Classification Matters: Improving VAD with Class-Specific Attention

—Supplementary Material—

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/

We present omitted experiments and details in this supplementary material as below:

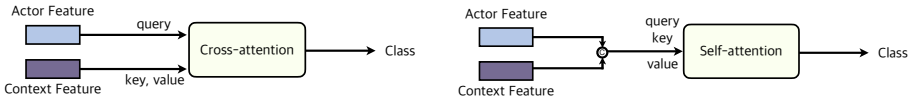
- Sec. A: Analyzing Previous Models' Attention Weight Contributions to Classification Logits
- Sec. B: Implementation for Single-label Datasets
- Sec. C: Detailed Logic of the 3D Deformable Encoder
- Sec. D: Detailed Logic of Localizing Decoder Layer (LDL)
- Sec. E: Detailed Logic of Classifying Decoder Layer (CDL)
- Sec. F: Details of the Hungarian Matching Process
- Sec. G: Comparison with Latest VAD Models
- Sec. H: Experimental Setup
- Sec. I: Additional Ablation Experiments
- Sec. J: Additional Qualitative Results
- Sec. K: Class-wise mAP Comparison

A Analyzing Previous Models' Attention Weight Contributions to Classification Logits

To further investigate the difference between prior transformer-based methods [2, 16] and our model, we analyze how attention weights of the previous models affect the final classification logits. In summary, we find that prior methods allow their each class logit to have comparably subtle differences within different classes with respect to the acquired attention weights, and thus, enforce their transformer outputs to include more commonly shared semantics across distinct classes.

In Fig. A1, we present how previous methods construct their classification features. The notations defined in Sec. 4 of the main paper are adaptively reused for a clearer comparison to our method: we denote the actor feature as $\mathbf{f} \in \mathbb{R}^{N_a \times D}$ and the context feature as $\mathbf{V} \in \mathbb{R}^{THW \times D}$. TubeR [16] constructs its classification feature with a cross-attention layer whose input query is $\mathbf{f}$ and input key/value

*Work done while doing an internship at NAVER Cloud.



(a) Classification module of TubeR [16] **(b)** Classification module of EVAD [2]

Fig. A1: Two previous approaches for action classification using transformer.

are $\mathbf{V}$. Then, the i-th actor's classification attention map $\mathcal{A}$ is constructed as follows:

$$\mathcal{A}_i \propto \text{softmax}\left(\left(\mathbf{f}_i^T W_Q\right)\left(\mathbf{V}W_K\right)^T\right), \tag{A1}$$

where W_Q and W_K are D by D projection matrices of the transformer. Since the same attention map is shared across to infer different classes, the final linear layer, which we will denote its parameter as $W^{\text{cls}} \in \mathbb{R}^{N_c \times D}$, needs to be involved to further analyze the impact of attention weights to the classification logits. For brevity, we disregard the bias term in this discussion. Let c-th class logit of the i-th actor be $\ell_{(i,c)}$, then it is derived from Eq. A1 as:

$$\begin{aligned}\ell_{(i,c)} &\propto W_c^{\text{cls}}\left(\sum_m \left(\mathcal{A}_{(i,m)} \odot \mathbf{V}_m W_V\right)\right) \\ &= \sum_d \left(W_{(c,d)}^{\text{cls}} \sum_m \nu_{(i,m,d)}\right),\end{aligned} \tag{A2}$$

where $m \in [1, THW], d \in [1, D], W_V$ is a projection matrix, and $\nu_{(i,m,d)}$ is a scalar that is conditioned to actor, region and channel dimension. As the class index c varies in Eq. A2, the only relevant factor that influences the impact of the attention weights on classification logits is $W_{(c,d)}^{\text{cls}}$, which is merely a scalar value that varies across different classes. Although slightly different, such impact is similarly derived in the case of EVAD [2]. Therefore, the weight responsible for distinguishing distinct classes is solely dependent on the elements of W^{cls}. Given the limited variation that can be generated within classes at the final layer, the transformer weights are trained to alleviate the burden of W^{cls} by capturing the commonly shared semantics (*i.e.*, actors) across different classes, so that each action class can be expressed with a unique combination of such semantics.

On the other hand, our model involves class-specificity during the generation of attention weights, and thus, the distinction between action classes no longer depends on the final linear layer. This effectively enhances the expressiveness of the classification feature, eventually aiding in solving VAD, which particularly requires the identification of subtle differences between individual action classes.

