Efficient Video Action Detection with Token Dropout and Context Refinement

Lei Chen\textsuperscript{1}  Zhan Tong\textsuperscript{2}  Yibing Song\textsuperscript{3}  Gangshan Wu\textsuperscript{1}  Limin Wang\textsuperscript{1,4}\textsuperscript{*}
\textsuperscript{1}State Key Laboratory for Novel Software Technology, Nanjing University  
\textsuperscript{2}Ant Group  
\textsuperscript{3}AI\textsuperscript{3} Institute, Fudan University  
\textsuperscript{4}Shanghai AI Lab

leichen1997@outlook.com  zhantong.2023@gmail.com  yibingsong.cv@gmail.com  
gswu@nju.edu.cn  lmwang@nju.edu.cn

Abstract

Streaming video clips with large-scale video tokens impede vision transformers (ViTs) for efficient recognition, especially in video action detection where sufficient spatiotemporal representations are required for precise actor identification. In this work, we propose an end-to-end framework for efficient video action detection (EVAD) based on vanilla ViTs. Our EVAD consists of two specialized designs for video action detection. First, we propose a spatiotemporal token dropout from a keyframe-centric perspective. In a video clip, we maintain all tokens from its keyframe, preserve tokens relevant to actor motions from other frames, and drop out the remaining tokens in this clip. Second, we refine scene context by leveraging remaining tokens for better recognizing actor identities. The region of interest (RoI) in our action detector is expanded into temporal domain. The captured spatiotemporal actor identity representations are refined via scene context in a decoder with the attention mechanism. These two designs make our EVAD efficient while maintaining accuracy, which is validated on three benchmark datasets (i.e., AVA, UCF101-24, JHMDB). Compared to the vanilla ViT backbone, our EVAD reduces the overall GFLOPs by 43\% and improves real-time inference speed by 40\% with no performance degradation. Moreover, even at similar computational costs, our EVAD can improve the performance by 1.1 mAP with higher resolution inputs. Code is available at https://github.com/MCG-NJU/EVAD.

1. Introduction

Vision transformers (ViTs) have improved a series of recognition performance, including image classification [9, 46, 20] and object detection [5, 8, 40]. The image patches are regarded as tokens for ViT inputs for self-attention computations. When recognizing a video clip, we notice that the

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{Spatiotemporal token dropout from a keyframe-centric perspective. We maintain tokens from the keyframe of a video clip, preserve a small amount of tokens from non-keyframes based on actor motions, and drop out the remaining tokens in this clip. On the first row, we preserve tokens relevant to the waving hand in non-keyframes as they benefit recognizing the action ‘point to’. On the second row, we drop out tokens irrelevant to the action ‘entering’ in non-keyframes for efficient recognition.}
\end{figure}

tokens are from each frame and thus formulate a large-scale input for ViTs. These video tokens introduce heavy computations during training and inference, especially in the self-attention computation step. While attempts have been made to reduce vision tokens [36, 28, 12, 47] for fast computations, it is still challenging for video action detection (VAD) [41, 16, 26] to balance accuracy and efficiency. This is because in VAD we need to localize actors in each frame and recognize their corresponding identities. For each actor, the temporal motion in video sequences shall be maintained for consistent identification. Meanwhile, the scene context ought to be kept to differentiate from other actors. Sufficient video tokens representing both actor motions and scene context will preserve VAD accuracy, which leaves a
research direction on how to select them for efficient VAD.

In this paper, we preserve video tokens representing actor motions and scene context, while dropping out irrelevant tokens. Based on the temporal coherency of video clips, we propose a spatiotemporal token dropout from a keyframe-centric perspective. For each video clip, we can select one keyframe representing the scene context where all the tokens shall be maintained. Meanwhile, we select tokens from other frames representing actor motions. Moreover, we drop out the remaining video tokens in this clip. Fig. 1 shows two examples. On the first row, we maintain all the tokens in the keyframe, preserve the tokens in non-keyframe relating to the eyes and mouths associating to the action ‘talk to’, the waving hand of the right person associating to the action ‘point to’, and drop out the remaining video tokens. On the second row, we maintain all the tokens in the keyframe, preserve the tokens in non-keyframes relating to the movement of entering the car from outside which is associated with the action ‘enter’, and drop out the remaining video tokens. Our spatiotemporal token dropout maintains the relevant actor motions and scene context from the coherent video clips, while discarding the remaining irrelevant ones for efficient computations.

