STMixer: A One-Stage Sparse Action Detector

Tao Wu[1,*] Mengqi Cao[1,*] Ziteng Gao[1] Gangshan Wu[1] Limin Wang[1,2,✉]
[1]State Key Laboratory for Novel Software Technology, Nanjing University [2]Shanghai AI Lab
{wt,mg20370004}@smail.nju.edu.cn, gzt@outlook.com, {gswu, lmwang}@nju.edu.cn

Abstract

Traditional video action detectors typically adopt the two-stage pipeline, where a person detector is first employed to generate actor boxes and then 3D RoIAlign is used to extract actor-specific features for classification. This detection paradigm requires multi-stage training and inference, and cannot capture context information outside the bounding box. Recently, a few query-based action detectors are proposed to predict action instances in an end-to-end manner. However, they still lack adaptability in feature sampling and decoding, thus suffering from the issues of inferior performance or slower convergence. In this paper, we propose a new one-stage sparse action detector, termed STMixer. STMixer is based on two core designs. First, we present a query-based adaptive feature sampling module, which endows our STMixer with the flexibility of mining a set of discriminative features from the entire spatiotemporal domain. Second, we devise a dual-branch feature mixing module, which allows our STMixer to dynamically attend to and mix video features along the spatial and the temporal dimension respectively for better feature decoding. Coupling these two designs with a video backbone yields an efficient end-to-end action detector. Without bells and whistles, our STMixer obtains the state-of-the-art results on the datasets of AVA, UCF101-24, and JHMDB.

1. Introduction

Video action detection [14, 18, 20, 30, 32, 44, 46] is an important problem in video understanding, which aims to recognize all action instances present in a video and also localize them in both space and time. It has drawn a significant amount of research attention, due to its wide applications in many areas like security and sports analysis.

Since the proposal of large-scale action detection benchmarks [16, 22], action detection has made remarkable progress. This progress is partially due to the advances of video representation learning such as video convolution neural networks [5, 11, 39–41, 45, 50] and video transformers [1, 3, 9, 27, 38, 43, 52].

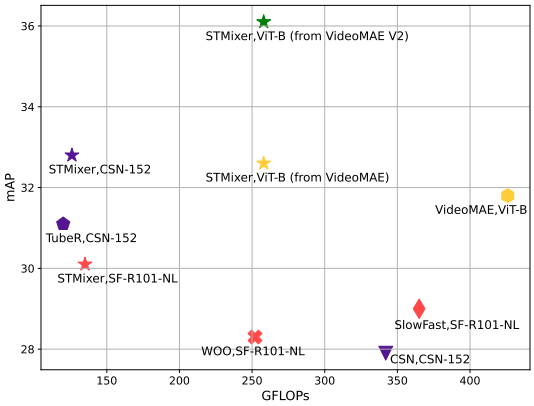

Figure 1. **Comparion of mAP versus GFLOPs.** We report detection mAP on AVA v2.2. The GFLOPs of CSN, SlowFast, and VideoMAE are the sum of Faster RCNN-R101-FPN detector GFLOPs and classifier GFLOPs. Different methods are marked by different makers and models with the same backbone are marked in the same color. The results of CSN are from [53]. Our STMixer achieves the best effectiveness and efficiency balance.

Most current action detectors adopt the two-stage Faster R-CNN-alike detection paradigm [31]. They share two basic designs. First, they use an auxiliary human detector to generate actor bounding boxes in advance. The training of the human detector is decoupled from the action classification network. Second, in order to predict the action category for each actor box, the RoIAlign [17] operation is applied on video feature maps to extract actor-specific features. However, these two-stage action detection pipeline has several critical issues. First, it requires multi-stage training of person detector and action classifier, which requires large computing resources. Furthermore, the RoIAlign [17] operation constrains the video feature sampling inside the actor bounding box and lacks the flexibility of capturing context information in its surroundings. To enhance RoI features, recent works use an extra heavy module that introduces interaction features of context or other actors [29, 36].

Recently sparse query-based object detector [4, 35, 54] has brought a new perspective on detection tasks. Several

*: Equal contribution. ✉: Corresponding author.

query-based sparse action detectors [6, 53] are proposed. The key idea is that action instances can be represented as a set of learnable queries, and detection can be formulated as a set prediction task, which could be trained by a matching loss. These query-based methods detect action instances in an end-to-end manner, thus saving computing resources. However, the current sparse action detectors still lack adaptability in feature sampling or feature decoding, thus suffering from inferior accuracy or slow convergence issues. For example, building on the DETR [4] framework, TubeR [53] adaptively attends action-specific features from single-scale feature maps but perform feature transformation in a static mannner. On the contrary, though decoding sampled features with dynamic interaction heads, WOO [6] still uses the 3D RoIAlign [17] operator for feature sampling, which constrains feature sampling inside the actor bounding box and fails to take advantage of other useful information in the entire spatiotemporal feature space.

Following the success of adaptive sparse object detector AdaMixer [12] in images, we present a new query-based one-stage sparse action detector, named STMixer. Our goal is to create a simple action detection framework that can sample and decode features from the complete spatiotemporal video domain in a more flexible manner, while retaining the benefits of sparse action detectors, such as end-to-end training and reduced computational cost. Specifically, we come up with two core designs. First, to overcome the aforementioned fixed feature sampling issue, we present a query-guided adaptive feature sampling module. This new sampling mechanism endows our STMixer with the flexibility of mining a set of discriminative features from the entire spatiotemporal domain and capturing context and interaction information. Second, we devise a dual-branch feature mixing module to extract discriminative representations for action detection. It is composed of an adaptive spatial mixer and an adaptive temporal mixer in parallel to focus on appearance and motion information, respectively. Coupling these two designs with a video backbone yields a simple, neat, and efficient end-to-end actor detector, which obtains a new state-of-the-art performance on the datasets of AVA [16], UCF101-24 [33], and JHMDB [19]. In summary, our contribution is threefold:

- We present a new one-stage sparse action detection framework in videos (STMixer). Our STMixer is easy to train in an end-to-end manner and efficient to deploy for action detection in a single stage.
- We devise two flexible designs to yield a powerful action detector. The adaptive sampling can select the discriminative feature points and the adaptive feature mixing can enhance spatiotemporal representations.
- STMixer achieves a new state-of-the-art performance on three challenging action detection benchmarks.

