# Progressive Boundary Refinement Network for Temporal Action Detection

**Qinying Liu, Zilei Wang**
Department of Automation, University of Science and Technology of China
lydyc@mail.ustc.edu.cn, zlwang@ustc.edu.cn

## Abstract

Temporal action detection is a challenging task due to vagueness of action boundaries. To tackle this issue, we propose an end-to-end progressive boundary refinement network (PBRNet) in this paper. PBRNet belongs to the family of one-stage detectors and is equipped with three cascaded detection modules for localizing action boundary more and more precisely. Specifically, PBRNet mainly consists of coarse pyramidal detection, refined pyramidal detection, and fine-grained detection. The first two modules build two feature pyramids to perform the anchor-based detection, and the third one explores the frame-level features to refine the boundaries of each action instance. In the fined-grained detection module, three frame-level classification branches are proposed to augment the frame-level features and update the confidence scores of action instances. Evidently, PBRNet integrates the anchor-based and frame-level methods. We experimentally evaluate the proposed PBRNet and comprehensively investigate the effect of the main components. The results show PBRNet achieves the state-of-the-art detection performances on two popular benchmarks: THUMOS'14 and ActivityNet, and meanwhile possesses a high inference speed.

Understanding human actions in videos is a fundamental issue in computer vision, including action recognition (Carreira and Zisserman 2017; Zhou et al. 2018), temporal action detection (Shou et al. 2017; Chao et al. 2018), actor-action analysis (Yan et al. 2017; Gavrilyuk et al. 2018), etc. Currently these research topics have been active as they are one of important components in extensive applications, *e.g.*, video analysis, video surveillance and human computer interaction. In recent years, action recognition has made great progress benefited from the advances of deep learning (LeCun, Bengio, and Hinton 2015). Compared with action recognition, temporal action detection is still lagging behind in technique and performance due to possessing much larger challenges. Temporal action detection is required to not only identify the class but also localize the start and end time of each action instance in long untrimmed videos.

Temporal action detection, like object detection, belongs to the visual detection family. While object detection aims to produce spatial bounding boxes to precisely surround object instances, temporal action detection aims to identify pairs of temporal frames to precisely segment action instances. Due to such similarity, many efforts (Xu, Das, and Saenko 2017; Lin, Zhao, and Shou 2017a) in action detection take advantage of the frameworks of image object detectors (Ren et al. 2015; Liu et al. 2016). In previous works of temporal action detection, the two-stage methods (Bai et al. 2018; Chao et al. 2018) are usually superior in performance. Comparatively the one-stage approaches (Buch et al. 2017a) are generally more efficient due to eliminating the time-consuming proposal generation. Actually, the one-stage architecture can also achieve good detection performance if we can inherit the merits of two-stage methods.

Compared to the objects which have solid internal consistency, the boundary of an action is vague, because the variations of the consecutive frames are subtle. As addressed in (Alwassel et al. 2018), incorrect localization of action boundaries is the most impactful error which significantly impedes the performances of current approaches. To handle this issue, one thread of methods (Xu, Das, and Saenko 2017; Gao, Yang, and Nevatia 2017; Bai et al. 2018; Chao et al. 2018; Gao, Chen, and Nevatia 2018) apply boundary regression to refine the boundaries. Among them, cascaded boundary regression is a classic yet effective paradigm to improve the localization accuracy. For example, CBR (Gao, Yang, and Nevatia 2017) is proposed to apply boundary regression in an iterative manner–the output boundaries are fed back as input to the network for the next time of refinement. But CBR uses the same structures and features in different steps of boundary regression, which is inconsistent with the requirements for progressive learning. Some other regression-based works (Xu, Das, and Saenko 2017; Bai et al. 2018; Chao et al. 2018) directly adapt the two-stage object detector Faster R-CNN (Ren et al. 2015) to solve the task of action detection, and thus are inborn two-stage cascaded. However, all these methods fail to obtain precise boundaries in a fine granularity. Besides, the additional proposal stage severely slows down the detection speed. Another line of works (Shou et al. 2017; Zhao et al. 2017; Lin et al. 2018; Liu et al. 2019) propose to densely evaluate frame-level scores, and then apply the thresholds on the scores to determine the boundaries of action candidates. As

a result, the boundaries of these proposals are supposed to be fine-grained. Nevertheless, these frame-level methods heavily rely on selection metric or preset threshold, which decides the precision of the action boundaries.