We develop spatiotemporal token dropout via a keyframe-centric token pruning module within the ViT encoder backbone. The keyframe is either uniformly sampled or manually predefined in the video clips. We select the middle frame of the input clip with box annotations as the keyframe by default. The feature map of the selected keyframe is retained completely for actor localization. Besides, we extract an attention map enhanced by this keyframe. This attention map guides token dropout in the non-keyframes. After localization, we need to classify each localized bounding box (bbox) for actor identification. As video tokens are incomplete in non-keyframes, the classification performance is inferior in the bbox regions where tokens have been dropped out. Nevertheless, the inherent temporal consistency in video benefits us to refine both actor and scene context from the remaining video tokens. We expand the localized bbxs in the temporal domain for RoIAlign [17] to capture token features related to actor motions. Then, we introduce a decoder to refine action features guided by the remaining video tokens in this clip. The decoder concatenates the actor and token features, and performs self-attention to produce enriched actor features for better identification. We find that after token dropout, the degraded action classification can be effectively recovered by using remaining video tokens for context refinement. The recovered performance is the same as that using the whole video tokens for action classification. Through this context refinement, we can maintain the VAD performance using reduced video tokens for efficient computations.

We conduct extensive experiments on three popular action detection datasets (AVA [16], UCF101-24 [41], JH-MDB [19]) to show the advantages of our proposed EVAD. In the ViT-B encoder backbone, for instance, we employ a keyframe-centric token pruning module three times with a keep rate $\rho$ of 0.7 (i.e., the dropout rate is 0.3). The encoder outputs 34% of the original video tokens, which reduces GFLOPs by 43% and increases throughput by 40% while achieving on-par performance. On the other hand, under the similar computation cost (i.e., a similar amount of tokens), we can take video clips in a higher resolution for performance improvement. For example, we can improve detection performance by 1.1 mAP when increasing the resolution from 256 to 288 on the short side with GFLOPs reduced by 22% and throughput increased by 10%. We also provide visualizations and ablation studies to show how our EVAD performs spatiotemporal token dropout to eliminate action irrelevant video tokens.

2. Related Works

2.1. Spatio-temporal Action Detection

Current state-of-the-art methods [14, 13, 51, 44, 33, 6] commonly adopt a two-stage pipeline with two separated backbones, i.e., a 2D backbone for actor localization on keyframes and a 3D backbone for video feature extraction. Some previous approaches [23, 42] simplified the pipeline by training two backbones in an end-to-end manner, which suffer from heavy complexity and optimization difficulty. Most recent methods [15, 7, 58, 54] utilize a unified backbone to perform action detection. VAT [15] is a transformer-style action detector to aggregate the spatiotemporal context around the target actors. WOO [7] and TubeR [58] are query-based action detectors following the detection frameworks of [43, 5] to predict actor bounding boxes and action classes. STMixer [54] is a one-stage query-based detector to adaptively sample discriminative features. Several newly transformer-based methods [10, 52, 45] apply a ViT variant backbone and achieve competitive results following the two-stage pipeline. Moreover, there are also emerging methods [2, 25, 56] that focus on the training paradigm of deep networks for action detection task.

2.2. Spatio-temporal Redundancy

Spatial redundancy. The success of vision transformers has inspired various works [32, 18, 36, 39, 28, 38, 12] to explore the spatial redundancy of intermediate tokens. DynamicViT [36] observes that accurate image recognition is based on a subset of most informative tokens and designs a dynamic token sparsification framework to prune redundant tokens. EViT [28] calculates the attentiveness of the class token to each token and identifies the top-k tokens using the attentiveness value. ATS [12] introduces a differentiable adaptive token sampler for adaptively sampling significant
our pipeline. The pipeline of end-to-end video action detection proposed in this paper is based on the vanilla ViT, as shown in Fig. 2. To alleviate the computational bottleneck caused by joint space-time attention, we devise an efficient video action detector (EVAD) with an encoder-decoder architecture with respect to the characteristics of action detection. EVAD enables the encoder with token pruning to remove the redundant tokens, and the decoder to refine actor spatiotemporal features. Following the setting in WOO [7], we utilize multiple intermediate spatial feature maps of the keyframe in the encoder for action localization, and the spatiotemporal feature map output from the last encoder layer for action classification. **Token Pruning.** We design a keyframe-centric token pruning module to progressively reduce the redundancy of video data and ensure that few and promising tokens are delivered for action localization and classification. The specific token pruning module will be detailed in the next subsection. **Localization.** We up-sample or down-sample intermediate ViT feature maps of the keyframe to produce multiscale feature maps and send them to feature pyramid network (FPN) [30] for multiscale fusion. The localization branch predicts $n$ candidate boxes in the keyframe by a query-based method, same as in [43, 7].**Classification.** We remap the compact context tokens from ViT encoder into a spatiotemporal feature map with a regular structure. Then, we conduct 3D RoIAlign on the restored feature map using extended prediction boxes from the localization branch to obtain $n$ actor RoI features. Subsequently, we utilize a context refinement decoder ($CRD$) for actor feature refinement and relational modeling between actor RoI features and compact context tokens from ViT encoder.

