Video Self-Stitching Graph Network for Temporal Action Localization

Chen Zhao Ali Thabet Bernard Ghanem
King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia
{chen.zhao, ali.thabet, bernard.ghanem}@kaust.edu.sa

Abstract

Temporal action localization (TAL) in videos is a challenging task, especially due to the large variation in action temporal scales. Short actions usually occupy a major proportion in the datasets, but tend to have the lowest performance. In this paper, we confront the challenge of short actions and propose a multi-level cross-scale solution dubbed as video self-stitching graph network (VSGN). We have two key components in VSGN: video self-stitching (VSS) and cross-scale graph pyramid network (xGPN). In VSS, we focus on a short period of a video and magnify it along the temporal dimension to obtain a larger scale. We stitch the original clip and its magnified counterpart in one input sequence to take advantage of the complementary properties of both scales. The xGPN component further exploits the cross-scale correlations by a pyramid of cross-scale graph networks, each containing a hybrid module to aggregate features from across scales as well as within the same scale. Our VSGN not only enhances the feature representations, but also generates more positive anchors for short actions and more short training samples. Experiments demonstrate that VSGN obviously improves the localization performance of short actions as well as achieving the state-of-the-art overall performance on THUMOS-14 and ActivityNet-v1.3. VSGN code is available at https://github.com/coolbay/VSGN.

1. Introduction

Nowadays has seen a growing interest in video understanding both from industry and academia, owing to the rapidly produced video content on the Internet. Temporal action localization (TAL) in untrimmed videos is one important task in this area, which aims to specify the start and the end time of an action as well as to identify its category. TAL is not only the key technique of various application such as extracting highlights in sports, but also lays the foundation for other higher-level tasks such as video grounding [10, 13] and video captioning [17, 27].

Though many methods (e.g., [1, 2, 8, 19, 20, 23, 40, 41, 42, 44]) in recent years have been continuously breaking the record of TAL performance, a major challenge hinders its substantial improvement – large variation in action duration. An action can last from a fraction of a second to minutes in the real-world scenario as well as in the datasets [6, 14]. We plot the distribution of action duration in the dataset ActivityNet-v1.3 [6] in Fig. 1 a). We notice that actions shorter than 30 seconds dominate the distribution, but their performance is obviously inferior to longer ones with all different TAL methods (Fig. 1 b)). Therefore, the accuracy of short actions is a key factor to determine the performance of a TAL method.

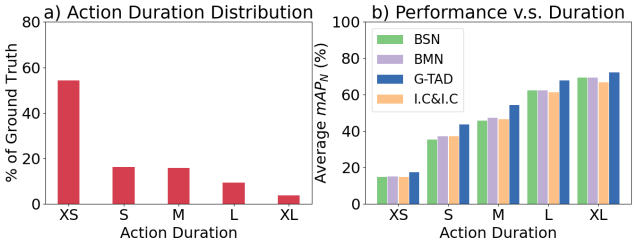

Figure 1. **Short actions are the majority in numbers, but have the lowest performance.** a) Distribution of action duration in ActivityNet-v1.3 [6]. Actions are divided into five duration groups (in seconds): XS (0, 30], S (30, 60], M (60, 120], L (120, 180], and XL (180, inf). b) TAL Performance of different methods on actions of different duration.

Why are short actions hard to localize? Short actions have *small temporal scales* with fewer frames, and therefore, their information is prone to loss or distortion throughout a deep neural network. Most methods in the literature process videos regardless of action duration, which as a consequence sacrifices the performance of short actions. Recently, researchers attempt to incorporate feature pyramid networks (FPN) [21] from the object detection problem to the TAL problem [23, 26], which generates different feature scales at different network levels, each level with different sizes for candidate actions. Though by this means short actions may go through fewer pooling layers to avoid being excessively down-scaled, yet their original small scale

as the source of the problem still limits the performance.

Then how can we attack the small-scale problem of short actions? A possible solution is to temporally up-scale videos to obtain more frames to represent an action. Recent literature shows the practice of re-scaling videos via linear interpolation before feeding into a network [2, 19, 20, 42, 46], but these methods actually *down*-scale rather than *up*-scale videos (e.g., using only 100 snippets on AcitivityNet-v1.3). Even if we can adapt a method to using a larger-scale input, how can we ensure that the up-scaled videos contain sufficient and accurate information for detecting an action? Moreover, it makes the problem even harder that re-scaling is usually not performed on original frames, but on video features which do not satisfy linearity.

Up-scaling a video could transform a short action into a long one, but may lose important information for localization. Thus both the original scale and the enlarged scale have their limitations and advantages. The original video scale contains the original intact information, while the enlarged one is easier for the network to detect. In contrast to other works that either use the original-scale video or a down-scaled video, in this paper, we use both to take advantage of their complementary properties and mutually enhance their feature representations.

Specifically, we propose a **V**ideo self-**S**titching **G**raph **N**etwork (VSGN) for improving performance of short actions in the TAL problem. Our VSGN is a multi-level cross-scale framework that contains two major components: video self-stitching (VSS); cross-scale graph pyramid network (xGPN). In VSS, we focus on a short period of a video and magnify it along the temporal dimension to obtain a larger scale. Then using our self-stitching strategy, we piece together both the original-scale clip and its magnified counterpart into one single sequence as the network input. In xGPN, we progressively aggregate features from cross scales as well as from the same scale via a pyramid of cross-scale graph networks. Hence, we enable direct information pass between the two feature scales. Compared to simply using one scale, our VSGN adaptively rectifies distorted features in either scales from one another by learning to localize actions, therefore, it is able to retain more information for the localization task. In addition to enhancing the features, our VSGN augments the datasets with more short actions to mitigate the bias towards long actions during the learning process, and enables more anchors, even those with large scales, to predict short actions.

We summarize our contributions as follows:

1) To the best of our knowledge, this is the first work that sheds light on the problem of short actions for the task of temporal action localization. We propose a novel solution that utilizes *cross*-scale correlations of *multi*-level features to strengthen their representations and facilitate localization.