2. Related Work

Action detectors using an extra human detector. Most current action detectors [9–11, 29, 36, 38, 47] rely on an auxiliary human detector to perform actor localization on the keyframes. Typically, the powerful Faster RCNN-R101-FPN [31] detector is used as the human detector, which is first pre-trained on the COCO [26] dataset and then fine-tuned on the AVA [16] dataset. With actor bounding boxes predicted in advance, the action detection problem is reduced to a pure action classification problem. The RoIAlign [17] operation is applied on the 3D feature maps extracted by a video backbone to generate actor-specific features. SlowFast [11] and MViT [9] directly use the RoI features for action classification. However, RoI features only contain the information inside the bounding box but overlook context and interaction information outside the box. To remedy this inherent flaw of RoI features, AIA [36] and ACARN [29] resort to using an extra heavy module that models the interaction between the actor and context or other actors. The models with an extra human detector require two-stage training and reference, which is computing resources unfriendly. Besides, they suffer from the aforementioned issue of fixed RoI feature sampling.

End-to-end action detectors. Methods of another research line use a single model to perform action detection. Most of them [6, 13, 21, 34] still follow the two-stage pipeline but simplify the training process by jointly training the actor proposal network and action classification network in an end-to-end manner. These methods still have the issue of fixed RoI feature sampling. To remedy this, VTr [13] attends RoI features to full feature maps while ACRN [34] introduces an actor-centric relation network for interaction modeling. Recently, several one-stage action detectors [24, 53] are proposed. MOC [24] is a point-based dense action detector, which uses an image backbone for frame feature extraction. It concatenates frame features along the temporal axis to form the video feature maps. Each point on the feature maps is regarded as an action instance proposal. The bounding box and action scores of each point are predicted by convolution. MOC [24] relies more on appearance features, lacks temporal and interaction modeling and requires post-process. Building on DETR [4] framework, TubeR [53] is a query-based action detector. TubeR [53] adaptively samples features from single-scale feature maps, neglecting multi-scale information which is important for detection tasks. As DETR [4], TubeR [53] transforms features in a static manner, resulting in slower convergence.

Inspired by AdaMixer [12], we propose a new one-stage query-based detector for video action detection. Different from former query-based action detectors [6, 53], we adaptively sample discriminative features from a multi-scale spatiotemporal feature space and decode them with a more flexible scheme under the guidance of queries.

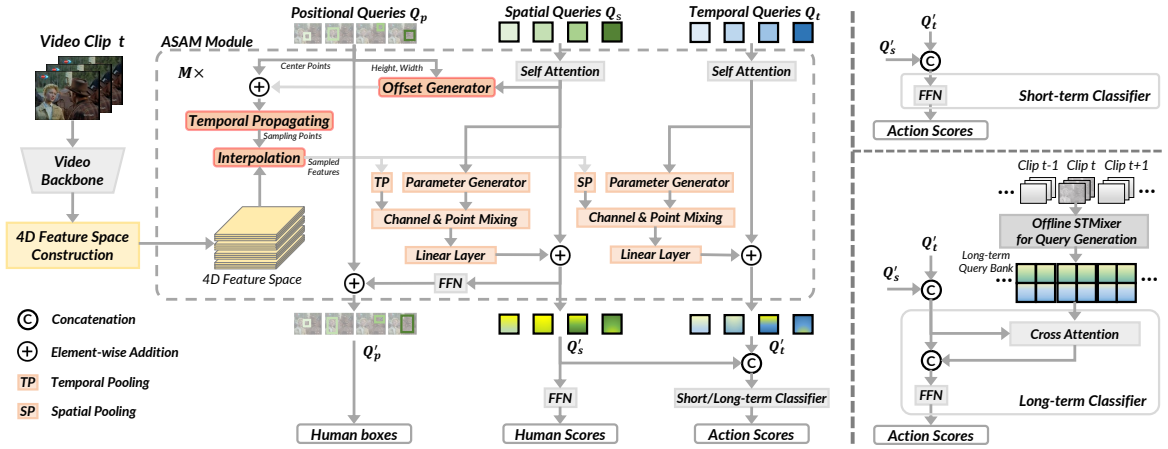

Figure 2. **Pipeline of STMixer**. In the left, we present the overall STMixer framework. A video clip is input to the video backbone for feature extraction and a 4D feature space is constructed based on the feature maps (see Section 3.1). Then, a decoder containing M ASAM modules iteratively performs adaptive feature sampling (see Section 3.3) from the 4D feature space and adaptive mixing (see Section 3.4) on the sampled features under the guidance of a set of learnable queries. Inversely, the queries are updated with mixed features. Optionally, a short-term or long-term classifier can be used for action scores prediction, whose detailed structures are illustrated in the right. The long-term classifier refers to the long-term query bank produced by an offline STMixer for long-term information (see Section 3.5).

3. Method

This section presents our one-stage sparse action detector, called STMixer. The overall pipeline of STMixer is shown in Figure 2. Our STMixer comprises a video backbone for feature extraction, a feature space construction module, and a sparse action decoder composed of M adaptive sampling and adaptive mixing (ASAM) modules followed by prediction heads. As a common practice, we set the middle frame of an input clip as the keyframe. We first use the video backbone to extract feature maps for the video and a 4D feature space is constructed based on the feature maps. Then, we develop the sparse action decoder with a set of learnable queries. Under the guidance of these queries, we perform adaptive feature sampling and feature mixing in the 4D feature space. The queries are updated iteratively. Finally, we decode each query as a detected action instance of action scores, human scores, and a human box. We will describe the technical details of each step in the next subsections.