### 3.2. Keyframe-centric Token Pruning

The high spatiotemporal redundancy of video with similar semantic information between adjacent frames makes it
possible to perform token pruning with a high dropout rate on video-level recognition. In this paper, we design a keyframe-centric token pruning module, as shown in Fig. 3. We split all spatiotemporal tokens into keyframe tokens and non-keyframe tokens. And we preserve all keyframe tokens for accurate actor localization in the keyframe.

**Non-keyframe token pruning.** For the importance measure of non-keyframe tokens, we refer to the approach of EViT [28] in image classification, using the pre-computed attention map to represent the importance of each token without additional learnable parameters and nontrivial computational costs. As shown in the top part of Fig. 3, we first average the num_heads dimension of the attention map to obtain an $N \times N$ matrix, which represents the attentiveness between tokens (omitting the batch size). For example, $\text{attn}(i, j)$ denotes the token $i$ considers the importance of token $j$. Since there is no classification token specifically for video-level recognition, we calculate the average importance score of each token by $I_j = \frac{1}{N} \sum_{i=1}^{N} \text{attn}(i, j)$. Then we identify the top $(N \times \rho - N_1)$ tokens from the $N_2$ non-keyframe tokens in descending order by importance scores, where $N, N_1, N_2$ represent the number of all, keyframe and non-keyframe tokens respectively, and $\rho$ represents token keeping rate. Normally, the keyframe contains the most accurate semantic information for the current sample, and other frames away from the keyframe incur nontrivial information bias. Thus, it is practical to conduct token pruning guided by the keyframe. To this end, we insert a Keyframe Attentiveness Enhancement step between acquiring the attention map and calculating the importance of each non-keyframe token. As presented in Fig. 3, we apply a greater weight value to the keyframe queries, thereby retaining tokens with higher correlations to keyframe tokens. The importance score of each token is updated as follows:

$$I_j = \frac{1}{N} \sum_{i=1}^{N} w_{kf} \cdot \text{attn}(i, j), \quad i \in (0, N_1) \backslash \{N_1, N\}$$

where we assume the first $N_1$ tokens belong to the keyframe and the weight value $w_{kf}$ is a hyper-parameter will be ablated in Sec. 4.2. In other words, we discard some tokens that only have a high response with non-keyframes, which may not be high-quality tokens. Only if the non-keyframe becomes the keyframe of previous/next samples, these highly responsive tokens are high-quality. By dropping out these redundant tokens with a high response to non-keyframes, we can further reduce the number of tokens. After the execution of token pruning, we send the preserved tokens to the follow-up FFN of the encoder layer.

The first token pruning is started at 1/3 of encoder layers to ensure that the model is capable of high-level semantic representation. Then, we perform token pruning every 1/4 of the total layers, discarding the redundant tokens and keeping the effective ones. As shown in Fig. 4, we visualize the preserved tokens of each pruning layer in ViT-B, and the model is able to retain important cues such as people and chairs. We present more visualizations in Appendix § D. After multiple pruning, the number of tokens is drastically reduced, which enables the model to save computation costs and accelerate training and inference process.

### 3.3 Video Action Detection

**Actor localization branch.** Benefit from preserving all
keyframe tokens in Sec. 3.2, we can obtain multiple complete keyframe feature maps. We then up-sample or down-sample those keyframe feature maps for generating hierarchical features from the plain ViT. We then introduce a query-based actor localization head, inspired by Sparse R-CNN [43], to detect actors in the keyframe. The details of the localization head are provided in Appendix § B. Finally, the outputs of actor localization branch are n prediction boxes in the keyframe and corresponding actor confidence scores.