B Implementation for single-label datasets

The description of Sec. 4 of the main paper assumes that the given dataset has a multi-label property, since it covers more general scenarios that include the cases of single-label datasets. Still, we slightly change the model's implementation

to inject this single-label prior. To be specific, since the ground-truth label $\{\mathbf{C}_i \in \{0,1\}^{T\times N_c} |\ i \in [1, N_X]\}$ is a one-hot vector for each timestep, the model's classification output $\{\hat{\mathbf{C}}_i \in [0,1]^{T\times N_c} |\ i \in [1, N_X]\}$ is under the constraint of $\sum_{c=1}^{N_c} \hat{\mathbf{C}}_i[t,c] = 1\ \forall i \in [1, N_X], \forall t \in [1, T]$. Thus, the output $\tilde{\boldsymbol{q}} \in \mathbb{R}^{N_c\times 1}$ (after mean pooling) undergoes through a softmax layer instead of the sigmoid layer.

Furthermore, classification outputs that are not matched with GT labels, i.e., $\hat{\mathbf{C}}_{\omega(i)} \forall i > N_X$, are trained to output zero probabilities for all classes. However, it is not possible if the output is processed through the softmax layer. Thus, we additionally involve the confidence score $\hat{\boldsymbol{p}}_i \in [0,1]$ of each actor candidate to refine the classification logits, so that the unmatched outputs can return zero vector if necessary.

C Detailed logic of 3D Deformable Transformer Encoder

Due to the heavy nature of multi-scale spatio-temporal feature maps, we adopt ideas from Deformable DETR [17] for an efficient encoding process. We extend the idea to incorporate temporal dynamics, since the original approach only allows the information exchange within a single frame. Note that every $\boldsymbol{v} \in \{\boldsymbol{v}^l(t,h,w) | (t,h,w) \in [1,T_l] \times [1,H_l] \times [1,W_l]\}_{l=1}^{L} \subset \mathbb{R}^D$ is regarded as the query and goes through the identical encoding process, where $\boldsymbol{v}^l(t,h,w)$ indicates the 1D feature located in the (t,h,w)-th position in $\boldsymbol{v}^l$. Let q index a query element of the encoder where its normalized coordinates and its corresponding query feature are denoted by $\hat{\boldsymbol{p}}_q \in [0,1]^3$ and $\boldsymbol{v}_q \in \mathbb{R}^D$, respectively. Given $\mathbf{V} = \{\boldsymbol{v}^l\}_{l=1}^{L}$, the 3D multi-scale deformable encoder module is applied for each $\boldsymbol{v}_q$ as

$$\begin{aligned}&\text{3DMSDeformableEncoder}\Big(\boldsymbol{v}_q, \hat{\boldsymbol{p}}_q, \{\boldsymbol{v}^l\}_{l=1}^{L}\Big) \\ &\qquad = \sum_{m=1}^{M} \boldsymbol{W}_m \Big[\sum_{l=1}^{L}\sum_{k=1}^{K} A_{mlqk} \cdot \boldsymbol{W}'_m \boldsymbol{v}^l\Big(\phi_l(\hat{\boldsymbol{p}}_q) + \Delta\boldsymbol{p}_{mlqk}\Big)\Big],\end{aligned} \tag{A3}$$

where m indexes the attention head, l indexes the input feature level, and k indexes the sampling point. Similar to the previous work, $\boldsymbol{W}'_m \in \mathbb{R}^{(D/M)\times D}$ and $\boldsymbol{W}_m \in \mathbb{R}^{(D/M)\times D}$ are learnable weights that weigh across multiple attention heads. Also, the offset $\Delta\boldsymbol{p}_{mlqk} \in \mathbb{R}^3$ and the weight $A_{mlqk} \in \mathbb{R}$, which is dedicated to each k-th sampling point at the l-th level of the feature map for the q-th query's m-th attention head, are obtained by applying linear projection layers to $\boldsymbol{z}_q$. Note that $\phi_l(\cdot)$ re-scales the coordinates to the l-th level input feature map, and $\sum_{k=1}^{K} A_{mlqk} = 1$.

D Detailed logic of Localizing Decoder Layer (LDL)

In this section, we provide details of Localizing Decoder Layer (LDL), of which we omit the details in the main paper. Fig. A2 illustrates the transformation process of LDL. Let the i-th actor box and zero-initialized actor embedding