2) We propose a novel temporal action localization framework VSGN, which features two key components: video self-stitching (VSS); cross-scale graph pyramid network (xGPN). For effective feature aggregation, we design a cross-scale graph network for each level in xGPN with a hybrid module of a temporal branch and a graph branch.

3) VSGN shows obvious improvement on short actions over other concurrent methods, and also achieves new state-of-the-art overall performance. On THUMOS-14, VSGN reaches 52.4% mAP@0.5, compared to previous best score 40.4% under the same features. On ActivityNet-v1.3, VSGN reaches an average mAP of 35.07%, compared to the previous best score 34.26% under the same features.

2. Related Work

2.1. Multi-scale solution in object detection

Temporal action localization is analogous to the task of object detection in images, though the scale variation in images is not as large as in videos. Multiple methods have been proposed to deal with small objects specifically [3, 28] or object scale variation in general [4, 21] in images.

A representative work for object scale invariance is the feature pyramid network (FPN) [21], which generates multi-scale features using an architecture of encoder and decoder pyramids. FPN has become a popular base architecture for many object detection methods in recent years (e.g., [29, 34, 35, 45]). Following FPN, some methods are proposed to further improve the architecture for higher efficiency and better accuracy, such as PANet [24], NAS-FPN [11], BiFPN [33]. Our proposed cross-scale graph pyramid (xGPN) adopts the idea of FPN and builds a pyramid of *video* features in the *temporal* domain instead of *images* in the *spatial* domain. Moreover, we embed *cross-scale graph networks* in the pyramid levels.

Another perspective to address the scale issue, especially for the small scale, is data augmentation, e.g. mosaic augmentation in YOLOv4 [4], which pieces together four images into one large image and crop a center area for training. It helps the model learn to not overemphasize the activations for large objects so as to enhance the performance for small objects. Our VSGN is inspired by mosaic augmentation, but it stitches *the same video clip of different scales* along the temporal dimension rather than different videos.

2.2. Temporal action localization

Recent temporal action localization methods can be generally classified into two categories based on the way they deal with the input sequence. In the first category, the works such as BSN [20], BMN [19], G-TAD [42], BC-GNN [2] re-scale each video to a fixed temporal length (usually a small length such as 100 snippets) regardless of the original video duration. Methods using this strategy are effi-

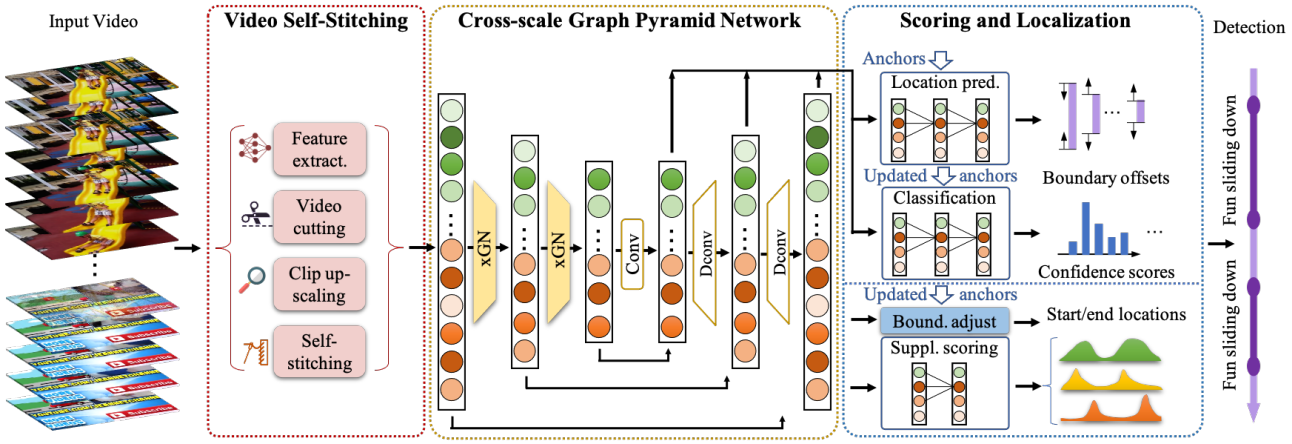

Figure 2. **Architecture of the proposed video self-stitching graph network (VSGN)**. Its takes a video sequence and generates detected actions with start/end time as well as their categories. It has three components: **video self-stitching (VSS)**, **cross-scale graph pyramid network (xGPN)**, and **scoring and localization (SoL)**. **VSS** (red dashed box, see Fig. 3 for details) contains four steps to prepare a video sequence as xGPN input. **xGPN** is composed of multi-level encoder and decoder pyramids. The encoder aggregates features in different levels via a stack of cross-scale graph networks (xGN) (yellow trapezoid area, see Fig. 4 for details); the decoder restores the temporal resolution and generates multi-level features for detection. **SoL** (blue dashed box) contains four modules, the top two predicting action scores and boundaries, the bottom two producing supplementary scores and adjusting boundaries.

cient owing to the small input scale, but would harm short actions especially those in long videos, since these short actions are essentially down-scaled and their information easily gets lost or distorted. However, it is non-trivial to *up*-scale videos as input instead for these methods limited by their architectures. For example, BSN relies on the startness/endness curves to identify proposal candidates, but when more frames are used, the curves will have too many peaks and valleys to generate meaningful proposals. In G-TAD, if too many snippets are interpolated and neighboring snippets become similar, it tends to find graph neighbors only in the temporal vicinity (referred to as *scaling curse*).

The second category is to use sliding windows to crop the original video into multiple input sequences. This can preserve the original information of each frame. The works R-C3D [40], TAL-NET [8], PBRNet [23], belonging to this category, perform pooling / strided convolution to obtain multi-scale features. Compared to these two categories, our proposed VSGN uses both the original video clip and its up-scaled counterpart, and takes advantage of their complementary properties to enhance their representations.