**Action classification branch.** Different from conventional feature extraction, EVAD produces M discrete video tokens. We need to restore the spatiotemporal structure of video feature map and then can perform location-related operations such as RoIAlign. We initialize a blank feature map shaped as \((T/2, H/16, W/16)\), fill the preserved tokens into this feature map according to their corresponding spatiotemporal positions, and pad the rest with zeros.

Next, we use the boxes generated by the localization branch to extract actor RoI features via 3D RoIAlign for subsequent action prediction. Due to actor movement or camera translation, actor spatial position is changed across frames, and using the keyframe box for 3D RoIAlign cannot obtain partial actor feature deviated from the box. Directly extending the scope of the box to cover whole motion trajectory might harm actor feature representation by introducing background or other interfering information. Nevertheless, in EVAD feature extraction stage, the interfering ones in the video are progressively eliminated, thus we can properly extend the box scope to add the deviated feature. The results in Sec. 4.2 demonstrate the ability of EVAD to cover large motion with extended boxes, while its counterpart ViTs impairs actor native feature representation.

We observe that directly applying token pruning methods of vision classification can greatly reduce the number of tokens involved in the calculations, but have a negative impact on final detection performance. Video action detection requires localizing and classifying the actions of all actors, but token pruning algorithms lead to incoherent actor features in space-time. As shown in Fig. 4, the static appearance of people sitting in the chairs is not retained completely in every frame. In the encoder, pair-wise self-attention is capable of modeling global dependency among tokens. The semantic information of the dropout tokens within the actor regions can be incorporated into some preserved tokens. Thus, we can recover the removed actor features from the preserved video tokens. To this end, we design a context refinement decoder to refine actor spatiotemporal representation. Concretely, we concatenate n actor RoI features with M video tokens and feed them into a sequentially-stacked transformer decoder consisting of MHSA and FFN. Guiding by the preserved tokens, actor features can enrich themselves with actor representation and motion information from the decoder can be used as a kind of relational modeling modules [42, 57, 15, 44, 33] to capture inter-actor and actor-context interaction information. The actor feature refining and relational modeling capabilities of our decoder will be ablated in Sec. 4.2.

The n refined actor features output by the decoder are retrieved and passed through a classification layer to make final action prediction.

### 4. Experiments

#### 4.1. Experimental Setup

**Datasets.** We evaluate our EVAD on three common datasets for video action detection: AVA [16], UCF101-24 [41] and JHMDB [19]. AVA is a large-scale benchmark and contains 299 15-minute videos, divided into 211k training clips and 57k validation clips. The videos are annotated at 1FPS for boxes and labels. Following the standard evaluation protocol, we report our performance on 60 common action classes. UCF101-24 is a subset of UCF101. It includes 3,207 videos from 24 sports classes. Each video contains a single action class. Following the common practice, we report the performance on split-1. JHMDB contains 928 trimmed videos from 21 action classes. We report the average results over all three splits.

**Evaluation criteria.** We evaluate the performances with frame level mean Average Precision (mAP) under IoU threshold of 0.5 without multiple scales and flips for fair comparisons. We measure the throughput on a single A100 GPU with a batch size of 8 to estimate the average number of images that can be processed in one second. We specify a sample of video clip containing 16 frame images by default.

Our implementation details are described in Appendix § B.

#### 4.2. Ablation Studies

We conduct in-depth ablation studies to investigate the effectiveness of our design in EVAD. All results are reported on AVA v2.2 with a 16-frame VideoMAE ViT-B backbone pre-trained and fine-tuned on Kinetics-400 [22].

**Inference w/o re-training.** EVAD does not add any additional learnable parameters for token pruning. Thus, we directly add our keyframe-centric token pruning module on the off-the-shelf pre-trained ViTs and show the performance of the models with different token keep rates in Fig. 5. We observe that directly applying token pruning negatively affects the model performance, and we speculate that the model needs re-training to adapt to a dynamic number of tokens at different layers. Then, we incorporate our token pruning in the training process, and retrain EVAD with token keep rate of 0.7 and infer on multiple keep rates. As
Table 1: Ablation experiments on A VA v2.2. All models here adopt 16-frame vanilla ViT-B as backbone and keep rate $\rho$ is set to 0.7 as the default setting unless otherwise specified.