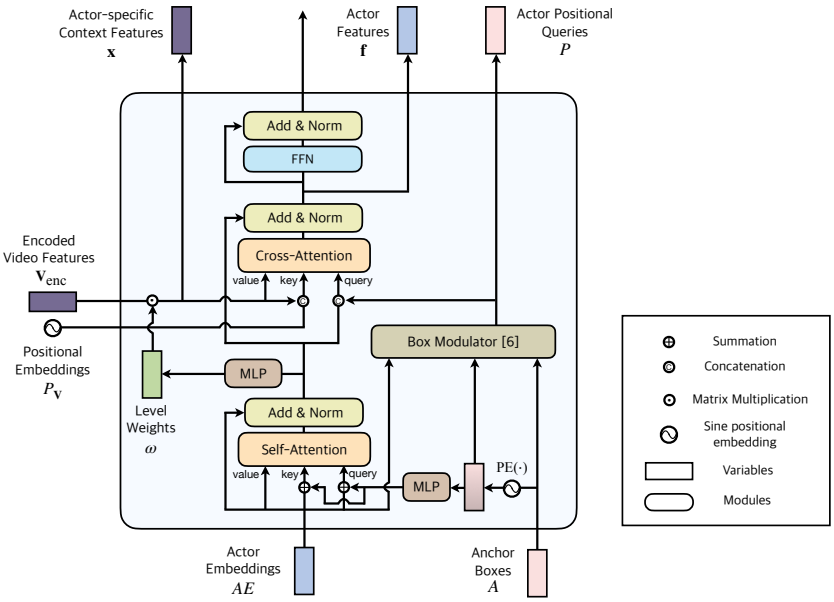


Fig. A2: Structure of Localizing Decoder Layer (LDL)

be $A_i = (x_i, y_i, w_i, h_i)$ and AE_i, respectively, where $i \in [1, N_a]$ and N_a is the number of actor candidates that are each assigned to capture a potential actor that appears in the video. $AE_i \in \mathbb{R}^{T \times D}$ is processed in a temporally parallel manner to output prediction per frame. Hence, for brevity, we omit the time index t and focus on how an individual frame is processed within the decoder layer.

AE initially passes through a self-attention layer to ensure that each i-th actor embedding represents different actors. The key and value are first embedded with positional information provided by the actor box A, and then the AE is updated with self-attention. Before concatenating the actor boxes to the updated AE, the boxes undergo modulation [6] to incorporate more precise position and size information of actors. This utilizes AE (before the self-attention), which is trained to contain the actor's information. To be specific, the box modulation BM is applied to each i-th element of A as follows:

$$\begin{aligned} P_i &= \mathrm{BM}\Big(\Big[\mathrm{PE}(x_i), \mathrm{PE}(y_i)\Big]; \mathrm{AE}_i, A_i\Big) \\ &= \Big[\mathrm{PE}(x_i)\frac{w_{i,\mathrm{ref}}}{w_i}, \mathrm{PE}(y_i)\frac{h_{i,\mathrm{ref}}}{h_i}\Big], \end{aligned} \tag{A4}$$

where $(w_{i,\mathrm{ref}}, h_{i,\mathrm{ref}})$ is obtained by applying a fully connected layer to AE_i, and PE: $\mathbb{R} \to \mathbb{R}^{D/2}$ is a sinusoidal positional encoding.

Note that in the cross-attention layer, the role of the modulated actor box P is to guide the actor embedding AE to gather information relevant to the

i-th instance from the encoded feature maps $\mathbf{V}_{\text{enc}}$. Accordingly, we compose actor-specific context features $\mathbf{x} \in \mathbb{R}^{N_a \times T \times H \times W \times D}$, where $\mathbf{x}_i$ is a feature map dedicated to the i-th actor. Specifically, we apply the weighted summation on $\mathbf{V}_{\text{enc}}$ while conditioning the weights to AE_i:

$$\mathbf{x}_i = \sum_{l=1}^{L} \omega_i^l \mathbf{v}_{\text{enc}}^l \in \mathbb{R}^{T \times H \times W \times D}, \tag{A5}$$

where $(\omega_i^1, \omega_i^2, \ldots, \omega_i^L) = \text{MLP}(AE_i)$. Constructing distinct values that are specific to each actor enhances actor-specificity, helping the model to generate different features for each actor. Afterwards, the cross-attention integrates the features composed so far as follows:

$$AE_i \leftarrow f_{\text{CA}}^{\text{loc}}\Big(q = \Big[AE_i, P_i\Big],\, k = \Big[\mathbf{x}_i, P_{\mathbf{V}}\Big],\, v = \mathbf{x}_i\Big), \tag{A6}$$

where P_i is from Eq. A4 and $P_{\mathbf{V}} \in \mathbb{R}^{T \times H \times W \times D}$ is a 3D positional embedding of the video feature maps $\mathbf{V}$.

Before the output AE undergoes a feedforward network, it is passed to the CDL, and we denote this vector $\mathbf{f}$ as an actor feature. Afterwards, AE contributes to refine the anchor box A as follows:

$$A \leftarrow \sigma(\text{FFN}(AE) + \sigma^{-1}(A)), \tag{A7}$$

where $\text{FFN}(\cdot)$, $\sigma : \mathbb{R} \rightarrow [0, 1]$ and $\sigma^{-1} : [0, 1] \rightarrow \mathbb{R}$ are a feedforward network, sigmoid function, and its inverse that normalizes and unnormalizes the input, respectively.