2.3. Graph neural networks for TAL

Graph neural networks (GNN) are a useful model for exploiting correlations in irregular structures [16]. As they become popular in different computer vision fields [12, 36, 38], researchers also find their application in temporal action localization [2, 42, 44]. G-TAD [42] breaks the restriction of temporal locations of video snippets and uses a graph to aggregate features from snippets not located in a temporal neighborhood. It models each snippet as a node and snippet-snippet correlations as edges, and applies edge convolutions [36] to aggregate features. BC-GNN [2] improves localization by modelling the boundaries and content of temporal proposals as nodes and edges of a graph neural network. P-GCN [44] considers each proposal as a graph node, which can be combined with a proposal method to generate better detection results.

Compared to these methods, our VSGN builds a graph on video snippets as G-TAD, but differently, beyond modelling snippets from the same scale, VSGN also exploits correlations between *cross-scale* snippets and defines a cross-scale edge to break the *scaling curse*. In addition, our VSGN contains *multiple-level* graph neural networks in a pyramid architecture whereas G-TAD only uses one level.

3. Video Self-Stitching Graph Network

Fig. 2 demonstrates the overall architecture of our proposed **V**ideo self-**S**titching **G**raph **N**etwork (VSGN). It is comprised of three components: video self-stitching (VSS), cross-scale graph pyramid network (xGPN), scoring and localization (SoL), which will be elaborated in Sec. 3.2, 3.3, and 3.4, respectively. Before delving into the details, in Sec. 3.1 we first introduce our ideas behind these components to deal with the problem of short actions.

3.1. VSGN for Short Actions

Larger-scale clip. To solve the problem of short action scales, let us first think about how humans react when they find themselves interested in a short video clip that just

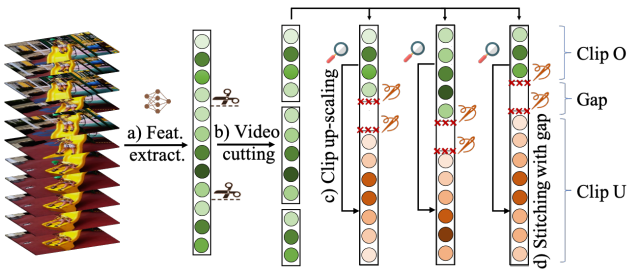

Figure 3. **Video self-stitching (VSS).** a) Snippet-level features are extracted for the entire video. b) Long video is cut into multiple short clips. c) Each video clip is up-scaled along the temporal dimension. d) Original clip (green dots) and up-scaled clip (orange dots) are stitched into one feature sequence with a gap.

fleeted away. They would scroll back to the clip and re-play it with a lower speed, by pause-and-play for example. We mimic this process when preparing a video before feeding it into a neural network. We propose to focus on a short period of a video, and magnify it along the temporal dimension to obtain a video clip of a larger temporal scale (**VSS** in Fig. 2, see Sec. 3.2 for details). A larger temporal scale, is not only able to retain more information through the network aggregation and pooling, but also associated with larger anchors which are easier to detect.

Multi-scale input. The magnification process may inevitably impair the information in the clip, thus the original video clip, which contains the original intact information, is also necessary. To take advantage of the complementary properties of both scales, we design a video stitching technique to piece them together as one single network input (**VSS** in Fig. 2, see Sec. 3.2 for details). This strategy enables the network to process both scales in one single pass, and the clip to have more positive anchors of different scales. It is also an effective way to augment the dataset.

Cross-scale correlations. The original clip and the magnified clip, albeit different, are highly correlated since they contain the same video content. If we can utilize their correlations and draw connections between their features, then the impaired information in the magnified clip can be rectified by the original clip, and the lost information in the original clip during pooling can be restored by the magnified clip. To this end, we propose a cross-scale graph pyramid network (**xGPN** in Fig. 2, see Sec. 3.3 for details), which aggregates features not only from the same scale but from cross scales, and which progressively enhances the features of both scales at multiple network levels.

3.2. Video Self-Stitching

The video self-stitching (VSS) component transforms a video into multi-scale input for the network. As illustrated in Fig. 3, it takes a video sequence, extracts snippet-level features, cuts into multiple short clips if it is long, up-scales each short clip along the temporal dimension, and stitches together each pair of original and up-scaled clips into one sequence. Please note that in addition to using VSS to generate multi-scale input, we also directly use all original long videos as input in order to detect long actions as well.

Feature extraction (Fig. 3 a)). Let us denote a video sequence as $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^{T} \in \mathbb{R}^{W\times H\times T\times 3}$, where $W \times H$ refers to the spatial resolution and T is the total number of frames. We use a feature encoding method (such as TSN [39], I3D [7]) to extract its features on a snippet basis (one snippet is defined as τ consecutive video frames). We generate one feature vector for each snippet and obtain a feature sequence denoted as $\mathbf{F} = \{\mathbf{f}_t\}_{t=1}^{T/\tau} \in \mathbb{R}^{T/\tau\times C}$, where C is the feature dimension.

Video cutting (Fig. 3 b)). Suppose the requirement for our network input is L snippet features[1] $\mathbf{F}^0 = \{\mathbf{f}_t^0\}_{t=1}^{L} \in \mathbb{R}^{L\times C}$. We define a *short clip* as those that contain no more than γL snippets, where $0 < \gamma < 1$ is called a *short factor*. In training, if a sequence is no longer than γL, we directly use the whole sequence without cutting; otherwise, we need to cut it to into multiple *short clips*. When determining the cutting positions, we include as many actions as possible in one *short clip*, and shift the clip boundary inward to exclude boundary actions that are cut in halves. If an action is longer than γL, we don't include it in the video self-stitching stage (note that long actions still get to be detected because we also directly use all original sequences without cutting as xGPN input). Therefore *short clips* may vary in length with the cutting positions. In inference, we do not cut sequences.

Clip up-scaling (Fig. 3 c)). In order to obtain a larger scale, we magnify each *short clip* along the temporal dimension via an up-scaling strategy, such as linear interpolation [20]. For a *short clip*, the up-scaling ratio depends on its own scale. Specifically, if a *short clip* contains M snippet features, then it is up-scaled to a length $L - G - M$, where G is a constant representing a gap length (see next paragraph). In other words, the up-scaled clip will fill in the remaining space in the network input $\mathbf{F}^0$. The shorter a clip is, the longer its up-scaled counterpart will be. This not only makes full use of the input space, but also put more focus on shorter clips.