In this paper, we propose a progressive boundary refinement network (PBRNet) to improve the accuracy and speed of temporal action detection. Different from most of previous works, the whole network including the feature extractor is jointly trained. For the procedure of detection, a pipeline of three-step cascaded regression is proposed to refine the boundaries from coarse to fine. Specifically, PBRNet contains three main detection modules: coarse pyramidal detection (CPD), refined pyramidal detection (RPD) and fine-grained detection (FGD). CPD and RPD are anchor-based detection systems, where two symmetric feature pyramids are employed to detect different scales of actions. FGD aims to refine the boundaries of action candidates by exploiting a frame-level feature. Besides, three branches with different kinds of frame-level supervision are used to enrich the frame-level feature and update the classification scores of each action instance. Specially, some learning strategies (*e.g.*, progressive matching strategy and preliminary anchor discarding) are used to corporate with the progressive learning. Hence, the anchors are delivered between the adjacent modules for cascaded regression, and confidence scores from different modules are fused for retrieval.

Our contributions are summarized as follows: (1) We propose an end-to-end temporal action detection network PBRNet, which belongs to the one-stage methods but are equipped with three-step cascaded boundary refinement. (2) PBRNet mainly contains three main detection modules which combine the anchor-based and frame-level methods. The first two modules build two inter-connected temporal pyramids for two-step anchor-based detection. The last module further refine the boundaries by utilizing the fine-grained features, and update the classification score of each anchor through the frame-level predictions. (3) Extensive experiments show that PBRNet can achieve significantly better action detection performance than other state-of-the-art methods, with a remarkable inference speed in the meantime.

## Related Work

Most of existing temporal action detection methods can be categorized into two groups as follows.

The first group of methods (Caba Heilbron, Carlos Niebles, and Ghanem 2016; Shou, Wang, and Chang 2016; Buch et al. 2017b; Gao et al. 2017; Shou et al. 2017; Zhao et al. 2017; Qiu et al. 2018; Gao, Chen, and Nevatia 2018; Lin et al. 2018; Liu et al. 2019; Lin et al. 2019) adopt a two-stage scheme consisting of proposal and classification, which is fairly popular now. Among them, some works (Caba Heilbron, Carlos Niebles, and Ghanem 2016; Gao et al. 2017; Buch et al. 2017b; Gao, Chen, and Nevatia 2018; Lin et al. 2018; Liu et al. 2019; Lin et al. 2019) mainly focus on generating high-quality proposals, and a few works explore how to better classify the proposals (Shou et al. 2017; Zhao et al. 2017; Yang et al. 2018). However, most of them train the two-stage networks separately. Recently, there are some attempts (Xu, Das, and Saenko 2017; Bai et al. 2018; Chao et al. 2018) to jointly train the two-stage networks in a unified framework by following image object detector Faster R-CNN (Ren et al. 2015).

The second group of methods (Yeung et al. 2016; Buch et al. 2017a; Lin, Zhao, and Shou 2017a; Long et al. 2019) draw more attention on the one-stage architectures, motivated by the development of region-free image object detectors such as SSD (Liu et al. 2016). For example, SSAD (Lin, Zhao, and Shou 2017a) assigns anchors to the bottom-up pyramidal layers for predicting the classes and locations of actions. SS-TAD (Buch et al. 2017a) devises two recurrent memory modules to capture temporal information of videos, in which neither the feature pyramid nor anchor mechanism is adopted. Similarly, another work (Yeung et al. 2016) proposes a recurrent network with reinforcement learning. Recently, CTAN (Long et al. 2019) proposes to explore the temporal structure of an action through learning a Gaussian kernel for each cell in the pyramidal layers. Different from these works, our method pursues to build an end-to-end trainable network with anchor mechanism, aiming at taking advantages of two-stage methods into one-stage method.