E Details of the Classifying Decoder Layer (CDL)

We provide details of Classifying Decoder Layer (CDL), describing its miscellaneous operations that are omitted in the main paper. Fig. A3 describes the process of CDL. For brevity, we explain how CDL operates to infer the i-th actor's action class ($i \in [1, N_a]$). Classifying decoder layer (CDL) takes as inputs the actor feature $\mathbf{f}_i \in \mathbb{R}^D$, the actor positional query $P_i \in \mathbb{R}^D$, and the actor-specific context features $\mathbf{x}_i \in \mathbb{R}^{T \times H \times W \times D}$, that are generated in LDL. It additionally takes as its input the learnable embeddings $\boldsymbol{q} \in \mathbb{R}^{N_c \times D}$, which we denote as *class queries*, to embed class-specific information. To elaborate, CDL aims to construct a classification feature $\tilde{\boldsymbol{q}} \in \mathbb{R}^{N_c \times D}$ for each i-th actor, where $\tilde{\boldsymbol{q}}_c \in \mathbb{R}^D$ implies the feature used to detect c-th action performed by the i-th actor.

First, the actor feature passes through a standard feedforward network to be refined for classification. However, it has been noted that optimization of the classification feature can sometimes interfere with localization [9, 14]. Thus, the gradient flow is halted before this feature enters the classifying decoder layer.

We begin with creating features that incorporate the interaction between the i-th actor and the actor-specific context feature $\mathbf{x}_i$. To this end, we spatially

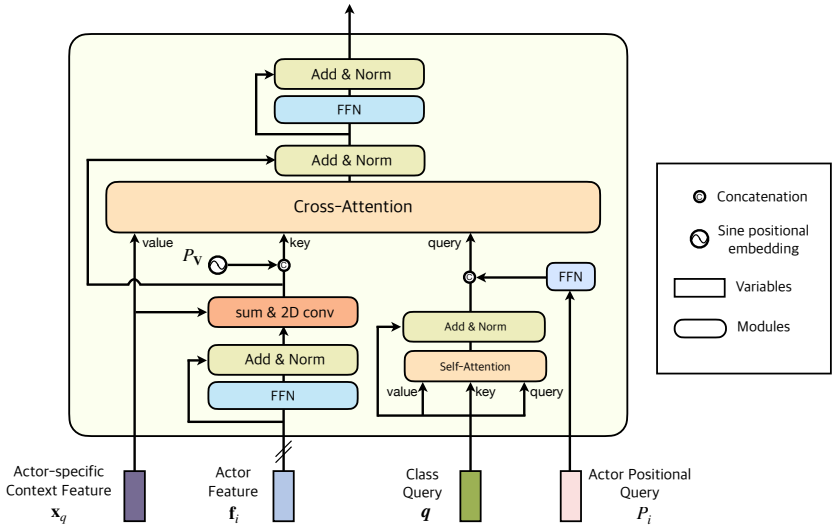


Fig. A3: Structure of Classifying Decoder Layer (CDL)

duplicate the actor feature $\mathbf{f}_i$ to the spatial extent of $\mathbf{x}_i$ and take the sum of these two features to obtain $(\mathbf{f}+\mathbf{x})_i \in \mathbb{R}^{H\times W\times D}$. $(\mathbf{f}+\mathbf{x})_i$ is processed through multiple convolutional layers to embed local relationships between features in $(\mathbf{f}+\mathbf{x})_i$. Let us denote the flattened resulting tensor $\mathbf{z}_i$ as follows:

$$\mathbf{z}_i = \mathrm{Conv2D}\Big((\mathbf{f}+\mathbf{x})_i\Big) \in \mathbb{R}^{HW\times D}. \tag{A8}$$

On the other hand, class queries are trained to encode class-specific information as the layers proceed. Also, the self-attention mechanism between these queries enables them to exchange information among different classes and obtain enriched class-specific features that are aware of the occurrence of other classes. To be specific, the class queries $\boldsymbol{q}$ are passed to the self-attention layer $f^{\mathrm{cls}}_{\mathrm{SA}}(q,k,v)$, serving themselves as the query, key, and value. The output class queries $\boldsymbol{q}$ are then joined with $\mathbf{z}_q$ and $\mathbf{x}_q$ to construct a triplet of inputs for cross-attention layer $f^{\mathrm{cls}}_{\mathrm{CA}}(q,k,v)$, in which the class queries collect information about each class from the features that represent the video context.