Self-stitching (Fig. 3 d)). Then we stitch the original *short clip* (**Clip O**) and the up-scaled clip (**Clip U**) into one single sequence. If we directly concatenate the two clips side by side, one issue arises that the network would easily mistake a stitched sequence for a long sequence and tend to generate predictions spanning across the the two clips. To address this issue, we devise a simple strategy: inserting

[1] If a video sequence contains more than L snippet features, we slide a window of length L along the temporal dimension with a stride $L/4$ to generate multiple sub-sequences, each used as an independent sequence in the following steps.

a gap between the two clips, as shown in Fig. 3 d). We simply fill in zeros in the gap to make the network learn to distinguish a long sequence and a stitched sequence by identifying the zeros. This turns out an effective approach.

3.3. Cross-Scale Graph Pyramid Network

Inspired by FPN [21], which computes multi-scale features with different levels, we propose a cross-scale graph pyramid network (xGPN). It progressively aggregates features from cross scales as well as from the same scale at multiple network levels via a hybrid module of a temporal branch and a graph branch. As shown in Fig. 2, our xGPN is composed of a multi-level encoder pyramid and a multi-level decoder pyramid, which are connected by a skip connection at each level. Each encoder level contains a cross-scale graph network (xGN), deeper levels having smaller temporal scales; each decoder level contains an up-scaling network comprised of a de-convolutional [43] layer, deeper levels having larger temporal scales.

Cross-scale graph network. The xGN module contains a temporal branch to aggregate features in a temporal neighborhood, and a graph branch to aggregate features from intra-scale and cross-scale locations. Then it pools the aggregated features into a smaller temporal scale. Its architecture is illustrated in Fig. 4. The temporal branch contains a Conv1d$(3,1)$[2] layer. In the graph branch, we build a graph on all the features from both Clip O and Clip U, and apply edge convolutions [36] for feature aggregation.

Graph building. We denote a directed graph as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_t\}_{t=1}^{J}$ are the nodes, and $\mathcal{E} = \{\mathcal{E}_t\}_{t=1}^{J}$ are the edges pointing to each node. Suppose we have input features $\mathbf{F}^i = \{\mathbf{f}_t^i\}_{t=1}^{J} \in \mathbb{R}^{J\times C}$ at the i^{th} level. We build such a directed graph that each node v_t is a feature $\mathbf{f}_t^i$, which has K inward edges formulated as $\mathcal{E}_t = \{(v_{t_k}, v_t)\,|1 \le k \le K, t_k \ne t\}$. The edges fall into one of the following two categories: free edges and cross-scale edges.

We illustrate these two types of edges in Fig. 4. We make $K/2$ edges of a node free edges, which are only determined based on *feature similarity* between nodes, without considering the source clips. We measure the *feature similarity* between two nodes v_t and v_s using their negative mean square error (MSE), formulated as $-\left\|\mathbf{f}_t^i - \mathbf{f}_s^i\right\|_2^2 / C$. As long as a node is among a target node's top $K/2$ closest neighbors in terms of *feature similarity*, it has a free edge pointing to the target node. Since free edges have no restriction on clip types of the two nodes, they can connect features within the same scale or cross different scales. We make the other $K/2$ edges cross-scale edges, which only connect nodes from different clips, meaning that nodes from Clip O can only have cross-scale edges with nodes from Clip U and vice versa. Given a target node, we pick from those nodes satisfying this condition the top $K/2$ in terms of *feature similarity* after excluding those that already have free edges with the target node. These cross-scale edges enforce correlations between the stitched two clips of different scales. It enables the two scales to exchange information and mutually enhance representations with their complementary properties. In addition, since it enables edges from beyond a node's temporal vicinity, it resolves the *scaling curse* (see Sec. 2.2) of using graph networks on interpolated features.

Feature aggregation. With all the edges of a node $\mathbf{f}_t^i$, we perform edge convolution operations [36] to aggregate features of all its correlated nodes. Specifically, we first concatenate the target node $\mathbf{f}_t^i$ with each of its correlated node $\mathbf{f}_{t_k}^i, 1 \le k \le K$, and apply a multi-layer perceptron (MLP) with weight matrix $\mathbf{W} = \{\mathbf{w}_c\}_{c=1}^{C} \in \mathbb{R}^{2C\times C}$ to transform each concatenated feature. Then, we take the maximum value in a channel-wise fashion to generate the aggregated feature $\tilde{\mathbf{f}}_t^i$. This process is formulated as

$$\tilde{\mathbf{f}}_t^i = \max_{t_k \in \{s|(v_s,v_t)\in\mathcal{E}_t\}} \left(\mathbf{f}_{t_k}^i \,||_C\, \mathbf{f}_t^i\right)^T \mathbf{W}, \tag{1}$$

where $||_C$ is concatenation along the channel dimension.

We fuse the aggregated features from the graph branch and those from the temporal branch by feature summation. Finally, we generate the features for the next level $i+1$ after applying activation and pooling. This is formulated as

$$\mathbf{F}^{i+1} = \sigma_{max}\left(\phi\left(\tilde{\mathbf{F}}^i + \hat{\mathbf{F}}^i\right)\right), \tag{2}$$

where $\tilde{\mathbf{F}}^i = \{\tilde{\mathbf{f}}_t^i\}_{t=1}^{J}$ is the aggregated features of the graph branch, $\hat{\mathbf{F}}^i$ is the output of the temporal branch, ϕ is the rectified linear unit (ReLU), and σ_{max} refers to max pooling.

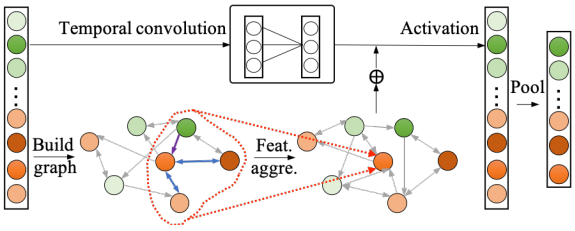

Figure 4. **Cross-scale graph network (xGN)**. Top: temporal branch; bottom: graph branch. The two branches are fused by addition, followed by an activation function and pooling. Each dot represents a feature, green dots from Clip O and orange dots from Clip U. In the graph branch, the blue arrows represent free edges and the purple arrow represents a cross-scale edge.