<table>
<thead>
<tr>
<th>Pruning strategy</th>
<th>$\rho=0.7$</th>
<th>$\rho=1.0$</th>
<th>$\rho=0.7$</th>
<th>$\rho=1.0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear (baseline)</td>
<td>30.5</td>
<td>25.8</td>
<td>30.5</td>
<td>25.8</td>
</tr>
<tr>
<td>WOO [7]</td>
<td>29.9</td>
<td>158.9</td>
<td>30.5</td>
<td>134.2</td>
</tr>
<tr>
<td>Our decoder</td>
<td>30.5</td>
<td>134.2</td>
<td>30.5</td>
<td>134.2</td>
</tr>
</tbody>
</table>

Figure 5: Comparison on the performance of EVAD without re-training and re-training with keep rate $\rho=0.7$. The numbers in ‘()’ indicate the gap of the corresponding values to re-training with the target keep rate.

shown in Fig. 5, the result on each keep rate is nearly comparable to the re-training performance with the target keep rate. This illustrates that our token pruning module allows to adapt to different levels of computation costs after re-training once.

Classification branch. To illustrate the necessity of refining actor spatiotemporal features when using token dropout for efficient action detection, we conduct two groups of experiments, i.e., the encoder with keyframe-centric token pruning ($\rho=0.7$) and without token pruning (keep rate $\rho=1.0$), and examine the effect of context refinement decoder for actor feature refinement and relational modeling. Without token pruning, the decoder is used to conduct actor-context modeling to capture action interaction information. It enables +4.1 mAP gains over the baseline and outperforms the well-designed WOO head with fewer parameters. When token pruning is applied, partial actor features are eliminated during the pruning process, and the linear model is much worse than the baseline. Nevertheless, the strong feature refining capability of decoder compensates this gap and achieves a comparable performance with $\rho=1.0$.

Localization branch. To explore the localization capability of EVAD and its impact on the overall performance, we directly use the boxes from an off-the-shelf detector [37], which is commonly adopted in previous two-stage pipeline [14, 51, 44]. Our results shows that though utilizing the off-the-shelf detector can lead to an increase of +1.7 mAP, this comes at the cost of additional computational complexity, such as 246 GFLOPs for Faster R-CNN, and results in a more complex pipeline. The localization branch in our EVAD is lightweight with the cost of only 13.5 GFLOPs, and enjoys a simple end-to-end manner.

Pruning strategy. We then ablate different token pruning strategies with $\rho=0.7$. (1) Random masking performs a random sampling of non-keyframe tokens before entering the encoder. (2) Token pruning based on CLS token, as in [28], uses a learnable CLS token pre-trained on video recognition [22] responsible for calculating the importance of non-keyframe tokens. (3) GAP denotes pruning guided by global average pooling of all token attentive values. As shown in Table 1b, both attention-based approaches outperform random masking and achieve comparable performance with $\rho=1.0$, demonstrating that those models have the ability to preserve complete action semantics. We choose the latter with higher flexibility as the default setting.

Keyframe. In Sec 3.2, we emphasize the necessity of preserving all keyframe tokens. In Table 1c, we compare the two strategies of keeping all keyframe tokens and unified pruning by treating keyframe tokens as normal tokens, the latter discarding some important tokens in the keyframe and leading to performance degradation.

Decoder input. To verify if some discarded tokens are important for action detection, we save those discarded ones...
Figure 6: RoI extension illustration. The red boxes denote the prediction box in the keyframe and are copied to adjacent frames. The blue boxes denote the extended boxes to capture large motion.

when performing token pruning and feed all tokens into the decoder. As shown in Table 1d, using all tokens does not improve the performance, indicating that the tokens preserved by the model contain sufficient action semantics.

**Decoder depth.** The results in Table 1e show that stacking 6 layers in the decoder has the highest mAP, and more than 6 layers cause performance degradation due to over-fitting, so we use depth=6 as the default setting.

**RoI extension.** Due to human large motion, the box copied from the keyframe cannot cover the whole motion trajectory. Intuitively, we can solve it by properly extending the box scope, as shown in Fig. 6, extending the scope of the red box in the keyframe to cover the swimming person. In Table 1f, the model without token pruning shows a decreasing trend in performance as the proportion of box extending increases. Instead, the token pruning method achieves the best performance when the extended ratio reaches (0.4,0.2), where the box is 1.68× of the original box, in which our pruning mechanism eliminates the interference information introduced by the extension. It is worth mentioning that our method increases 15.7% AP on ‘swim’ category. We observe that large displacements frequently happen in ‘swim’, and our token pruning combined with RoI extension can alleviate the effect of human movements to some extent.