In the following cross-attention layer, $\boldsymbol{q}$ and $\mathbf{z}_q$ act as the query and key, respectively. They are trained so that $\boldsymbol{q}_c$ and $\mathbf{z}_i[m]$ $(m \in [1, HW])$ becomes semantically similar when the location m is relevant to the c-th action class. Specifically, the resulting attention map $\mathcal{A}_i \in \mathbb{R}^{HW\times N_c}$ signifies informative regions for each class and is multiplied with the value $\mathbf{x}_i$ to output class-specific features $\tilde{\boldsymbol{q}} \in \mathbb{R}^{N_c\times D}$. However, we have discovered the c-th class query $\boldsymbol{q}_c$ often activates region where the c-th actions happen, but not the ones necessarily performed by the i-th actor. Since the c-th class query learns to activate the region $\mathcal{A}_i[m]$ whenever the region m is relevant to the c-th action, it occasionally confuses to which actor's action it should react. To handle this actor confusion

problem, we concatenated the global positional embedding $P_{\mathbf{V}}$ and the modulated actor positional queries P_i to key and query, respectively. Consequently, the resulting cross-attention $f_{\text{CA}}^{\text{cls}}$ happens as follows:

$$\tilde{\boldsymbol{q}} \leftarrow f_{\text{CA}}^{\text{cls}}\Big(q = \big[\boldsymbol{q}, P_i\big],\, k = \big[\mathbf{z}_i, P_{\mathbf{V}}\big],\, v = \mathbf{x}_i\Big). \tag{A9}$$

As the LDL's output feature of Eq. A6 is subject to localization, the spatial parts $P_{\mathbf{V}}$ and P_i are trained to obtain the positional information of the i-th actor. Hence, concatenating these positional information provides subtle cues about the actor's position to class queries and adds actor-specificity to the output $\tilde{\boldsymbol{q}}$.

The output $\tilde{\boldsymbol{q}} \in \mathbb{R}^{N_c \times D}$ of the cross-attention passes the ordinary feedforward network and 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.

F Details of the Hungarian matching process

For the Hungarian matching between the i-th element $Y_i = \big(\mathbf{B}_i, \mathbf{C}_i\big)$ in the padded ground-truth labels and the j-th element $\hat{Y}_j = \big(\hat{\mathbf{B}}_j, \hat{\mathbf{C}}_j\big)$ in the model predictions, we consider three matching costs, $\mathcal{H}_{i,j}^{\text{box}}$, $\mathcal{H}_{i,j}^{\text{giou}}$, and $\mathcal{H}_{i,j}^{\text{class}}$, which are defined as follows:

$$\begin{aligned} \mathcal{H}_{i,j}^{\text{box}} &= \|\mathbf{B}_i - \hat{\mathbf{B}}_j\|_1, \\ \mathcal{H}_{i,j}^{\text{GIoU}} &= -\text{GIoU}(\mathbf{B}_i, \hat{\mathbf{B}}_j\big), \\ \mathcal{H}_{i,j}^{\text{class}} &= \text{BCELoss}(\mathbf{C}_i, \hat{\mathbf{C}}_j), \end{aligned} \tag{A10}$$

where $\text{GIoU}(\cdot,\cdot)$ is a generalized IoU [8] between boxes and $\text{BCELoss}(\cdot,\cdot)$ is a binary cross entropy loss between two multi-hot vectors. Then, the Hungarian algorithm aims to find the optimal assignment $\hat{\omega} = \text{argmin}_{\omega \in \boldsymbol{\Omega}_{N_a}} \sum_{i=1}^{N_a} \big(\eta_{\text{box}} \mathcal{H}_{i,\omega(i)}^{\text{box}}$ $+ \eta_{\text{GIoU}} \mathcal{H}_{i,\omega(i)}^{\text{GIoU}} + \eta_{\text{class}} \mathcal{H}_{i,\omega(i)}^{\text{class}}\big)$, where $\eta_{\text{box}}, \eta_{\text{box}}$, and η_{box} are coefficients that balance the matching cost for each term. In the case of AVA, we figure that setting $\mathcal{H}_{i,j}^{\text{class}}$ as binary cross entropy loss leads to a slow convergence, which is potentially due to its multi-label property. Thus, we set $\mathcal{H}_{i,j}^{\text{class}}$ as $-\hat{\boldsymbol{p}}_j$, since the confidence $\hat{\boldsymbol{p}}_j$ is learned to have a higher value if j-th element is matched with a ground-truth label. Note that such matching method inherits the way from TubeR [16].

G Comparison with latest VAD models

We provide a comparison analysis between ours and latest VAD models, TubeR [16], STMixer [13], and EVAD [2].

Ours vs TubeR. TubeR is similar to ours in that it is established on DETR [1] architecture along with tube-shaped queries. Furthermore, it outputs multiple

Table A1: Comparison with other models [2, 16] in terms of module components. ‡ is to indicate that its weight is pretrained on COCO [5]. CSN-152 [11] backbone is used.