3.4. Scoring and Localization

As shown in the scoring and localization component of Fig. 2, we use four modules to predict action locations and scores. In the top area, the location prediction module (M_{loc}) and the classification (M_{cls}) module make a coarse

[2]For conciseness, we use Conv1d(m,n) to represent 1-D convolutions with kernel size m and stride n.

prediction directly from each decoder pyramid level. In the bottom area, the boundary adjustment module (M_{adj}) and the supplementary scoring module (M_{scr}) further improve the start/end locations and scores of each predicted segment from the top two modules.

M_{loc} and M_{cls} each contain 4 blocks of Conv1d$(3,1)$, group normalization (GN) [37] and ReLU layers, followed by one Conv1d$(1,1)$ to generate the location offsets and the classification scores for each anchor segment (anchor segments are multi-scale windows at uniformly distributed temporal locations given as references for a prediction module). Here we use pre-defined anchor segments for M_{loc}, whereas for M_{cls} we update the anchor segments by applying their predicted offsets from the M_{loc} module (we use the same mechanism to update the segment boundaries with predicted offsets as in [40]). These two modules are shared by all the decoder levels.

To further improve the boundaries generated from M_{loc}, we design M_{adj} inspired by FGD in [23]. For each updated anchor segment from the M_{loc}, we sample 3 features from around its start and end locations, respectively. Then we temporally concatenate the 3 feature vectors from each location and apply Conv1d$(3,1)$ $-$ ReLU $-$ Conv1d$(1,1)$ to predict start/end offsets. The anchor segment is further adjusted by adding the two offsets to the start and end locations respectively. M_{scr}, comprised of a stack of Conv1d$(3,1)$ $-$ ReLU $-$ Conv1d$(1,1)$, predicts actionness/startness/endness scores [20] for each sequence.

In training, we use a multi-task loss function based on the output of the four modules, which is formulated as

$$\mathcal{L} = \mathcal{L}_{loc} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{adj}\mathcal{L}_{adj} + \lambda_{scr}\mathcal{L}_{scr}, \quad (3)$$

where $\mathcal{L}_{loc}$, $\mathcal{L}_{cls}$, $\mathcal{L}_{adj}$ and $\mathcal{L}_{scr}$ are the losses corresponding to the four modules, respectively, and λ_{cls}, λ_{adj}, and λ_{scr} are their corresponding tradeoff coefficients. The losses $\mathcal{L}_{loc}$ and $\mathcal{L}_{adj}$ are computed based on the distance between the updated / adjusted anchor segments and their corresponding ground-truth actions, respectively. To represent the distance, we adopt the generalized intersection-over-union (GIoU) [30] and adapt it to the temporal domain. For $\mathcal{L}_{cls}$, we use focal loss [22] between the predicted classification scores and the ground-truth categories. $\mathcal{L}_{scr}$ is computed the same way as the TEM losses in [20]. To determine whether an anchor segment is positive or negative, we calculate its temporal intersection-over-union (tIoU) with all ground-truth action instances, and use tIoU thresholds 0.6 for $\mathcal{L}_{loc}$ and $\mathcal{L}_{cls}$, and 0.7 for $\mathcal{L}_{adj}$.

In inference, the score s of each predicted segment $\psi = (t_s, t_e, s)$ is computed with the confidence score c_ψ from M_{cls}, the startness probabilities p_s and the endness probability p_e from M_{scr}, formulated as $s = c_\psi \cdot p_s(t_s) \cdot p_e(t_e)$. We use predictions from both Clip O and Clip U. For predictions from Clip U, we shift the boundaries of each detected segments to the beginning of the sequence and down-scale them back to the original scale to get their locations.

4. Experiments

4.1. Datasets and Setup

Datasets and evaluation metrics. We present our experimental results on two representative datasets THUMOS-14 (THUMOS for short) [14] and ActivityNet-v1.3 (ActivityNet for short) [6]. **THUMOS-14** contains 413 temporally annotated untrimmed videos with 20 action categories, in which 200 videos are for training and 213 videos for validation[3]. **ActivityNet-v1.3** has 19994 temporally annotated untrimmed videos in 200 action categories, which are split into training, validation and testing sets by the ratio of 2:1:1. For both datasets, we use mean Average Precision (mAP) at different tIoU thresholds as the evaluation metric. On THUMOS-14, we use tIoU thresholds $\{0.3, 0.4, 0.5, 0.6, 0.7\}$; on ActivityNet-v1.3, we choose 10 values in the range $[0.5, 0.95]$ with a step size 0.05 as tIoU thresholds following the official evaluation practice.

Implementation Details. In order to achieve higher performance, some works directly process video frames and learn features for the task of temporal action localization (TAL) in an end-to-end fashion [23, 40]. However, this has humongous requirements for GPU memory and computational capability. Instead, we follow the practice of using off-the-shelf pre-extracted features, *without further finetuning on the target TAL task* [2, 18, 20, 42]. For THUMOS, we sample at the original frame rate of each video and pre-extract features using the two-stream network TSN [39] trained on Kinects [15]. For ActivityNet, we evaluate on two different types of features: TSN features at 5 snippets per second and I3D [7] features at 1.5 snippets per second (both networks are trained on Kinetics [15]).

We use an input sequence length $L = 1280$, a channel dimension $C = 256$ throughout the network and a short factor $\gamma = 0.4$. We have 5 levels in the encoder and decoder pyramids respectively, with lengths $L/2^{(l+1)}$, where $1 \le l \le 5$ is the level index. For each level, we have 2 different anchor sizes $\{s_1 \times 2^{(l-1)}, s_2 \times 2^{(l-1)}\}$, where s_1, s_2 are 4, 6 for THUMOS and 32, 48 for ActivityNet. The number of edges for each node is $K = 10$, and the gap is $G = 30$. $\lambda_{cls} = \lambda_{adj} = \lambda_{scr} = 0.2$ for THUMOS and $\lambda_{cls} = \lambda_{adj} = \lambda_{scr} = 1$ for ActivityNet. All these hyper-parameters are empirically selected.

