28.3
TubeR [43]	✗	32 × 2	CSN-152	IG65M, K400	31.1
STMixer [42]	✗	32 × 2	CSN-152	IG65M, K400	<u>32.8</u>
STMixer [42]	✗	16 × 4	ViT-B	K400	<u>32.6</u>
STMixer [42]	✗	16 × 4	ViT-B [35]	K710, K400	36.1
EVAD [2]	✗	16 × 4	ViT-B	K400	32.3
EVAD [2]	✗	16 × 4	ViT-B [35]	K710, K400	<u>37.7</u>
Ours	✗	32 × 2	CSN-152	IG65M, K400	**33.5**
Ours	✗	16 × 4	ViT-B	K400	**32.9**
Ours	✗	16 × 4	ViT-B [35]	K710, K400	**38.4**

(a) Performance comparison on AVA v2.2.

Model	D	Input		Backbone	f-mAP / v-mAP	
		JHMDB	UCF		JHMDB	UCF
MOC [17]	✓	7 × 1		DLA34	70.8 / 77.2	78.0 / 53.8
AVA [8]	✗	20 × 1		I3D-VGG	73.3 / 78.6	76.3 / <u>59.9</u>
ACRN [30]	✗	20 × 1	-	SF-R101	77.9 / 80.1	- / -
CA-RCNN [41]	✓	32 × 2	-	R50-NL	79.2 / -	- / -
YOWO [14]	✗	16 × 1		3DResNext-101	74.4 / <u>85.7</u>	80.4 / 48.8
WOO [3]	✗	32 × 2	-	SF-R101-NL	80.5 / -	78.8 / -
AIA [31]	✓	-	32 × 1	R50-C2D	- / -	78.8 / -
ACAR [22]	✓	-	32 × 1	SF-R50	- / -	84.3 / -
TubeR [43]	✗	32 × 2		CSN-152	- / 82.3	83.2 / 58.4
STMixer [42]	✗	32 × 2		SF-R101-NL	86.7 / -	83.7 / -
EVAD [2]	✗	16 × 4		ViT-B	**90.2** / 77.8	<u>85.1</u> / 58.8
Ours	✗	40 × 1	32 × 1	ViT-B	<u>86.9</u> / **88.5**	**85.9** / **61.7**

(b) Performance comparison on JHMDB and UCF.

to same actions of different actors. Table 2 shows its effectiveness: providing actor-specific context demonstrates superior results on both datasets.

Effectiveness of using the actor positional queries for classification. As explained in Sec. 4.2, we attach the actor positional queries to class queries in order to provide class queries with information regarding the actor the model aims to classify. Similarly to actor-conditioned aggregation from the previous section, the attachment of the actor positional queries improves actor-specificity of the class queries and prevents the activation of the context that belongs to wrong actors. Fig. 6(a) shows the case where the actor positional queries are not attached to class queries: class queries activate regions where a similar actor performs the same action. In contrast, Fig. 6(b) illustrates the impact of the actor positional queries, mitigating the issue of actor-agnostic activation. The results from Table 3 also support our claim: detaching the actor positional queries compromises the performance by a significant margin.

Different ways to combine actor and context features. Merging the actor and context feature is an essential step in VAD to model the interactions between the actor and the background. We explore various methods to merge the actor and context features to justify our chosen approach. Each method we chose in Table 4 corresponds to a way used in prior approaches [2, 30, 43]. Interestingly, we discover that simple summation yields superior performance overall.

5.3 Comparison to the state-of-the-art methods

Table 5(a) summarizes the experiment results on AVA v2.2. We compare our model with recent video backbones widely used in video action detection, ViT-B [4, 35] and CSN-152 [33]. Among the models that utilize ViT-B as their backbones, our model surpasses other models [2, 42] that have identical settings to ours. Out of the models employing CSN-152 as well our model exhibits a superior performance, highlighting its competitive edge.

Table 5(b) shows the results on JHMDB51-21 and UCF101-24. Our model demonstrates comparable or superior performance to current state-of-the-art models on both datasets. We conjecture the lower performance compared to

Table 6: Efficiency comparison on JHMDB and UCF. Each frame's resolution is set to 256 × 360.