Method	Encoder	Decoder	mAP
-	Transformer	DETR [1]	28.6
-	Transformer	LDL	29.1
-	Transformer	LDL + CDL	31.4
TubeR [16]	Transformer‡ + LSTR Decoder [15]	DETR‡ + Transformer	31.1
EVAD [2]	Transformer + KTP [2]	FPN [4] + Transformer	N/A
-	3D Deformable Transformer	LDL	31.3
STMixer [13]	Adaptive Feature Sampling [13]	Adaptive Feature Mixing [13]	32.8
Ours	3D Deformable Transformer	LDL + CDL	33.5

frames with a single feed-forward as well, showing greater efficiency compared to other methods. However, the encoder stage of TubeR consumes substantial memory since it performs self-attention over spatio-temporal features.

Ours vs STMixer. Our model and STMixer are similar in that they both utilize multi-scale feature maps to capture fine details of an actor's action. While ours has the deformable encoding process (Sec. C), STMixer does not explicitly have a distinguishable encoding module. However, it has an Adaptive Feature Sampling module, which resembles the deformable encoding process: it also determines which feature points to sample from the multi-scale feature maps with learnable parameters.

Ours vs EVAD. EVAD, in contrast, does not employ multi-scale feature maps. Instead, EVAD takes advantage of encoding the video features in a dense manner: although it prunes the features and makes the self-attention process less heavy, EVAD still exhaustively computes self-similarities between each feature vector and obtains richer encoded features. In addition, EVAD uses the final localization output to obtain classification features, which differs from ours in which the localization and classification features evolve together in the decoder module.

Comparison in a module-level. We provide a module-level comparison Table A1 for a thorough comparison with ours and previous methods. Details for each model are as follows:

- TubeR [16]
  - uses traditional transformer encoder, but also utilizes Long Short-Term Transformer [15] for temporal aggregation.
  - employs DETR [1] weights pretrained on COCO [5].
- STMixer [13]
  - has no explicit encoding module, but based on the roles of their modules, we categorize Adaptive Feature Sampling and Adaptive Feature Mixing modules as its encoder and decoder, respectively.
  - introduces Adaptive Feature Sampling, which resembles the mechanism of Deformable-DETR [17] in that features are encoded by sampling from the feature map.

Table A2: Comparison with other models [2, 16] in terms of computational complexity.

Method	Encoder	Decoder	N_a	K
TubeR [16]	$\mathcal{O}((HW)^2)$	$\mathcal{O}((HW)^2 + HW \times N_a)$	15	-
EVAD [2]	$\mathcal{O}((\rho \times THW)^2)$	$\mathcal{O}((\rho^3 \times THW + N_a)^2)$	100	-
STMixer [13]	$\mathcal{O}(N_a \times K)$	$\mathcal{O}(K \times D + N_a \times D)$	100	32
Ours	$\mathcal{O}(HW \times K)$	$\mathcal{O}(HW \times N_a + N_a \times N_c)$	15	4

  - decodes the sampled feature with Adaptive Feature Mixing, where the queries generate parameters of the modules through which they pass, similar to Sparse R-CNN [10].
- EVAD [2]
  - tokenizes video frames and prunes the tokens inside its encoder module. The tokens are mainly dropped besides the keyframe of the clip, hence it is named Keyframe-centric Token Pruning (KTP).
  - employs Feature Pyramid Network [4] for its localization module.
  - does not report its performance on CSN-152, so it is marked 'N/A'.

Efficiency analysis. We present computational complexities of the latest models in Table A2. The major bottleneck of TubeR [16] and EVAD [2] comes from exhaustive self-attention between HW feature vectors. Specifically, regarding that the flattened spatial size HW, which typically falls in between 300 and 400, is larger than other variables such as N_a or K, this operation creates a significant computational burden on both encoder and decoder architecture. On the other hand, STMixer [13] and ours resolve such computational burden of the encoder by sparsely collecting features from multi-level feature maps. However, STMixer generates adaptive parameters in its decoder architecture, and it results in complexities involving the channel dimension D, which typically is set to 256. Ours does not involve D, but still performs dense cross-attention using HW feature vectors. Yet, ours acquires greater efficiency by occupying much smaller number of actor candidates N_a.

In fact, the reason for our model being able to surpass other methods with comparably smaller N_a is benefited from the utilization of class queries. EVAD and STMixer extensively increase their number of actor candidates to obtain a pool of sufficiently diverse classification maps, which eventually improves their performance since such diversification increases a chance of capturing overlooked context needed to classify the action of the target actor. It is a crucial strategy for these models: because they expect the classification attention map for each actor to capture shared semantics of every action class, clues to identify some action classes (*e.g.*, clues that are distant from the actor) are easily missed when N_a is small. In contrast, our model can generate more precise attention maps for each actor and does not need to rely on large number of actor candidates. We utilize this saved memory to enable our model to process multiple frames with a single feedforward, maximizing the efficiency created from using class queries.

Table A3: Network configurations for each dataset.