The training batch size is 32 for both datasets. We train 10 epochs at learning rate 0.00005 for THUMOS and 15 epochs at learning rate 0.0001 for ActivityNet. We directly predict the 20 action categories for THUMOS; we conduct binary classification and then fuse our prediction scores

[3]The training and validation sets of THUMOS are temporally annotated videos from the validation and testing sets of UCF101 [32], respectively.

Table 1. **Action detection results on validation set of THUMOS-14**, measured by mAP (%) at different tIoU thresholds. Our VSGN achieves the highest mAP at tIoU threshold 0.5 (commonly adopted criteria), significantly outperforming all other methods.

Method	0.3	0.4	0.5	0.6	0.7	Short
End-to-end learned / Finetuned on THUMOS for TAL						
TCN [9]	-	33.3	25.6	15.9	9.0	-
R-C3D [40]	44.8	35.6	28.9	-	-	-
PBRNet [23]	58.5	54.6	51.3	41.8	29.5	-
Pre-extracted features						
TAL-Net [8]	53.2	48.5	42.8	33.8	20.8	-
P-GCN [44]	63.6	57.8	49.1	-	-	-
I.C&I.C [46]	53.9	50.7	45.4	38.0	28.5	49.1
MGG [26]	53.9	46.8	37.4	29.5	21.3	-
BSN [20]	53.5	45.0	36.9	28.4	20.0	-
DBG [18]	57.8	49.4	39.8	30.2	21.7	-
BMN [19]	56.0	47.4	38.8	29.7	20.5	-
TSI [25]	61.0	52.1	42.6	33.2	22.4	-
G-TAD [42]	54.5	47.6	40.2	30.8	23.4	44.2
BC-GNN [2]	57.1	49.1	40.4	31.2	23.1	-
PBRNet* [23]	54.8	49.2	42.3	33.1	23.0	43.6
VSGN (ours)	**66.7**	**60.4**	**52.4**	**41.0**	**30.4**	**56.6**

* Re-implementation with the same features as ours. We replace 3D convolutions with 1D convolutions to adapt to the feature dimension.

with video-level classification scores from [39] for ActivityNet following [20]. In post-processing, we apply soft-NMS [5] to suppress redundant predictions, keeping 200 predictions for THUMOS and 100 predictions for ActivityNet for final evaluation.

4.2. Comparison with State-of-the-Art

We compare the performance of our proposed VSGN to recent representative methods in the literature on the two datasets in Table 1 and Table 2, respectively. On both datasets, VSGN achieves state-of-the-art performance, reaching mAP 52.4% at tIoU 0.5 on THUMOS and average mAP 35.07% on ActivityNet. It significantly outperforms all other methods that use the same features. More remarkably, our VSGN which uses pre-extracted features without further finetuning, is on par with and even better than concurrent methods that finetune features end to end for TAL.

Besides evaluating all actions in general, we also provide average mAPs of short actions for VSGN as well as other methods that have detection results available. Here, we refer to action instances that are shorter than 30 seconds as short actions. On ActivityNet, there are 54.4% short actions, whereas on THUMOS, there are 99.7% short actions. We can see that our performance gains on short actions over other methods are even more evident.

4.3. Ablation Study

We provide ablation study for the key components VSS and xGPN in VSGN to verify their effectiveness on the two

Table 2. **Action localization results on validation set of ActivityNet-v1.3**, measured by mAPs (%) at different tIoU thresholds and the average mAP. Our VSGN achieves the state-of-the-art average mAP and the highest mAP for short actions. Note that our VSGN, which uses pre-extracted features without further finetuning, significantly outperforms all other methods that use the same pre-extracted features. It is even on par with concurrent methods that finetune the features on ActivityNet for TAL end to end.

Method	0.5	0.75	0.95	Average	Short
End-to-end learned / Finetuned on ActivityNet for TAL					
CDC [31]	45.30	26.00	0.20	23.80	-
R-C3D [40]	26.80	-	-	-	-
PBRNet [23]	53.96	34.97	8.98	35.01	-
Pre-extracted I3D features					
TAL-Net [8]	38.23	18.30	1.30	20.22	-
P-GCN [44]	48.26	33.16	3.27	31.11	-
I.C & I.C [46]	43.47	33.91	**9.21**	30.12	14.8
PBRNet* [23]	51.32	33.33	7.09	33.08	17.6
VSGN (ours)	**52.32**	**35.23**	8.29	**34.68**	**18.8**
Pre-extracted TSN features					
BSN [20]	46.45	29.96	8.02	30.03	15.0
BMN [19]	50.07	34.78	8.29	33.85	15.2
G-TAD [42]	50.36	34.60	9.02	34.09	17.5
TSI [25]	51.18	35.02	6.59	34.15	-
BC-GNN [2]	50.56	34.75	**9.37**	34.26	-
PBRNet* [23]	51.41	34.35	8.66	33.90	18.0
VSGN (ours)	**52.38**	**36.01**	8.37	**35.07**	**19.9**

* Re-implementation with the same features as ours. We replace 3D convolutions with 1D convolutions to adapt to the feature dimension.

Table 3. **Effectiveness of VSGN components for THUMOS-14.** VSS is highly effective for short actions and xGPN further improves the overall performance.

Baseline	VSS	xGPN	0.3	0.4	0.5	0.6	0.7	Short
✓			56.78	50.11	42.54	31.14	19.93	45.1
✓		✓	61.41	55.16	45.52	33.43	21.32	48.7
✓	✓		63.77	58.66	50.24	39.44	28.36	53.4
✓	✓	✓	**66.69**	**60.37**	**52.45**	**40.98**	**30.40**	**56.6**

Table 4. **Effectiveness of VSGN components for ActivityNet-v1.3.** VSS is highly effective for short actions. xGPN benefits actions of different lengths and improves the overall performance.