## Progressive Boundary Refinement Network

The architecture of the proposed progressive boundary refinement network (PBRNet) is illustrated in Figure 1, where the U-net (Ronneberger, Fischer, and Brox 2015) like structure is particularly adopted as one important component. Especially, two types of fusion blocks (*i.e.*, FBv1 and FBv2) are inserted to fuse features from different levels. PBRNet consists of three key modules along with a spatio-temporal feature extractor, *i.e.*, coarse pyramidal detection, refined pyramidal detection and fine-grained detection. Here the first two modules perform the anchor-based detection in multiple scales, and the last one completes fine-grained boundary refinement for each action candidate based on the frame-level feature. Features used in the three modules become much richer or finer gradually. In addition, the three modules will continuously refine action candidates produced by the previous module.

### Main Modules

**Feature Extraction** In this work, we segment videos into clips with a fixed temporal length as the inputs. Let $(L \times H \times W)$ denote the spatio-temporal shape of video clips, where $L$ is the number of continuous frames, $H$ and $W$ are the spatial height and width of the video frames. In our experiments, $H = W = 96$ is set. For feature extraction, we employ I3D (Carreira and Zisserman 2017) as the visual encoder due to its excellent performance in action recognition (Carreira and Zisserman 2017) and temporal action detection (Chao et al. 2018). Specifically, we use the layers of I3D before the last average pooling layer as the backbone network for feature extraction. Then a feature map with the size $(L/8 \times H/32 \times W/32)$ will be generated for each video clip.

**Coarse Pyramidal Detection** In this work, we build temporal feature pyramids in our anchor-based detection mod-

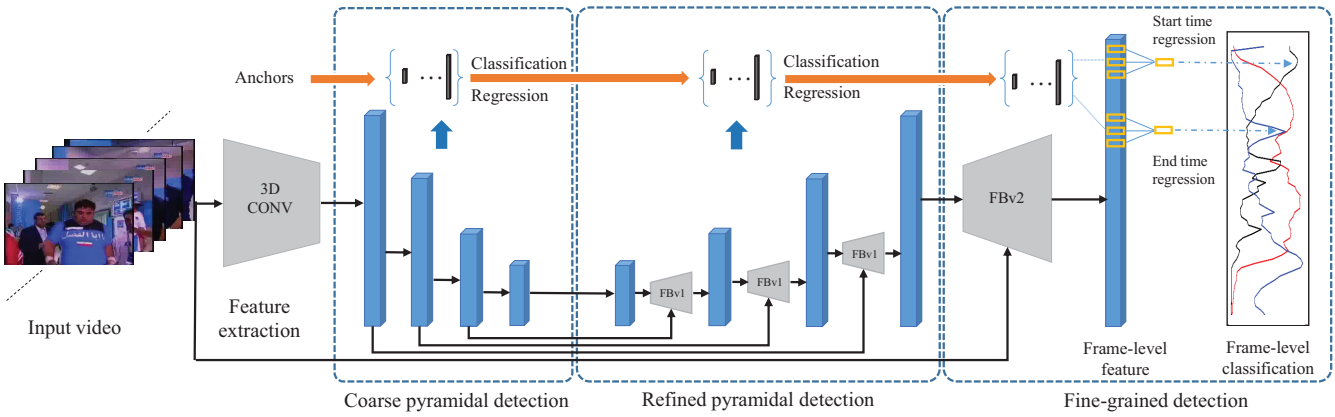


Figure 1: Architecture of our proposed progressive boundary refinement network (PBRNet). PBRNet belongs to the one-stage detectors and mainly consists of three key components: coarse pyramidal detection, refined pyramidal detection and fine-grained detection. The three components sequentially refine the boundaries of action candidates accompanied by the enhancement of features. Best viewed in color.