Hyper-parameter	Description	AVA	JHMDB	UCF
W_0	resized resolution (length of the shorter side)	256	288	256
T_0	clip length	32 or 16	40	32
N_a	number of actor candidates	15	5	15
N_{enc}	number of encoder layers	6	3	3
N_{dec}	number of decoder layers	6	3	3
N_c	number of class queries (number of classes)	80	21	24

H Experimental setup

Network configurations. We set the channel dimension D to 256, number of feature levels L to 4, number of sampling points in the encoder K to 8. Otherwise, the model hyperparameters differ across three datasets, so we specify them in Table A3.

Training details. We use AdamW [7] optimizer and linearly warm up the learning rate in earlier epochs. The step scheduler is applied, so the learning rate decays by 0.1 every step milestone. Following prior work [2, 13], standard data augmentation techniques such as color jittering, random horizontal flipping, and PCA jittering [3] are applied. We present further training configurations in A4.

Additionally, we have observed that the model struggles to converge from scratch at once. Hence, we follow the training practice introduced in TubeR [16]. TubeR utilizes DETR [1] weights pretrained on COCO [5] for its transformer module and fine-tune these weights on AVA before the actual training. The fine-tuned transformer weights are then used for the final training: the transformer module of the model is trained from the pretrained weights, while the remaining modules are trained from scratch. Similarly, we first train the model from scratch, possibly on smaller backbones for ease of training, and then use the transformer weights acquired from the first stage for the final training.

I Additional Ablation Experiments

Effectiveness of box modulation. We adopt the box modulator function [6] to effectively utilize the positional prior of the actor box. Specifically, it enhances the modulation process by involving the height and width information of the actor box. Hence, the actor positional queries can also be applied without the box modulator, thereby encoding the positional information with positional coordinates without its height and width prior. It turns out that such modulation

Table A4: Training configurations for each dataset.

Training configurations	AVA	JHMDB	UCF
learning rate	1e-4	2e-4	2e-4
learning rate milestone	[8,11]	[50]	[12]
warmup start learning rate	1e-5	2e-6	2e-6
weight decay	1e-4	2e-4	2e-4
epochs	12	100	14
warmup epochs	3	10	3
loss coefficient for class (λ_{class})	10	4	4
loss coefficient for box (λ_{box})	5	5	5
loss coefficient for GIoU (λ_{giou})	2	2	2
loss coefficient for confidence (λ_{conf})	1	6	3

Table A5: Ablation experiments on utilizing the box modulator [6].

Method	AVA	UCF
w/ modulated actor positional queries	33.5	85.9
w/ ordinary actor positional queries	32.7	84.8

Table A6: The effect of self-attention between class queries.

Method	AVA	UCF
w/ class query SA	33.5	85.9
w/o class query SA	31.6	83.2

Fig. A4: Failure cases are marked with red boundaries.

improves the performance by 0.8% in AVA and by 0.2% in UCF, as demonstrated in Table A5.

Effectiveness of class query self-attention. We allow class queries to attend to each other before they enter the transformer module. The motivation behind this process is to let the model consider the relationship between different classes. For example, while the class *'stand'* and *'sit'* can never co-occur at the same time, *'eat'* and *'carry/hold (an object)'* frequently happen simultaneously. The effect of the self-attention between the class queries is shown in Table A6.

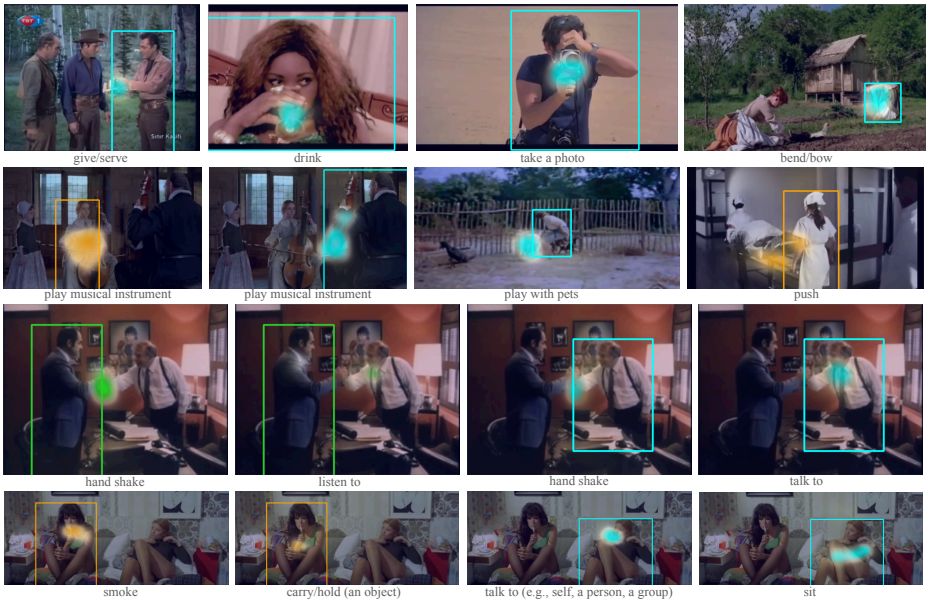


Fig. A5: Classification attention map on AVA.