Baseline	VSS	xGPN	0.5	0.75	0.95	Avg.	Short
✓			51.23	34.91	8.53	34.25	17.5
✓		✓	51.67	35.17	**9.79**	34.70	18.3
✓	✓		50.87	33.99	9.09	33.79	19.7
✓	✓	✓	**52.38**	**36.01**	8.37	**35.07**	**19.9**

datasets in Table 3 and 4, respectively. The baselines are implemented by replacing each xGN module in xGPN with a layer of $\text{Conv1d}(3, 2)$ and ReLU, and not using cutting, up-scaling and stitching in VSS.

Video self-stitching (VSS). For both datasets, VSS shows its effectiveness in improving short actions whether used with or without xGPN. For THUMOS, because most actions are short, the overall performance also has a boost

Table 5. **Complementary properties of Clip O and Clip U (ActivityNet-v1.3)**. Combining predictions from both clips results in higher performance than using either of them.

Predictions from	0.5	0.75	0.95	Avg.	Short
Clip O	52.26	**36.03**	7.98	34.96	19.3
Clip U	51.80	34.79	**8.68**	34.32	19.3
Clip O + Clip U	**52.38**	36.01	8.37	**35.07**	**19.9**

with VSS. For ActivityNet, VSS sacrifices long actions since it reduces the bias towards long actions with more short training samples. We design xGPN to mitigate this effect.

Cross-scale graph pyramid network (xGPN). From Table 3 and 4, we can see that xGPN obviously improves the performance of short actions as well as the overall performance. On the one hand, xGPN utilizes long-range correlations in multi-level features and benefits actions of various lengths. On the other hand, xGPN enables exploitation of cross-scale correlations when used with VSS, thus further enhancing short actions.

Clip O and Clip U. In Table 5, we compare the performance when generating predictions only from Clip O, only from Clip U, and from both with the same well-trained VSGN model. We can see that the two clips still result in different performance even after their features are aggregated throughout the network. Clip O is better at lower tIoU thresholds, whereas Clip U has advantage at a higher tIoU threshold. Combining both predictions can take advantage of the complementary properties of both clips and results in higher performance than using either of them.

4.4. Observations of xGPN

In Table 6, we compare VSGN to the model of only using the xGN modules at certain encoder levels. When we only use xGN in one level, having it in the middle level achieves the best performance. Our VSGN uses xGN for all encoder levels, which achieves the best performance. In Table 7, we compare the mAPs of using different edge types in xGN. Our proposed VSGN uses the top $K/2$ edges as free edges, and then chooses $K/2$ cross-scale edges from the rest. The performance drops if we only use K free edges or K cross-scale edges. K cross-scale edges is better than K free edge, showing the effectiveness of using cross-scale edges.

4.5. Computational Complexity

We compare the inference time of different methods on the ActivityNet validation set on a 1080ti GPU in Table 8. Compared to end-to-end frameworks such as PBRNet, the methods using pre-extracted features such as BMN, G-TAD and VSGN can re-use the features extracted for other tasks, and these methods do not introduce complex 3D convolutions in the TAL architecture, therefore, they have obviously lower inference time. Our VSGN has negligible computation in VSS, and has similar cost in xGPN to the GNNs in G-TAD. Addtionally, it uses fewer anchors (1240 vs 4950), and does not have the stage of ROIAlign, so it runs faster than G-TAD.

Table 6. **xGN levels in xGPN (ActivityNet-v1.3).** We show the mAPs (%) at different tIoU thresholds, average mAPs as well as mAPs for short actions (less than 30 seconds) when using xGN at different xGPN encoder levels. The levels in the columns with ✓use xGN and the ones in the blank columns use a Conv1d$(3, 2)$ layer instead.

xGN levels					mAP (%) at tIoU threshold				Avg. mAP (%)
1	2	3	4	5	0.5	0.75	0.95	Avg.	Short
✓					51.22	34.14	8.22	33.82	19.5
	✓				51.92	34.45	8.89	34.17	19.6
		✓			51.61	34.94	**9.26**	34.46	19.2
			✓		51.10	34.83	8.90	34.19	19.3
				✓	51.10	34.68	8.50	34.03	19.0
✓	✓	✓	✓	✓	**52.38**	**36.01**	8.37	**35.07**	**19.9**

Table 7. **Edge types of each xGN module (ActivityNet-v1.3).** We show the mAPs (%) at different tIoU thresholds 0.5, 0.75, 0.95, average mAPs as well as mAPs for short actions (less than 30 seconds) when using different types of edges in xGN.

Edge types	0.5	0.75	0.95	Avg.	Short
K free	51.59	35.23	7.77	34.48	19.0
K cross-scale	52.33	35.79	7.91	34.75	19.7
$K/2$ free + $K/2$ cross-scale	**52.38**	**36.01**	**8.37**	**35.07**	**19.9**

Table 8. **Inference time of ActivityNet validation set.**

Method	PBRNet	PBRNet*	BMN	G-TAD	**VSGN**
Time (sec)	1600	128	120	183	158

* Our re-implementation using the same pre-extracted features.

5. Conclusions

In this paper, to tackle the challenging problem of large action scale variation in the temporal action localization (TAL) problem, we target short actions and propose a multi-level cross-scale solution called video self-stitching graph network (VSGN). It contains a video self-stitching (VSS) component that generates a larger-scale clip and stitches it with the original-scale clip to utilize the complementary properties of different scales. It has a cross-scale graph pyramid network (xGPN) to aggregate features from across different scales as well as from the same scale. This is the first work to focus on the problem of short actions in TAL, and has achieved significant improvement on short action performance as well as overall performance.

Acknowledgments. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research through the Visual Computing Center (VCC) funding.