Method	Backbone	Input ($T \times \tau$)	Params	FLOPs	Inf. time (ms)	f-mAP/v-mAP
JHMDB: inferring a 40-frame tube						
TubeR	CSN-152	32×2	**91.8M**	6.99T	1263.1	- / 82.3
STMixer	SF-R101-NL	32×2	219.2M	7.64T	2088.2	86.7 / -
EVAD	ViT-B	16×4	185.4M	10.68T	8363.1	**90.2** / 77.8
Ours	ViT-B	40×1	117.8M	**3.26T**	**432.0**	86.9 / **88.5**
UCF: inferring a 32-frame tube						
TubeR	CSN-152	32×2	**91.8M**	5.60T	1161.9	83.2 / 58.4
STMixer	SF-R101-NL	32×2	219.2M	6.12T	1671.5	83.7 / -
EVAD	ViT-B	16×4	185.4M	8.54T	6797.8	85.1 / 58.8
Ours	ViT-B	32×1	117.8M	**3.73T**	**370.0**	**85.9** / **61.7**

Table 7: Performance comparison on the latest VAD models.

Method	Backbone	f-mAP	when GT box is provided
TubeR	CSN-152	31.1	$33.1_{(+2.0)}$
STMixer	ViT-B	32.6	$34.7_{(+2.1)}$
STMixer	ViT-B [35]	36.1	$38.7_{(+2.6)}$
EVAD	ViT-B	32.3	$34.5_{(+2.2)}$
EVAD	ViT-B [35]	37.7	$40.0_{(+2.3)}$
Ours	CSN-152	**33.5**	$\mathbf{37.2}_{(+\mathbf{3.7})}$
Ours	ViT-B	**32.9**	$\mathbf{36.9}_{(+\mathbf{4.0})}$
Ours	ViT-B [35]	**38.4**	$\mathbf{42.1}_{(+\mathbf{3.7})}$

the prior method [2] is due to the low class diversity of JHMDB51-21, since the strength of our model comes from its ability to create diverse classification feature conditioned to each action class.

Moreover, the tube-creating property of our model largely improves the model's efficiency. Table 6 shows the amount of computation utilized by current models to generate a single tube. Note that the input size $T \times \tau$ signifies the number of frames T and sampling rate τ. The current best-performing models [2, 42] produce predictions only for a single frame within a single feedforward, while ours produce a whole tube at once. Thus, although the input size differs, taking the input with larger T with sampling rate 1 is to infer a tube with a single feedforward, thereby compensating for the multi-feedforwarding property of other models that process smaller input sizes, thus ensuring fairness in comparison. Specificially, for JHMDB51-21, with its longest clip comprising 40 frames, we set $T = 40$, while for UCF101-24, where the longest clip spans 900-frame long, thus we adopt $T = 32$, following the standard practice. Accordingly, we measure the amount of computation and speed needed for generating a 40-frame tube for JHMDB51-21 and a 32-frame tube for UCF101-24. Notably, our model achieves competitive results with fewer parameters and the least FLOPs. Due to the page limit, we provide a detailed explanation of the model's efficiency in Sec.G of the supplementary material.

5.4 Attention visualization and analysis

One of key advantages of our model is its ability to provide interpretable attention maps for individual class labels. Fig. 7 shows the classification attention maps the model provides across various scenarios. The model dynamically focuses on crucial regions; for example, when classifying actions like *'hand shake'* or *'write,'* it tends to examine the areas near the actor's hands and arms. If necessary, the model also directs its attention towards regions outside the actor's body such as *'listen to (a person)'* in the first and second row, or *'watch (e.g., TV)'* in the fourth row. Moreover, it differentiates class-relevant context of actors well in multi-actor scenarios, even when multiple actors are performing identical actions.

Is performance improvement really from classification? Since average precision (AP) simultaneously measures a model's localization and classification