3.1. 4D Feature Space Construction

Hierarchical video backbone. Formally, let $X_z \in \mathbb{R}^{C_z \times T \times H_z \times W_z}$ denote the feature map of convolution stage z of the hierarchical backbone, where $z \in \{2, 3, 4, 5\}$, C_z stands for the channel number, T for time, H_z and W_z for the spatial height and width. The stage index z can be seen as the scale index of the feature map as X_z has the downsampling rate of 2^z. We first transform each feature map X_z to the same channel D by $1 \times 1 \times 1$ convolution. Then, we rescale the spatial shape of each stage feature map to $H_2 \times W_2$ by simple nearest-neighbor interpolation and align them along the x- and y-axis. The four dimensions of the constructed feature space are x-, y-, t-axis, and scale index z, respectively. This process is illustrated in Figure 3.

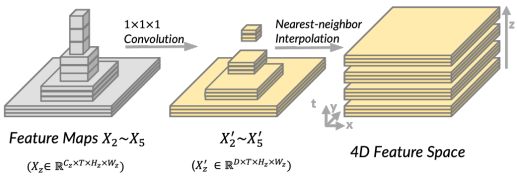

Figure 3. **4D feature space construction for hierarchical video backbone.** We construct 4D feature space on multi-scale 3D feature maps from hierarchical video backbone by simple lateral convolution and nearest-neighbor interpolation. The four dimensions of the 4D feature space are x-, y-, t-axis, and scale index z.

Plain ViT backbone. To make STMixer framework compatible with plain ViT [7] backbone, inspired by ViTDet [23], we construct 4D feature space based on the last feature map from ViT backbone. Specifically, with the output feature map of the default downsampling rate of 2^4, we first produce hierarchical feature maps $\{X_z\}$ of the same channel number D using convolutions of spatial strides $\{\frac{1}{4}, \frac{1}{2}, 1, 2\}$, where a fractional stride indicates a deconvolution. Then we also rescale each feature map to the spatial size of $H_2 \times W_2$.

3.2. Query Definition

The definition of our queries derives from the object query in Sparse R-CNN [35], but we specify the action query in a disentangled fashion. Specifically, we factorize

action queries into spatial queries $Q_s \in \mathbb{R}^{N\times D}$ and temporal queries $Q_t \in \mathbb{R}^{N\times D}$ to refer to the spatial content and temporal content respectively. N represents the number of queries, while D denotes the dimension of each query.

We further define positional queries $Q_p \in \mathbb{R}^{N\times 4}$. Each positional query Q_p^n (n stands for the query index) is formulated as a proposal box vector (x^n, y^n, z^n, r^n). Formally, x^n and y^n stand for the x- and y-axis coordinates of the box center, and z^n and r^n denote the logarithm of its scale (*i.e.* the area of the bounding box) and aspect ratio. The positional queries Q_p are initialized in such a way that every box vector is the whole keyframe.

3.3. Adaptive Spatiotemporal Feature Sampling

Different from previous work [6, 11, 29, 36] that samples RoI features by pre-computed proposal boxes, we sample actor-specific features adaptively from the aforementioned 4D feature space under the guidance of spatial queries. Specifically, given a spatial query Q_s^n and the corresponding positional query Q_p^n, we regard the center point (x^n, y^n, z^n) of the proposal box as the reference point, and regress P_{in} groups of offsets along x-, y-axis and scale index z on query Q_s^n, using a linear layer:

$$\begin{aligned}\{(\triangle x_i^n, \triangle y_i^n, \triangle z_i^n)\} &= \text{Linear}(Q_s^n),\\ \text{where } i \in \mathbb{Z} &\text{ and } 1 \leqslant i \leqslant P_{in},\end{aligned} \tag{1}$$

where P_{in} is the number of sampled points for each query. Then, the offsets are added to the reference point, thus P_{in} spatial feature points are obtained:

$$\begin{cases}\widetilde{x}_i^n = x^n + \triangle x_i^n \cdot 2^{z^n - r^n},\\ \widetilde{y}_i^n = y^n + \triangle y_i^n \cdot 2^{z^n + r^n},\\ \widetilde{z}_i^n = z^n + \triangle z_i^n.\end{cases} \tag{2}$$

where $2^{z^n - r^n}$ and $2^{z^n + r^n}$ are the width and height of the box respectively. We offset the spatial position of sampling points with respect to the width and height of the box to reduce the learning difficulty.

Finally, we propagate sampling points along the temporal axis, thus obtaining $T \times P_{in}$ points to sample from the 4D feature space. In our implementation, we simply copy these spatial sampling points along the temporal dimension because current action detection datasets yield *temporal slowness property* that the variation of actor locations along the temporal dimension is very slow. We compare different ways of temporal propagating in our ablation study.

Given $T \times P_{in}$ sampling points, we sample instance-specific features by interpolation from the 4D feature space. In the following sections, the sampled spatiotemporal feature for spatial query Q_s^n is denoted by $F^n \in \mathbb{R}^{T\times P_{in}\times D}$.

3.4. Adaptive Dual-branch Feature Mixing

After feature sampling, we factorize the sampled features into spatial features and temporal features by pooling and then enhance each by adaptive mixing respectively. As the dual-branch mixing module is completely symmetrical, we only describe spatial mixing in detail and temporal mixing is performed in a homogeneous way.

Different from MLP-Mixer [37], our mixing parameters are generated adaptively. Given a spatial query $Q_s^n \in \mathbb{R}^D$ and its corresponding sampled features $F^n \in \mathbb{R}^{T\times P_{in}\times D}$, we first use a linear layer to generate query-specific channel-mixing weights $M_c \in \mathbb{R}^{D\times D}$, and then apply plain matrix multiplication on temporally pooled feature and M_c to perform channel-mixing, given by:

$$M_c = \text{Linear}(Q_s^n) \in \mathbb{R}^{D\times D}, \tag{3}$$

$$\text{CM}(F^n) = \text{ReLU}(\text{LayerNorm}(\text{GAP}(F^n) \times M_c)). \tag{4}$$

where GAP stands for the global average pooling operation in the temporal dimension while LayerNorm for the layer normalization [2]. We use $\text{CM}(F^n) \in \mathbb{R}^{P_{in}\times D}$ to denote the channel-wise mixing output features.