References

[1] Humam Alwassel, Silvio Giancola, and Bernard Ghanem. Tsp: Temporally-sensitive pretraining of video encoders for localization tasks. *arXiv preprint arXiv:2011.11479*, 2020. 1

[2] Yueran Bai, Yingying Wang, Yunhai Tong, Yang Yang, Qiyue Liu, and Junhui Liu. Boundary content graph neural network for temporal action proposal generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 121–137, 2020. 1, 2, 3, 6, 7

[3] Yancheng Bai, Yongqiang Zhang, Mingli Ding, and Bernard Ghanem. Sod-mtgan: Small object detection via multi-task generative adversarial network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 206–221, 2018. 2

[4] Alexey Bochkovskiy, Chien-Yao Wang, and H. Liao. Yolov4: Optimal speed and accuracy of object detection. *ArXiv*, abs/2004.10934, 2020. 2

[5] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-nms – improving object detection with one line of code. In *Proceedings of The IEEE International Conference on Computer Vision (ICCV)*, 2017. 7

[6] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 6

[7] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4, 6

[8] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A. Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 3, 7

[9] Xiyang Dai, Bharat Singh, Guyue Zhang, Larry S. Davis, and Yan Qiu Chen. Temporal context network for activity localization in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 7

[10] Victor Escorcia, Mattia Soldan, Josef Sivic, Bernard Ghanem, and Bryan Russell. Temporal localization of moments in video collections with natural language. *ArXiv*, abs/1907.12763, 2019. 1

[11] G. Ghiasi, Tsung-Yi Lin, R. Pang, and Quoc V. Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7029–7038, 2019. 2

[12] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. *arXiv preprint arXiv:1906.02739*, 2019. 3

[13] Lisa Anne Hendricks, O. Wang, E. Shechtman, Josef Sivic, Trevor Darrell, and Bryan C. Russell. Localizing moments in video with natural language. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5804–5813, 2017. 1

[14] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. http://crcv.ucf.edu/THUMOS14/, 2014. 1, 6

[15] W. Kay, J. Carreira, K. Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, F. Viola, T. Green, T. Back, A. Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *ArXiv*, abs/1705.06950, 2017. 6

[16] Thomas Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017. 3

[17] R. Krishna, Kenji Hata, F. Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 706–715, 2017. 1

[18] Chuming Lin, Jian Li, Yabiao Wang, Ying Tai, Donghao Luo, Zhipeng Cui, Chengjie Wang, Jilin Li, Feiyue Huang, and Rongrong Ji. Fast learning of temporal action proposal via dense boundary generator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11499–11506, 2020. 6, 7

[19] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. Bmn: Boundary-matching network for temporal action proposal generation. In *Proceedings of The IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 7

[20] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 4, 6, 7

[21] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. 1, 2, 5

[22] Tsung-Yi Lin, Priyal Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 42:318–327, 2020. 6

[23] Qinying Liu and Zilei Wang. Progressive boundary refinement network for temporal action detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11612–11619, 2020. 1, 3, 6, 7

[24] Shu Liu, Lu Qi, Haifang Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8759–8768, 2018. 2

[25] Shuming Liu, Xu Zhao, Haisheng Su, and Zhilan Hu. Tsi: Temporal scale invariant network for action proposal generation. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2020. 7

[26] Yuan Liu, Lin Ma, Yifeng Zhang, Wei Liu, and Shih-Fu Chang. Multi-granularity generator for temporal action proposal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3604–3613, 2019. 1, 7

[27] Jonghwan Mun, L. Yang, Zhou Ren, N. Xu, and B. Han. Streamlined dense video captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6581–6590, 2019. 1

[28] Mahyar Najibi, Bharat Singh, and Larry S Davis. Autofocus: Efficient multi-scale inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9745–9755, 2019. 2

[29] Han Qiu, Yuchen Ma, Zeming Li, Songtao Liu, and J. Sun. Borderdet: Border feature for dense object detection. In *Proceedings of the European conference on computer vision (ECCV)*, 2020. 2

[30] S. H. Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, I. Reid, and S. Savarese. 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 (CVPR)*, pages 658–666, 2019. 6

[31] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. Cdc: Convolutional-deconvolutional networks for precise temporal action localization in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 7

[32] K. Soomro, A. Roshan Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012. 6

[33] Mingxing Tan, R. Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10778–10787, 2020. 2

[34] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9626–9635, 2019. 2

[35] J. Wang, Wenwei Zhang, Y. Cao, Kai Chen, Jiangmiao Pang, Tao Gong, Jianping Shi, Chen Change Loy, and D. Lin. Side-aware boundary localization for more precise object detection. In *Proceedings of the European conference on computer vision (ECCV)*, 2020. 2

[36] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):1–12, 2019. 3, 5

[37] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, 2018. 6

[38] Zhuyang Xie, Junzhou Chen, and Bo Peng. Point clouds learning with attention-based graph convolution networks. *Neurocomputing*, 402:245–255, 2020. 3

[39] Yuanjun Xiong, Limin Wang, Zhe Wang, Bowen Zhang, Hang Song, Wei Li, Dahua Lin, Yu Qiao, L. Van Gool, and Xiaoou Tang. Cuhk & ethz & siat submission to activitynet challenge 2016. 2016. 4, 6, 7

[40] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, page 5783–5792, 2017. 1, 3, 6, 7

[41] Mengmeng Xu, Juan-Manuel Pérez-Rúa, Victor Escorcia, Brais Martinez, Xiatian Zhu, Li Zhang, Bernard Ghanem, and Tao Xiang. Boundary-sensitive pre-training for temporal localization in videos. *arXiv preprint arXiv:2011.10830*, 2020. 1

[42] Mengmeng Xu, Chen Zhao, David S Rojas, Ali Thabet, and Bernard Ghanem. G-tad: Sub-graph localization for temporal action detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10156–10165, 2020. 1, 2, 3, 6, 7

[43] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2528–2535, 2010. 5

[44] Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan. Graph convolutional networks for temporal action localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7094–7103, 2019. 1, 3, 7

[45] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Z. Lei, and S. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9756–9765, 2020. 2

[46] Peisen Zhao, Lingxi Xie, C. Ju, Y. Zhang, Yanfeng Wang, and Q. Tian. Bottom-up temporal action localization with mutual regularization. In *Proceedings of the European conference on computer vision (ECCV)*, 2020. 2, 7