ules to address the variation of action instances in different temporal scales. In CPD, we take the output of the backbone network as the first anchor layer. For other anchor layers, the temporal scale is reduced by a factor of 2 with maintaining the spatial shape of $3 \times 3$. Taking our implementation on THUMOS'14 (Jiang et al. 2014) for instance, the feature pyramid consists of five layers, denoted by $(B_0, B_1, B_2, B_3, B_4)$, and the corresponding temporal scales are $(L/8, L/16, L/32, L/64, L/128)$, respectively. Here the high-resolution feature maps in the low layers of the pyramid are designated to detect the short action instances and the low-resolution ones are for the long action instances. Like in SSD (Liu et al. 2016), we assign each layer of feature maps a set of preset anchors with several scales. Suppose a feature map has a temporal length of $N$, and we assign $K$ anchors of different scales to each temporal location. Then there are $NK$ anchors in total associated with this anchor layer. For each layer in the pyramid, we conduct the classification and location regression (*i.e.*, predicting offsets to default boundaries) by 3D convolutional layers with the kernel size of $(3 \times 3 \times 3)$.

With CPD, we will obtain the first-level action candidates which are refined from preset anchors. However in the pyramid of CPD, the low layers lack enough semantics and the high layers lack enough fine details, hence the first-level action candidates only have coarse boundaries.

**Refined Pyramidal Detection** To amend the drawbacks of CPD, we propose a more powerful pyramid in RPD. The feature pyramid upscales the temporal resolution gradually, and enriches the features by merging with the feature maps of CPD, as shown in Figure 1. Thus, the feature hierarchy is $(U_0, U_1, U_2, U_3, U_4)$ with the temporal scales $(L/128, L/64, L/32, L/16, L/8)$ in our implementation on THUMOS'14. Specifically, the feature map $U_n$ in this module is generated by merging the feature map $U_{n-1}$ in the previous layer and the corresponding feature map $B_m$ in CPD. More specifically, we use the first kind of fusion block

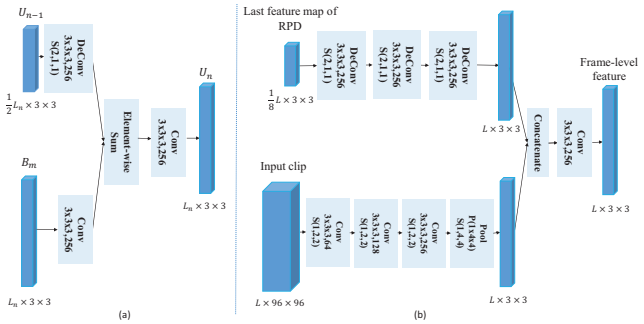


Figure 2: Illustration of two kinds of fusion blocks. (a) FBv1, (b) FBv2. The kernel sizes (along with channels if necessary) and strides of operators are shown. The strides are (1,1,1) where not specified. Best viewed in color.

(*i.e.*, FBv1) for feature fusion, as shown in Figure 2. In each FBv1, we use a 3D deconvolutional layer over $U_{n-1}$ to enlarge the temporal length by a factor of 2, and then apply a 256-channel convolution on $B_m$ to fit the channels of $U_{n-1}$. The element-wise summation is followed to merge both features, and a convolutional layer is then applied to produce $U_n$. Since CPD and RPD have symmetric structure, we directly assign the first-level action candidates as the anchors of RPD. Afterwards, we conduct the classification and regression to produce the second-level action candidates.

Although the structure of two inter-connected temporal pyramids is also applied in (Liu et al. 2019), the novelty of our method is that classification and regression are conducted on both pyramids to form a cascaded detection pipeline.

**Fine-grained Detection** FGD is designed to refine the candidates in a finer granularity. In FGD, we use the second type of fusion block (*i.e.*, FBv2) to generate a frame-level feature. As shown in Figure 2, the frame-level feature

is merged from two parts. The first part takes the last feature map of RPD as input, and then stacks three 3D deconvolutional layers to make temporal length of the feature map equal to the input clip. The second part takes the raw frames as input, and then uses three convolutional layers and a average pooling layer to make its spatio-temporal shape same with that of the first part. We concatenate these two feature maps for fusion, and then use a convolutional layer to generate the ultimate frame-level feature. The feature from the second part can supplement spatial details to the first part. Moreover, we add three frame-level classification branches upon the frame-level feature, which densely outputs three kinds of frame-level class-specific probabilities whether each frame is inside or outside, at or not at the boundaries of a ground-truth instance, denoted by actionness, starting and ending probability, respectively. Each branch is composed of a convolutional layer with the kernel size of $(3 \times 3 \times 3)$ and a softmax layer. The frame-level classification branches have two main functions. First, the auxiliary frame-level supervisions can help to enrich the semantics of the frame-level feature, which will facilitate the fine-grained boundary regression. Second, the frame-level classification scores are used to fuse with the anchor-based scores in the inference time.