After channel-mixing, we perform point-wise mixing in a similar way. Suppose P_{out} is the number of spatial point-wise mixing out patterns, we use $\text{PCM}(F^n) \in \mathbb{R}^{D\times P_{out}}$ to denote the point-wise mixing output features, given by:

$$M_p = \text{Linear}(Q_s^n) \in \mathbb{R}^{P_{in}\times P_{out}}, \tag{5}$$

$$\text{PCM}(F^n) = \text{ReLU}(\text{LayerNorm}(\text{CM}(F^n)^T \times M_p)). \tag{6}$$

The final output $\text{PCM}(F^n)$ is flattened, transformed to D dimension, and added to the spatial query Q_s^n. The temporal query Q_t^n is updated in a homogeneous way, except that the pooling is applied in the spatial dimension. After global spatial pooling, there are T feature points of different temporal locations for temporal mixing and we set the number of temporal point-wise mixing out patterns to T_{out}.

3.5. Sparse Action Decoder

STMixer adopts a unified action decoder for both actor localization and action classification. The decoder comprises M stacked ASAM modules followed by a feed-forward network (FFN) for human scores prediction and a short-term or long-term classifier for action score prediction. In this section, we represent the structure of ASAM module and specify the outputs of the prediction heads.

ASAM module. The overall structure of ASAM module is shown in Figure 2. We first perform self-attention [42] on spatial queries Q_s and temporal queries Q_t to capture the relation information between different instances. Then we perform adaptive sampling from the 4D feature space and dual-branch adaptive mixing on the sampled feature as described before. The spatial queries Q_s and temporal queries

Q_t are updated with mixed features. An FFN is applied on updated spatial queries Q'_s to update the positional queries Q_p. The updated queries Q'_p, Q'_s and Q'_t are used as inputs for the next ASAM module.

Outputs of the prediction heads. The output human boxes are decoded from positional queries Q'_p. We apply an FFN on Q'_s to predict human scores $S_H \in \mathbb{R}^{N\times 2}$ which indicates the confidence values that each box belongs to the human category and background. Based on the concatenation of spatial queries Q'_s and temporal queries Q'_t, we use a short-term or long-term classifier to predict action scores $S_A \in \mathbb{R}^{N\times C}$, where C is the number of action classes.

Short-term and long-term classifier. Short-term classifier (see Figure 2 right top) is a simple FFN which predicts action scores based on short-term information of current queries while long-term classifier (see Figure 2 right bottom) refers to long-term query bank for long-term information. Our design of the long-term classifier is adapted from LFB [47]. We train an STMixer model without long-term information first. Then, for a video of $\mathcal{T}$ clips, we use the trained STMixer to perform inference on each clip of it. We store the concatenation of the spatial and temporal queries from the last ASAM module corresponding to the k highest human scores in the query bank for each clip. We denote the stored queries for clip of time-step t as $L_t \in \mathbb{R}^{k\times d}$ where $d = 2D$, and the long-term query bank of the video as $L = [L_0, L_1, ..., L_{\mathcal{T}-1}]$. Given the long-term query bank of all videos, we train an STMixer model with a long-term classifier from scratch. For video clip t, we first sample a window of length w from the long-term query bank centered at it and stack this window into $\tilde{L}_t \in \mathbb{R}^{K\times d}$:

$$\tilde{L}_t = \text{stack}([L_{t-w/2}, ..., L_{t+w/2-1}]). \tag{7}$$

We then infer current queries to $\tilde{L}_t$ for long-term information by cross-attention [42]:

$$S'_t = \text{cross-attention}(S_t, \tilde{L}_t), \tag{8}$$

where $S_t \in \mathbb{R}^{N\times d}$ is the concatenation of the output spatial queries and temporal queries of the last ASAM module. The output S'_t is then channel-wise concatenated with S_t for action scores prediction.

3.6. Training

We compute training loss based on the output human boxes, human scores, and action scores. Consistent with WOO [6], the loss function $\mathcal{L}$ consists of set prediction loss $\mathcal{L}_{set}$ [4,35,54] and action classification loss $\mathcal{L}_{act}$. Formally,

$$\mathcal{L}_{set} = \lambda_{cls}\mathcal{L}_{cls} + \lambda_{L_1}\mathcal{L}_{L_1} + \lambda_{giou}\mathcal{L}_{giou}, \tag{9}$$

$$\mathcal{L} = \mathcal{L}_{set} + \lambda_{act}\mathcal{L}_{act}. \tag{10}$$

$\mathcal{L}_{cls}$ denotes the cross-entropy loss over two classes (human and background). $\mathcal{L}_{L_1}$ and $\mathcal{L}_{giou}$ are box loss inherited from [4, 35, 54]. As in [4, 6], we first use Hungarian algorithm to find an optimal bipartite matching between predicted actors and ground truth actors according to $\mathcal{L}_{set}$. Then we calculate full training loss $\mathcal{L}$ based on the matching results. $\mathcal{L}_{act}$ is binary cross entropy loss for action classification. We only compute $\mathcal{L}_{act}$ for the prediction matched with a ground truth. λ_{cls}, λ_{L_1}, λ_{giou}, and λ_{act} are corresponding weights of each term.

4. Experiments

4.1. Experimental Setup

Datasets. The AVA dataset [16] contains 211k video clips for training and 57k for validation segmented from 430 15-minute videos. The videos are annotated at 1FPS over 80 atomic action classes. Following the standard evaluation protocol [16], we report our results on 60 action classes that have at least 25 validation examples. JHMDB [19] consists of 928 temporally trimmed videos from 21 action classes. Results averaged over three splits are reported. UCF101-24 is a subset of UCF101 [33]. It contains 3,207 videos annotated with action instances of 24 classes. As the common setting, we report the performance on the first split.

Network configurations. We configure the dimension D of both spatial and temporal queries to 256 and set the number of both queries N equaling 100. The number of sampling point P_{in} in each temporal frame is set to 32. The spatial and temporal mixing out patterns P_{out} and T_{out} are set to 4 times the number of sampling points and temporal frames respectively, that is, 128 and 32 for SlowFast [11] backbone and 128 and 16 for CSN [40] and ViT [7] backbone. Following multi-head attention [42] and group convolution [49], we split the channel D into 4 groups and perform group-wise sampling and mixing. We stack 6 ASAM modules in our action decoder as default. For the long-term classifier, we set the number of stored queries of each clip k as 5 and window length w as 60. The number of cross-attention layers is set to 3.