**Attentiveness enhancement.** In Table 2b, there is a significant decrease in performance when the ρ is reduced from 0.7 to 0.6. It indicates that the number of tokens retained at ρ=0.6 is insufficient to contain complete information. In Sec. 3.2, we analyze the portion of preserved tokens only has high responses to non-keyframes, which could be redundant to the current sample. Thus, we conduct experiments on ρ=0.6 to explore the feasibility of further reducing the number of tokens in Table 1g. We ablate the attention weight of keyframe queries, where \(w_{k_f}=4\) yields the best result. It shows that the keyframe plays a greater role in identifying the importance of non-keyframe tokens, leading to a lower redundancy of preserved tokens.

**Discussion on the preserved tokens.** To explore what the preserved tokens correspond to, we compute the ratio of the preserved tokens inside the GT area on AVA validation set, as shown in Fig. 7. We observe that the average GT area is 56.4% of the input image, and 74.4% of the preserved tokens belong to the GT area (56.4%). This evidence indicates that the proposed token pruning strategy allows our EVAD to focus on tokens in the GT area even without explicit box supervision.

### 4.3. Efficiency Analysis

To verify the efficiency of EVAD, we measure several facets of computation, inference speed and training cost at multiple resolutions.

First, we compare saved computation and improved inference speed with different keep rates, measured by GFLOPs and throughput, respectively, as shown in Table 2. When the keep rate \(\rho\) is in the range of [0.7, 1.0), the performance is comparable to \(\rho=1.0\), which indicates that high spatiotemporal redundancy exists in action detection and our method can effectively remove redundancy and maintain critical cues. The model with \(\rho=0.7\) achieves the best performance-efficiency trade-off for both resolutions. For the short side of 224, the detection mAP improves 0.2%, GFLOPs decrease by 40%, and throughput increases by 38%; for the short side of 288, mAP improves 0.2%, GFLOPs decrease by 43%, and throughput increases by 40%. Further reducing the keep rate degrades the performance because the number of tokens kept is insufficient to contain complete semantic information.

Then, we fix the keep rate \(\rho\) at 0.7 and compare the performance and efficiency gains of EVAD at higher resolutions, as shown in Table 3. We observe that: (1) EVAD with \(\rho=0.7\) can obtain comparable performance to the model with \(\rho=1.0\) at the same resolution with significantly lower computation and faster inference speed. (2) EVAD from a higher resolution provides a stable performance boost with relatively lower computation and faster speed. For instance, EVAD can improve performance by 1.1% when increasing...
We compare our EVAD with the state-of-the-art methods on AVA v2.2 in Table 5a. The result on AVA v2.1 is provided in Appendix § C. To the best of our knowledge, the prevailing best model on AVA is VideoMAE, a two-stage model that requires an off-the-shelf person detector to pre-compute person proposals. Using the same pre-trained backbone, we can obtain comparable performance to Video-MAE, with mAP of 32.3 vs. 31.8 on ViT-B and 39.7 vs. 39.3 on ViT-L, and surpass other two-stage models. Compared to end-to-end models such as WOO and TubeR, we also have a significant performance gain, while benefiting

Table 2: Comparison on the performance and efficiency of EVAD with different token keep rates. The green numbers indicate the gap of the corresponding values to EVAD with $\rho=1.0$.

<table>
<thead>
<tr>
<th>keep rate $\rho$</th>
<th>mAP</th>
<th>GFLOPs</th>
<th>throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>30.6</td>
<td>186.3</td>
<td>(+17%)</td>
</tr>
<tr>
<td>0.8</td>
<td>30.5</td>
<td>157.0</td>
<td>(+30%)</td>
</tr>
<tr>
<td>0.7</td>
<td>30.7</td>
<td>134.2</td>
<td>(+40%)</td>
</tr>
<tr>
<td>0.6</td>
<td>30.2</td>
<td>116.3</td>
<td>(+48%)</td>
</tr>
</tbody>
</table>

Figure 8: Comparison on GPU memory and training time required of EVAD training with different token keep rates. The blue and green numbers indicate the gap of the corresponding values to EVAD with $\rho=1.0$.