J More qualitative results

Failure cases. We observe that our model occasionally uses spurious biases, especially in cases where a class covers various scenarios: *e.g.*, the class *'open'* is labeled not only when an actor opens an ordinary door but also a car door or a book. The same happens for *'dress/put on clothing'*, in which there are many ways to dress up various clothing (Fig. A4).

Classification attention map on other datasets. Fig. A5, A6, and A7 illustrate the classification attention map visualization results from AVA, JHMDB51-21, and UCF101-24, respectively. Our model concentrates on areas that are crucial for classification and they are not necessarily on the actor's body. For example, the model sees an animal (Fig. A5: *play with pets*, Fig. A7: *walking with dog*) or an object (Fig. A5: *push*, Fig. A6: *pour*, Fig. A7: *pole vault*, *trampoline jumping*, *biking*) which is an essential clue to understand the action happening in the video clip.

K Class-wise mAP comparison

We provide the comparison chart (Fig. A8) by the class labels in AVA. As the comparison group, we choose EVAD [2] and STMixer [13] trained on ViT-B [12]. Out of total 60 class labels, our model demonstrates the best performance on 33 labels, while EVAD and STMixer show their superiorities on 24 and 3 labels, respectively.

Fig. A6: Classification attention map on JHMDB51-21.

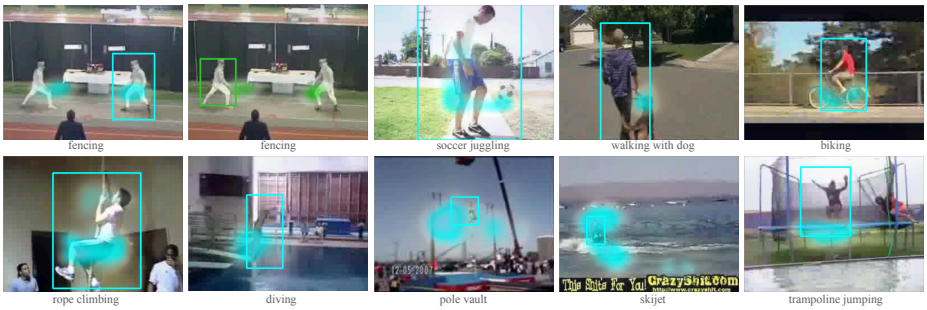


Fig. A7: Classification attention map for UCF101-24.

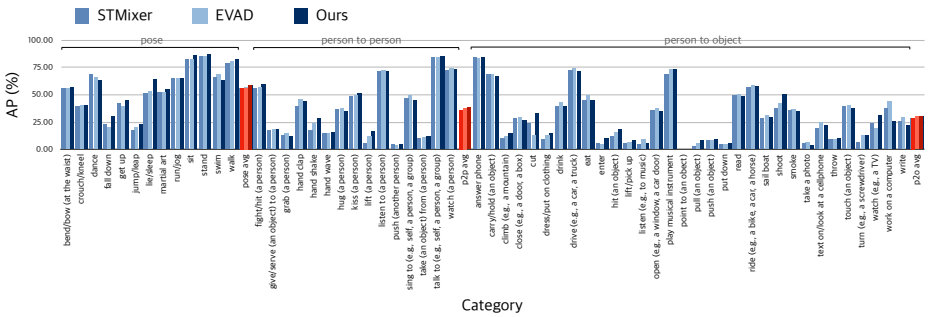


Fig. A8: Class-wise comparison among latest VAD models, STMixer [13] and EVAD [2]. Red bars are mean value that aggregates for each category type: pose, person to person, and person to object.

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. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)
4. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
5. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: ECCV (2014)
6. 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)
7. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. ICLR (2019)
8. 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)
9. Song, G., Liu, Y., Wang, X.: Revisiting the sibling head in object detector. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11563–11572 (2020)
10. Sun, P., Zhang, R., Jiang, Y., Kong, T., Xu, C., Zhan, W., Tomizuka, M., Li, L., Yuan, Z., Wang, C., et al.: Sparse r-cnn: End-to-end object detection with learnable proposals. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 14454–14463 (2021)
11. 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)
12. 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)
13. 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)
14. Wu, Y., Chen, Y., Yuan, L., Liu, Z., Wang, L., Li, H., Fu, Y.: Rethinking classification and localization for object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10186–10195 (2020)
15. Xu, M., Xiong, Y., Chen, H., Li, X., Xia, W., Tu, Z., Soatto, S.: Long short-term transformer for online action detection. Advances in Neural Information Processing Systems **34**, 1086–1099 (2021)
16. 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)
17. 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)