Losses and optimizers. We set the loss weight in STMixer as $\lambda_{cls} = 2.0$, $\lambda_{L_1} = 2.0$, $\lambda_{giou} = 2.0$ and $\lambda_{act} = 24.0$. We use AdamW [28] optimizer with weight decay 1×10^{-4} for all experiments. Following [4, 6], intermediate supervision is applied after each ASAM module.

Training and inference recipes. We train STMixer detectors for 10 epochs with an initial learning rate of 2.0×10^{-5} and batchsize of 16. The learning rate and batchsize can be tuned according to the linear scaling rule [15]. We randomly scale the short size of the training video clips to 256 or 320 pixels. Color jittering and random horizontal flipping are also adopted for data augmentation.

For inference, we scale the short size of input frames

to 256 as the common setting. Given an input video clip, STMixer predicts N bounding boxes associated with their human scores and action scores from the last ASAM module. If the confidence score that a box belongs to the human category is higher than the preset threshold, we take it as a detection result. We set the threshold to 0.6 for AVA. The performances are evaluated with official metric frame-level mean average precision(mAP) at 0.5 IoU threshold.

4.2. Ablation Study

We conduct ablation experiments on AVA v2.2 dataset to investigate the influence of different components in our STMixer framework. A SlowOnly ResNet-50 backbone [8] is used for our ablation experiments. We report both mAP and GFLOPs for effectiveness and efficiency comparison.

Ablations on 4D feature space. We first show the benefit of sampling features from the unified 4D feature space. For comparison, we design a two-stage counterpart of our STMixer. In the first stage, we sample features from keyframe features for actor localization. In the second stage, we sample features from res5 features for action classification. As shown in Tabel 1a, sampling features from a unified 4D feature space and performing action detection in a one-stage manner significantly reduce computing costs, and detection mAP also gets improved as multi-scale information is also utilized for classification. We then investigate two different ways for the 4D feature space construction. As shown in Table 1b, constructing 4D feature space by simple lateral $1 \times 1 \times 1$ convolution achieves comparable detection accuracy while being more efficient than using full FPN [25] with a top-down pathway.

Ablations on feature sampling. In Tabel 1c, we compare 3 different feature sampling strategies. For fixed grid feature sampling, we sample 7×7 feature points inside each actor box by interpolation, which is actually the RoIAlign [17] operation adopted by many former methods [6, 13]. Though sampling fewer points per group, our adaptive sampling strategy improves the detection mAP by 1.2. The results show that the fixed RoIAlign operation for feature sampling fails to capture useful context information outside the box while our adaptive sampling strategy enables the detector to mine discriminative features from the whole 4D feature space. Beyond simply copying sampling points, we try sampling different feature points in different frames by predicting the offset of the reference bounding box for each frame. The improvement is marginal due to the slowness issue of current action detection datasets that the variation of actors' location and action is very slow. In Table 1d, we further investigate the influence of the number of sampling points per group P_{in} in each temporal frame. Setting $P_{in} = 32$ achieves the best detection performance.

Ablations on feature mixing. In Table 1f, we compare different feature mixing strategies. We first demonstrate the benefit of query-guided adaptive mixing. The detection mAP drops by 0.6 when using fixed parameter mixing. For adaptive mixing, the mixing parameters are dynamically generated based on a specific query, thus more powerful to enhance the presentation of each specific action instance proposal. We further compare different adaptive feature mixing strategies. From the results in Table 1f, it is demonstrated that both spatial appearance and temporal motion information are important for action detection. However, coupled spatiotemporal feature mixing using queries of a single type has a high computational complexity. Decoupling features along spatial and temporal dimensions saves computing costs. Our dual-branch spatiotemporal feature mixing outperforms sequential spatiotemporal feature mixing by 0.5 mAP. This is because actor localization at keyframes only needs spatial appearance information and parallel dual-branch mixing will reduce the effect of temporal information on localization. Also, by concatenating temporal queries to spatial queries, more temporal information is leveraged for action classification. In Table 1g, we investigate different spatial and temporal mixing out patterns P_{out} and T_{out} from 64 and 8 to 192 and 24, that is, 2 times to 6 times the number of sampling points and temporal frames. Setting P_{out} and T_{out} equaling 128 and 16 achieves the best performance.

Ablations on network configuration. As shown in Table 1e, the detection mAP is saturated when the number of queries is increased to 100. From the results in Table 1h, a stack of 6 ASAM modules achieves a good effectiveness and efficiency balance.

4.3. Comparison with State-of-the-arts on AVA

We compare our proposed STMixer with state-of-the-art methods on AVA v2.1 and v2.2 in Table 2. We first compare our STMixer to methods using an extra offline human detector. Our STMixer with SlowFast-R101 backbone achieves 30.6 and 30.9 mAP when not using long-term features. With long-term query support, our STMixer reaches 32.6 and 32.9 mAP on AVA v2.1 and v2.2 respectively. To demonstrate the generalization ability of our method, we conduct experiments with ViT [7] backbone. Compared with the two-stage counterparts, STMixer brings performance improvements while getting rid of the dependence on an extra detector. Although ViT is considered to have a global receptive field, our adaptive sampling and decoding mechanism could serve as a supplement to improve the flexibility of the model. Compared to previous end-to-end methods, our STMixer achieves the best results. Our STMixer outperforms WOO [6] by 2.0 and 1.7 mAP even though WOO test models at 320 resolution. STMixer also consistently outperforms TubeR [53] on AVA v2.1 or v2.2, using or not using long-term features.

We compare mAP versus GFLOPs on AVA v2.2 in Fig-

Classification	Localization	mAP	GFLOPs
4D Feature Space		23.1	44.4
Key-Frame Features	Res5 Features	22.8	53.7

(a) **Feature space.** Sampling features from 4D feature space is more effective and efficient than sampling from key-frame features for classification and res5 features for localization.

	mAP	GFLOPs
Simple Lateral Conv.	23.1	44.4
Full FPN	22.9	45.8

(b) **4D feature space construction.** Simple lateral convolution achieves comparable performance while being more efficient than using full FPN.