Table 3: Comparison on the performance and efficiency improvements of EVAD with higher resolutions.

<table>
<thead>
<tr>
<th>model</th>
<th>backbone</th>
<th>mAP</th>
<th>GFLOPs</th>
<th>throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>WOO [7]</td>
<td>SFR101</td>
<td>28.3</td>
<td>252</td>
<td>147</td>
</tr>
<tr>
<td>WOO [7]</td>
<td>ViT-B</td>
<td>30.0</td>
<td>378</td>
<td>176</td>
</tr>
<tr>
<td>TubeR [58]</td>
<td>CSN-152</td>
<td>33.6</td>
<td>240</td>
<td>64*</td>
</tr>
<tr>
<td>VideoMAE [45]</td>
<td>ViT-B</td>
<td>31.8</td>
<td>180+246</td>
<td>N/A</td>
</tr>
<tr>
<td>VideoMAE [45]</td>
<td>ViT-L</td>
<td>39.3</td>
<td>597+246</td>
<td>N/A</td>
</tr>
<tr>
<td>EVAD, $\rho=0.7$</td>
<td>ViT-B</td>
<td>32.3</td>
<td>243</td>
<td>240</td>
</tr>
<tr>
<td>EVAD, $\rho=0.7$</td>
<td>ViT-L</td>
<td>39.7</td>
<td>737</td>
<td>153</td>
</tr>
</tbody>
</table>

Table 4: Comparison on the performance and efficiency of EVAD and other state-of-the-art methods. † denotes the results re-implemented by us for fair comparison. *: the code provided by TubeR does not include the implementation of its long-term context head, so the actual throughput will be less than we measured. ‘N/A’ indicates that the two-stage pipeline utilized in VideoMAE is not applicable to measure throughput.

We first observe that vanilla ViT allows WOO to achieve faster inference speed compared to traditional CNN-based back-
from the structural nature of transformers and token pruning with a low keep rate, we have a faster inference speed than CNN-based models and are more friendly to real-time action detection. Moreover, we can achieve slightly better performance than STMixer. STMixer is the most recent end-to-end model that designs a decoder to sample discriminative features. Orthogonal to it, our EVAD is devised for efficient video feature extraction, and further combining the two may yield better detection performance.

To demonstrate the generalizability of EVAD, we further verify our model on JHMDB and UCF101-24. As shown in Table 5b, we achieve state-of-the-art performance on both datasets with comparable improvement. Different from the experimental results on AVA, using the keep rate of 0.6 can lead to similar or better results than the model without token pruning on these two datasets. We consider the reason is that the scenarios of JHMDB and UCF01-24 are simpler and do not require complex relational modeling of actor-actor and actor-object, and hence can preserve a fewer number of tokens. In addition, the JHMDB dataset is small and using a lower keep rate for training might alleviate over-fitting and learn a better feature representation. We also provide the video-mAP results on both datasets in Appendix § C.

5. Conclusion and Future Work

Motivated by expensive computational costs of transformers with a video sequence and high spatiotemporal redundancy in video action detection, we design an Efficient Video Action Detector (EVAD) by dropping out spatiotemporal tokens and refining scene context to enable efficient transformer-based action detection. EVAD can achieve comparable performance to the vanilla ViT while considerably reducing computational costs and expediting inference speed, and it can reach the state-of-the-art on three popular action detection datasets. We hope that EVAD can serve as an efficient end-to-end baseline for future studies.

One limitation of our approach is that EVAD requires re-training once to take the benefits of reduced computations and faster inference from removing redundancy. A potential research approach is to explore transformer-adaptive token pruning algorithms. Moreover, we follow the end-to-end framework of WOO to verify the efficiency and effectiveness of EVAD, but WOO is still a ‘two-stage’ pipeline, which sequentially conducts actor localization and action classification modules. In future work, we aim to integrate those two modules into a unified head, which can reduce the inference time of passing through the detector head and hence amplify the efficiency benefits of EVAD.

Acknowledgements. This work is supported by National Key R&D Program of China (No. 2022ZD0160900), National Natural Science Foundation of China (No. 62076119, No. 61921006, No. 62072323), Fundamental Research Funds for the Central Universities (No. 020214380091, No. 020214380099), and Collaborative Innovation Center of Novel Software Technology and Industrialization.
References