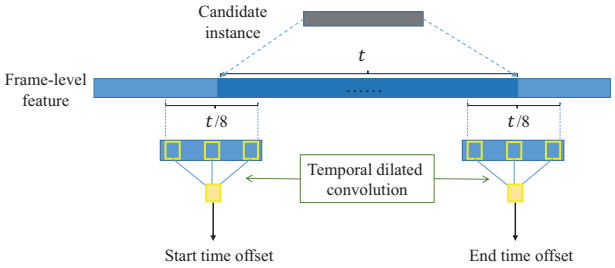


Figure 3: Illustration of boundary refinement in FGD. Best viewed in color.

For fine-grained boundary regression, we separately adjust the start and end time of each action candidate, as shown in Figure 3. Let $s$ and $e$ denote the start/end time of an action candidate, and we will equally process each of them. Without loss of generalization, we explain the refinement procedure by taking $s$ for instance. To be specific, we first locate it in the frame-level feature map, and then choose the neighboring features centered at $s$ as the input to refine $s$ with the temporal length $t/\beta$ ($\beta = 8$ is used in our experiments), where $t = e - s$ is the length of the action candidate. We employ a temporally dilated 3D convolutional layer to predict the final boundary, where the network output is the temporal offset. The kernel size of the dilated convolutional layer is set as $3 \times 3 \times 3$ and the temporal dilation rate is $t/(2 \cdot \beta)$. Such settings enable the receptive field of the dilated convolutional layer reach at $t/\beta$ in the temporal dimension.

After the last time of boundary refinement, we get the third-level action candidates. Then we map each candidate on the frame-level classification branches, and find the corresponding class-specific starting probability of the start location and ending probability of the end location, denoted as $p_s^k$ and $p_e^k$ for the action class of $k$ respectively. We will use these two frame-level class-specific scores to fuse with the anchor-based scores in the test phase.

## Training of PBRNet

**Progressive Matching Strategy** The matching strategy is to determine which anchors correspond to a ground-truth instance. In this work, we calculate the IoU scores of each anchor with all action instances. An anchor is regarded positive if its highest IoU score is greater than the preset threshold $h$, and background otherwise. For positive samples, the matched action instance with the highest score is used as the ground truth. We apply this matching strategy to three detection modules with the thresholds $h_{cp}$, $h_{rp}$ and $h_{fg}$, respectively. We set the IoU thresholds of three modules with increasing values such that actions can be refined progressively. In our implementation, $h_{cp} = 0.5$, $h_{rp} = 0.6$ and $h_{fg} = 0.7$ are used. The similar idea is proposed by the multi-stage cascaded object detector (Cai and Vasconcelos 2018). Here we adopt it in our single-shot framework.

**Preliminary Anchor Discarding** There is serious imbalance between background and foreground after matching, which will badly bias the optimization. To alleviate this issue, we take two steps before feeding the anchors from one module into the next module. First, we discard some well-recognized background anchors. Concretely, we compute the background score for each anchor according to the prediction of previous module, and then only keep the anchors whose background scores are lower than a constant threshold. Second, hard example mining is then applied to keep an acceptable balance between foreground and background samples. Specifically, we only preserve the background anchors with high loss values to make the amounts of the background and foreground anchors approximately equal.

**Loss Function** We use a multi-task loss function to train the network. The overall loss $\mathcal{L}_{total}$ is defined as

$$\mathcal{L}_{total} = \mathcal{L}_{cp} + \lambda_1 \mathcal{L}_{rp} + \lambda_2 \mathcal{L}_{fg}, \tag{1}$$

where $\mathcal{L}_{cp}$, $\mathcal{L}_{rp}$, and $\mathcal{L}_{fg}$ represent the loss functions for CPD, RPD and FGD, respectively. $\lambda_1$ and $\lambda_2$ are the control parameters which are both empirically set to 1.