Sampling Strategy	P_{in}	mAP	GFLOPs
Fixed Grid Sampling	49	21.9	45.6
Adaptive Sampling + Temporal Copying	32	23.1	44.4
Adaptive Sampling + Temporal Moving	32	23.3	44.6

(c) **Sampling strategy.** Sampling fewer feature points, our adaptive sampling strategy achieves better performance than fixed grid sampling. Temporal moving brings slight improvement in mAP.

P_{in}	8	16	32	48	64
mAP	22.4	22.5	23.1	23.0	22.5
GFLOPs	42.4	43.1	44.4	45.6	46.9

(d) **Number of sampling points.** Sampling 32 points per frame achieves the best performance.

N	15	50	100	150
mAP	22.1	22.9	23.1	22.5
GFLOPs	31.7	36.9	44.4	51.8

(e) **Number of queries.** Using 100 queries works the best.

Mixing Strategy	mAP	GFLOPs
fixed parameter dual-branch mixing	22.5	36.8
dual-branch spatiotemporal mixing	23.1	44.4
spatial mixing only	22.4	40.4
temporal mixing only	22.1	33.5
sequential spatiotemporal mixing	22.6	43.6
coupled spatiotemperal mixing	22.8	93.2

(f) **Mixing strategy.** Query-guided adaptive feature mixing outperforms fixed parameter mixing and our dual-branch spatiotemporal feature mixing strategy works the best.

P_{out}/T_{out}	64/8	96/12	128/16	160/20	192/24
mAP	22.5	22.8	23.1	22.9	22.6
GFLOPs	40.2	42.3	44.4	46.4	48.4

(g) **Number of mixing out patterns.** A moderate number of mixing out patterns works the best.

M	1	3	6	9
mAP	18.4	22.5	23.1	22.6
GFLOPs	32.0	36.9	44.4	51.8

(h) **Number of ASAM modules.** Using 6 ASAM modules works the best.

Table 1. **Ablations Experiments.** We use a SlowOnly ResNet-50 backbone to perform our ablation studies. Models are trained on the training set of AVA v2.2 and evaluated on the validation set. Default choices for our model are colored in gray.

Method	Detector	One-stage	Input	Backbone	Pre-train	LF	mAP	
							v2.1	v2.2
Compare to methods with an extra human detector								
SlowFast [11]	✓	✗	32×2	SF-R101-NL	K600	✗	28.2	29.0
LFB [47]	✓	✗	32×2	I3D-R101-NL	K400	✓	27.7	-
CA-RCNN [48]	✓	✗	32×2	R50-NL	K400	✓	28.0	-
AIA [36]	✓	✗	32×2	SF-R101	K700	✓	31.2	32.3
ACARN [29]	✓	✗	32×2	SF-R101	K400	✓	30.0	-
ACARN [29]	✓	✗	32×2	SF-R101-NL	K600	✓	-	31.4
VideoMAE [38]	✓	✗	16×4	ViT-B	K400	✗	-	31.8
VideoMAE [38]	✓	✗	16×4	ViT-L	K700	✗	-	39.3
STMixer	✗	✓	32×2	SF-R101	K700	✗	30.6	30.9
STMixer	✗	✓	32×2	SF-R101	K700	✓	32.6	32.9
STMixer	✗	✓	16×4	ViT-B (from [38])	K400	✗	-	32.6
STMixer	✗	✓	16×4	ViT-B (from [43])	K710+K400	✗	-	36.1
STMixer	✗	✓	16×4	ViT-L (from [38])	K700	✗	-	**39.5**
Compare to end-to-end methods								
AVA [16]	✗	✗	20×1	I3D-VGG	K400	✗	14.6	-
ACRN [34]	✗	✗	20×1	S3D-G	K400	✗	17.4	-
STEP [51]	✗	✗	12×1	I3D-VGG	K400	✗	18.6	-
VTr [13]	✗	✗	64×1	I3D-VGG	K400	✗	24.9	-
WOO [6]	✗	✗	32×2	SF-R50	K400	✗	25.2	25.4
WOO [6]	✗	✗	32×2	SF-R101-NL	K600	✗	28.0	28.3
TubeR [53]	✗	✓	32×2	CSN-152	IG-65M	✗	29.7	31.1
TubeR [53]	✗	✓	32×2	CSN-152	IG-65M	✓	31.7	33.4
STMixer	✗	✓	32×2	SF-R50	K400	✗	27.2	27.8
STMixer	✗	✓	32×2	SF-R101-NL	K600	✗	29.8	30.1
STMixer	✗	✓	32×2	CSN-152	IG-65M	✗	31.7	32.8
STMixer	✗	✓	32×2	CSN-152	IG-65M	✓	**34.4**	34.8

Table 2. **Comparisons with state-of-the-arts on validation sets of AVA v2.1 and v2.2.** ✓of column "Detector" denotes an extra human detector Faster RCNN-R101-FPN [31] is used. ✓of column "LF" denotes long-term features are used.

ure 1 to show the efficiency of our STMixer. AIA [36] and ACARN [29] do not report their GFLOPs. As they are built on SlowFast [11] framework and also use an offline human detector but introduce extra modules to model interaction, SlowFast can serve as a lower bound of complexity for them. For a fair comparison, we report results for no long-term feature version of each method. As shown in Figure 1, due to an extra human detector being needed, SlowFast [11], CSN [40], and VideoMAE [38] have much higher GFLOPs than end-to-end methods with same backbone. Among end-to-end methods, STMixer achieves the best effectiveness and efficiency balance. STMixer outperforms WOO [6] by 1.8 mAP while having much lower GFLOPs (135 versus 252). With a slight GFLOPs increase (126 versus 120), STMixer outperforms TubeR [53] by 1.7 mAP.