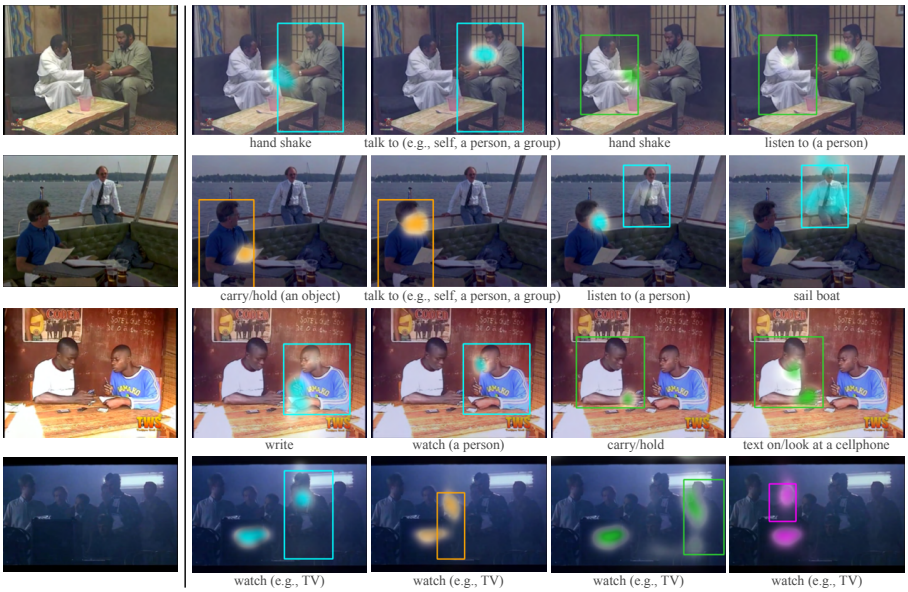


Fig. 7: Classification attention map visualization results on AVA v2.2. Each map represents the region where the model has observed to classify the action of the actor, marked with the same color as the bounding box.

ability, we conduct experiments to measure the model's classification ability explicitly (Table 7). The model outputs are Hungarian-matched with the ground-truth (GT) instances using the same matching cost, and their predicted box coordinates are replaced with that of GT annotations. The experiment results show that the performance gap between ours and other methods has increased, implying that our model's performance improvement is mainly due to its enhanced classification ability.

6 Conclusion and limitations

We have introduced a model designed for effective classification, constructing features dedicated to each class and each actor. Our approach adeptly manages classification features by considering context not bounded by the actor's location, while still capturing information specific to the targeted actor. We achieve the state-of-the-art performance on the most challenging benchmark, while being the most efficient network when inferring a tube.

Limitations. However, there still exists room for further improvement. The current decoder architecture does not exchange information across frames due to the restricted size of memory. As a result, a role of capturing temporal dynamics is solely dependent on the encoder architecture, placing a significant burden on the encoder. Thus, further studies might aim to achieve greater performance by sparsely collecting class information from the spatial domain, leaving residual memory available for capturing temporal dynamics.

Acknowledgements

This work was partly supported by the NAVER Cloud Corporation and the NRF grant and IITP grant funded by Ministry of Science and ICT, Korea (NRF-2021R1A2C3012728, RS-2022-II220926, RS-2022-II220290, RS-2019-II191906).

References

1. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (2020)
2. Chen, L., Tong, Z., Song, Y., Wu, G., Wang, L.: Efficient video action detection with token dropout and context refinement. arXiv preprint arXiv:2304.08451 (2023)
3. Chen, S., Sun, P., Xie, E., Ge, C., Wu, J., Ma, L., Shen, J., Luo, P.: Watch only once: An end-to-end video action detection framework. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8178–8187 (2021)
4. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)
5. Feichtenhofer, C.: X3d: Expanding architectures for efficient video recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 203–213 (2020)
6. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6202–6211 (2019)
7. Ghadiyaram, D., Tran, D., Mahajan, D.: Large-scale weakly-supervised pre-training for video action recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12046–12055 (2019)
8. Gu, C., Sun, C., Ross, D.A., Vondrick, C., Pantofaru, C., Li, Y., Vijayanarasimhan, S., Toderici, G., Ricco, S., Sukthankar, R., et al.: Ava: A video dataset of spatio-temporally localized atomic visual actions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6047–6056 (2018)
9. Guo, C., Fan, B., Zhang, Q., Xiang, S., Pan, C.: Augfpn: Improving multi-scale feature learning for object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12595–12604 (2020)
10. Herzig, R., Ben-Avraham, E., Mangalam, K., Bar, A., Chechik, G., Rohrbach, A., Darrell, T., Globerson, A.: Object-region video transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3148–3159 (2022)
11. Jhuang, H., Gall, J., Zuffi, S., Schmid, C., Black, M.J.: Towards understanding action recognition. In: Proceedings of the IEEE international conference on computer vision. pp. 3192–3199 (2013)
12. Josmy Faure, G., Chen, M.H., Lai, S.H.: Holistic interaction transformer network for action detection. arXiv e-prints pp. arXiv–2210 (2022)
13. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al.: The kinetics human action video dataset. arXiv preprint arXiv:1705.06950 (2017)
14. Köpüklü, O., Wei, X., Rigoll, G.: You only watch once: A unified cnn architecture for real-time spatiotemporal action localization. arXiv preprint arXiv:1911.06644 (2019)

15. Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955)
16. Li, F., Zhang, H., Liu, S., Guo, J., Ni, L.M., Zhang, L.: Dn-detr: Accelerate detr training by introducing query denoising. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13619–13627 (2022)
17. Li, Y., Wang, Z., Wang, L., Wu, G.: Actions as moving points. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16. pp. 68–84. Springer (2020)
18. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
19. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
20. Liu, S., Li, F., Zhang, H., Yang, X., Qi, X., Su, H., Zhu, J., Zhang, L.: Dab-detr: Dynamic anchor boxes are better queries for detr. In: ICLR (2022)
21. Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L., Wang, J.: Conditional detr for fast training convergence. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3651–3660 (2021)
22. Pan, J., Chen, S., Shou, M.Z., Liu, Y., Shao, J., Li, H.: Actor-context-actor relation network for spatio-temporal action localization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 464–474 (2021)
23. Pang, Y., Zhao, X., Zhang, L., Lu, H.: Multi-scale interactive network for salient object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9413–9422 (2020)
24. Peng, X., Schmid, C.: Multi-region two-stream r-cnn for action detection. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. pp. 744–759. Springer (2016)
25. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 658–666 (2019)
26. Saha, S., Singh, G., Sapienza, M., Torr, P.H., Cuzzolin, F.: Deep learning for detecting multiple space-time action tubes in videos. arXiv preprint arXiv:1608.01529 (2016)
27. Singh, G., Saha, S., Sapienza, M., Torr, P.H., Cuzzolin, F.: Online real-time multiple spatiotemporal action localisation and prediction. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3637–3646 (2017)
28. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
29. Sui, L., Zhang, C.L., Gu, L., Han, F.: A simple and efficient pipeline to build an end-to-end spatial-temporal action detector. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 5999–6008 (2023)
30. Sun, C., Shrivastava, A., Vondrick, C., Murphy, K., Sukthankar, R., Schmid, C.: Actor-centric relation network. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 318–334 (2018)
31. Tang, J., Xia, J., Mu, X., Pang, B., Lu, C.: Asynchronous interaction aggregation for action detection. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16. pp. 71–87. Springer (2020)

32. Tong, Z., Song, Y., Wang, J., Wang, L.: Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. Advances in neural information processing systems **35**, 10078–10093 (2022)
33. Tran, D., Wang, H., Torresani, L., Feiszli, M.: Video classification with channel-separated convolutional networks. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 5552–5561 (2019)
34. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
35. 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: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14549–14560 (2023)
36. Wang, T., Borji, A., Zhang, L., Zhang, P., Lu, H.: A stagewise refinement model for detecting salient objects in images. In: Proceedings of the IEEE international conference on computer vision. pp. 4019–4028 (2017)
37. Wang, Y., Zhang, X., Yang, T., Sun, J.: Anchor detr: Query design for transformer-based detector. In: Proceedings of the AAAI conference on artificial intelligence. vol. 36, pp. 2567–2575 (2022)
38. Weinzaepfel, P., Harchaoui, Z., Schmid, C.: Learning to track for spatio-temporal action localization. In: Proceedings of the IEEE international conference on computer vision. pp. 3164–3172 (2015)
39. Wu, C.Y., Feichtenhofer, C., Fan, H., He, K., Krahenbuhl, P., Girshick, R.: Long-term feature banks for detailed video understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 284–293 (2019)
40. Wu, C.Y., Li, Y., Mangalam, K., Fan, H., Xiong, B., Malik, J., Feichtenhofer, C.: Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13587–13597 (2022)
41. Wu, J., Kuang, Z., Wang, L., Zhang, W., Wu, G.: Context-aware rcnn: A baseline for action detection in videos. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16. pp. 440–456. Springer (2020)
42. Wu, T., Cao, M., Gao, Z., Wu, G., Wang, L.: Stmixer: A one-stage sparse action detector. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14720–14729 (2023)
43. Zhao, J., Zhang, Y., Li, X., Chen, H., Shuai, B., Xu, M., Liu, C., Kundu, K., Xiong, Y., Modolo, D., et al.: Tuber: Tubelet transformer for video action detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13598–13607 (2022)
44. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. In: ICLR (2021)