4.4. Results on JHMDB and UCF101-24

To verify the effectiveness of our STMixer, we further evaluate it on the JHMDB [19] and UCF101-24 [33]

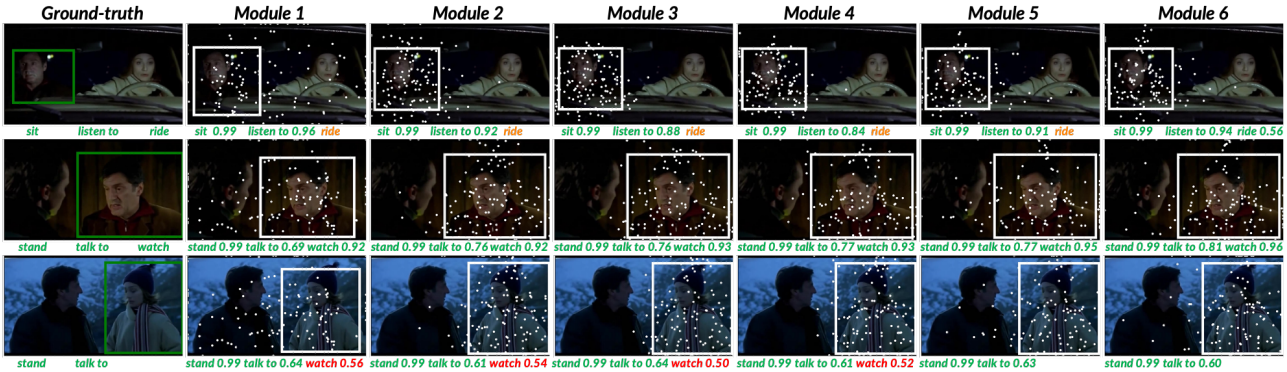

Figure 4. **Sampling points and detection results visualization.** We display the actor bounding boxes and action classes of three ground-truth action instances in the first column. Each ASAM module's sampling points, predicted actor bounding boxes and action scores are displayed in the following columns. The correctly predicted action classes are displayed in green, missing in orange, and wrongly predicted in red. Intuitively, STMixer mines discriminative context features outside the bounding box for better action detection.

Method	Detector	Input	Backbone	JHMDB	UCF101-24
MOC* [24]	✓	7×1	DLA34	70.8	78.0
AVA* [16]	✗	20×1	I3D-VGG	73.3	76.3
ACRN [34]	✗	20×1	S3D-G	77.9	-
CA-RCNN [48]	✓	32×2	R50-NL	79.2	-
YOWO [21]	✗	16×1	3DResNext-101	80.4	75.7
WOO [6]	✗	32×2	SF-R101-NL	80.5	-
AIA [36]	✓	32×1	SF-R50-NL	-	78.8
ACARN [29]	✓	32×1	SF-R50	-	**84.3**
TubeR* [53]	✗	32×2	I3D	-	81.3
STMixer	✗	32×2	SF-R101-NL	**86.7**	83.7

Table 3. Comparison on JHMDB and UCF101-24. Methods marked with * use extra optical flow features.

datasets. We report the frame-level mean average precision (frame-mAP) with an intersection-over-union (IoU) threshold of 0.5 in Table 3. Experimental results demonstrate that STMixer outperforms the current state-of-the-art methods with remarkable performance gain on JHMDB and achieves competitive results on UCF101-24.

4.5. Visualization

We provide the visualization of sampling points and detection results of each ASAM module in order in Figure 4. In the first few ASAM modules, the sampling points are quickly concentrated on the action performer and the actor localization accuracy gets improved rapidly. In the following ASAM modules, some of the sampling points get out of the human box and spread to the context. Benefiting from the gathered context and interaction information, the action recognition accuracy gets improved while localization accuracy is not compromised. In Figure 4, we show clear improvements in three aspects: predicting missing classes (ride in row 1), improving the confidence score of correctly predicted classes (talk to in row 2), and removing wrongly predicted classes (watch in row 3). To recognize an action "ride", we need to observe the person is in a car and someone is driving. For recognition of "talk to", we need to know if a person is listening in the context, and for "watch", we need to know if a person is in the actor's sight. The improvements are mostly related to these classes of interaction, which indicates our STMixer is capable of mining discriminative interaction information from the context.

5. Conclusion and Future Work

In this paper, we have presented a new one-stage sparse action detector in videos, termed STMixer. Our STMixer yields a simple, neat, and efficient end-to-end action detection framework. The core design of our STMixer is a set of learnable queries to decode all action instances in parallel. The decoding process is composed of an adaptive feature sampling module to identify important features from the entire spatiotemporal domain of video, and an adaptive feature mixing module to dynamically extract discriminative representations for action detection. Our STMixer achieves a new state-of-the-art performance on three challenging benchmarks of AVA, JHMDB, and UCF101-24 with less computational cost than previous end-to-end methods. We hope STMixer can serve as a strong baseline for future research on video action detectors.

One limitation of our STMixer is that the long-term query bank is implemented in an offline way where another STMixer without long-term query support is pre-trained for long-term query generation. We leave the design of an online query bank to future research and hope our STMixer is extended to extract long-form video information in an end-to-end manner.

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. 62072232), Fundamental Research Funds for the Central Universities (No. 020214380091), and Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Luči'c, and Cordelia Schmid. Vivit: A video vision transformer. In *ICCV*, pages 6836–6846, 2021. 1

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[3] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 813–824. PMLR, 2021. 1

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229. Springer, 2020. 1, 2, 5

[5] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, pages 6299–6308, 2017. 1

[6] Shoufa Chen, Peize Sun, Enze Xie, Chongjian Ge, Jiannan Wu, Lan Ma, Jiajun Shen, and Ping Luo. Watch only once: An end-to-end video action detection framework. In *ICCV*, pages 8178–8187, 2021. 2, 4, 5, 6, 7, 8

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*. OpenReview.net, 2021. 3, 5, 6

[8] Haoqi Fan, Yanghao Li, Bo Xiong, Wan-Yen Lo, and Christoph Feichtenhofer. Pyslowfast. https://github.com/facebookresearch/slowfast, 2020. 6

[9] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, pages 6824–6835, 2021. 1, 2

[10] Gueter Josmy Faure, Min-Hung Chen, and Shang-Hong Lai. Holistic interaction transformer network for action detection. In *WACV*, pages 3329–3339. IEEE, 2023. 2

[11] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, pages 6202–6211, 2019. 1, 2, 4, 5, 7

[12] Ziteng Gao, Limin Wang, Bing Han, and Sheng Guo. Adamixer: A fast-converging query-based object detector. In *CVPR*, pages 5354–5363. IEEE, 2022. 2

[13] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *CVPR*, pages 244–253, 2019. 2, 6, 7

[14] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *CVPR*, pages 759–768, 2015. 1

[15] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017. 5

[16] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, pages 6047–6056, 2018. 1, 2, 5, 7, 8

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 1, 2, 6

[18] Roei Herzig, Elad Levi, Huijuan Xu, Hang Gao, Eli Brosh, Xiaolong Wang, Amir Globerson, and Trevor Darrell. Spatio-temporal action graph networks. In *ICCVW*, pages 0–0, 2019. 1

[19] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J. Black. Towards understanding action recognition. In *ICCV*, pages 3192–3199. IEEE Computer Society, 2013. 2, 5, 7

[20] Jianwen Jiang, Yu Cao, Lin Song, Shiwei Zhang, Yunkai Li, Z Xu, Q Wu, C Gan, C Zhang, and G Yu. Human centric spatio-temporal action localization. In *CVPRW*, 2018. 1

[21] Okan Köpüklü, Xiangyu Wei, and Gerhard Rigoll. You only watch once: A unified cnn architecture for real-time spatiotemporal action localization. *arXiv preprint arXiv:1911.06644*, 2019. 2, 8

[22] Yixuan Li, Lei Chen, Runyu He, Zhenzhi Wang, Gangshan Wu, and Limin Wang. Multisports: A multi-person video dataset of spatio-temporally localized sports actions. In *ICCV*, pages 13516–13525. IEEE, 2021. 1

[23] Yanghao Li, Hanzi Mao, Ross B. Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. *CoRR*, abs/2203.16527, 2022. 3

[24] Yixuan Li, Zixu Wang, Limin Wang, and Gangshan Wu. Actions as moving points. In *ECCV*, pages 68–84. Springer, 2020. 2, 8

[25] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 6

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014. 2

[27] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *CVPR*, pages 3192–3201. IEEE, 2022. 1

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[29] Junting Pan, Siyu Chen, Mike Zheng Shou, Yu Liu, Jing Shao, and Hongsheng Li. Actor-context-actor relation network for spatio-temporal action localization. In *CVPR*, pages 464–474, 2021. 1, 2, 4, 7, 8

[30] Xiaojiang Peng and Cordelia Schmid. Multi-region two-stream r-cnn for action detection. In *ECCV*, pages 744–759. Springer, 2016. 1

[31] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 1, 2, 7

[32] Gurkirt Singh, Suman Saha, Michael Sapienza, Philip HS Torr, and Fabio Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *ICCV*, pages 3637–3646, 2017. 1

[33] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. 2, 5, 7

[34] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *ECCV*, pages 318–334, 2018. 2, 7, 8

[35] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *CVPR*, pages 14454–14463, 2021. 1, 3, 5

[36] Jiajun Tang, Jin Xia, Xinzhi Mu, Bo Pang, and Cewu Lu. Asynchronous interaction aggregation for action detection. In *ECCV*, pages 71–87. Springer, 2020. 1, 2, 4, 7, 8

[37] Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. In *NeurIPS*, pages 24261–24272, 2021. 4

[38] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *NeurIPS*, 2022. 1, 2, 7

[39] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, pages 4489–4497, 2015. 1

[40] Du Tran, Heng Wang, Matt Feiszli, and Lorenzo Torresani. Video classification with channel-separated convolutional networks. In *ICCV*, pages 5551–5560. IEEE, 2019. 1, 5, 7

[41] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, pages 6450–6459. Computer Vision Foundation / IEEE Computer Society, 2018. 1

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017. 4, 5

[43] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yihan He, Yi Wang, Yali Wang, and Yu Qiao. VideoMAE V2: Scaling video masked autoencoders with dual masking. In *CVPR*, 2023. 1, 7

[44] Limin Wang, Yu Qiao, Xiaoou Tang, and Luc Van Gool. Actionness estimation using hybrid fully convolutional networks. In *CVPR*, pages 2708–2717, 2016. 1

[45] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, pages 7794–7803, 2018. 1

[46] Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Learning to track for spatio-temporal action localization. In *ICCV*, pages 3164–3172, 2015. 1

[47] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krahenbuhl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *CVPR*, pages 284–293, 2019. 2, 5, 7

[48] Jianchao Wu, Zhanghui Kuang, Limin Wang, Wayne Zhang, and Gangshan Wu. Context-aware rcnn: A baseline for action detection in videos. In *ECCV*, pages 440–456. Springer, 2020. 7, 8

[49] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 5987–5995. IEEE Computer Society, 2017. 5

[50] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, pages 305–321, 2018. 1

[51] Xitong Yang, Xiaodong Yang, Ming-Yu Liu, Fanyi Xiao, Larry S Davis, and Jan Kautz. Step: Spatio-temporal progressive learning for video action detection. In *CVPR*, pages 264–272, 2019. 7

[52] Yanyi Zhang, Xinyu Li, Chunhui Liu, Bing Shuai, Yi Zhu, Biagio Brattoli, Hao Chen, Ivan Marsic, and Joseph Tighe. Vidtr: Video transformer without convolutions. In *ICCV*, pages 13577–13587, 2021. 1

[53] Jiaojiao Zhao, Yanyi Zhang, Xinyu Li, Hao Chen, Bing Shuai, Mingze Xu, Chunhui Liu, Kaustav Kundu, Yuanjun Xiong, Davide Modolo, Ivan Marsic, Cees G. M. Snoek, and Joseph Tighe. Tuber: Tubelet transformer for video action detection. In *CVPR*, pages 13588–13597. IEEE, 2022. 1, 2, 6, 7, 8

[54] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. In *ICLR*. OpenReview.net, 2021. 1, 5