Since the CPD and RPD have similar detection structures, $\mathcal{L}_{cp}$ and $\mathcal{L}_{rp}$ share a unified formulation $\mathcal{L}_x$ as

$$\mathcal{L}_x = \frac{1}{N}\Big(\sum_i \mathcal{L}_{cls}(\mathbf{p}_i, p_i^*) + \sum_i [p_i^* \geq 1]\mathcal{L}_{loc}(\mathbf{t}_i, \mathbf{t}_i^*)\Big), \tag{2}$$

where $N$ is the number of training anchors, $\mathcal{L}_{cls}$ is the classification loss, and $\mathcal{L}_{loc}$ is the regression loss. Specifically, $\mathbf{p}_i$ is the predicted classification scores for the anchor $i$ and $p_i^*$ is the corresponding ground-truth label. $\mathbf{t} = (t_c, t_l)$ denotes the temporal offset of the predicted center location and duration, and $\mathbf{t}^*$ is the corresponding ground truth.

$\mathcal{L}_{fg}$ consists of two parts, *i.e.*,

$$\mathcal{L}_{fg} = \mathcal{L}_{fg}^{br} + \gamma \mathcal{L}_{fg}^{fc}, \tag{3}$$

The weight factor $\gamma$ is empirically set as 1. $\mathcal{L}_{fg}^{br}$ represents the boundary regression loss, *i.e.*,

$$\mathcal{L}_{fg}^{br} = \frac{1}{N_{br}} \sum_i [p_i^* \geq 1] \mathcal{L}_{loc}(\mathbf{b}_i, \mathbf{b}_i^*), \tag{4}$$

where $N_{br}$ is the number of training action candidates, $\mathbf{b_i} = (s_i, e_i)$ represents the offsets of the predicted start time and end time for the action candidate $i$, and $\mathbf{b}_i^*$ is the corresponding ground truth.

$\mathcal{L}_{fg}^{fc}$ is the loss function for frame-level classification, *i.e.*,

$$\mathcal{L}_{fg}^{fc} = \frac{1}{N_f} \sum_i \big( \mathcal{L}_{cls}(\mathbf{p}_i^a, p_i^{a*}) + \mathcal{L}_{cls}(\mathbf{p}_i^s, p_i^{s*}) + \mathcal{L}_{cls}(\mathbf{p}_i^e, p_i^{e*}) \big). \tag{5}$$

where $N_f$ is the number of frames. $\mathbf{p}_i^a$, $\mathbf{p}_i^s$ and $\mathbf{p}_i^e$ represent the predicted class-specific actionness probability, starting probability and ending probability of the frame $i$, respectively. The $p_i^{a*}$, $p_i^{s*}$ and $p_i^{e*}$ are the corresponding ground-truth label of the frame $i$. To assign frame-level label, for each ground-truth instance with boundary of $(s^*, e^*)$ and category of $c^*$, we define the actionness region, starting region, ending region as $[s^*, e^*]$, $[s^* - t^*/\eta, s^* + t^*/\eta]$ and $[e^* - t^*/\eta, e^* + t^*/\eta]$, where $t^* = e^* - s^*$ is the duration of the ground-truth instance and $\eta$ is empirically set as 10. For each frame, its actionness, starting or ending label will be set as $c^*$ if it lies in the corresponding region.

We use the cross-entropy loss for classification and smooth $\ell_1$ loss for regression throughout the experiments.

## Inference of PBRNet

**Scores Fusion** For each anchor, we first get two anchor-based classification scores output by CPD and RPD (denoted as $p_{cp}^k$ and $p_{rp}^k$ for action class $k$, respectively). We then fuse the anchor-based scores with the frame-level scores ( *i.e.*, $p_s^k$ and $p_e^k$) from FGD by multiplication. Concretely, for each anchor the final confidence score of class $k$ is computed as:

$$p^k = p_{cp}^k \cdot p_{rp}^k \cdot p_s^k \cdot p_e^k \tag{6}$$

The score contain both the anchor-based information and fine-grained information, hence is able to help achieve better evaluation performances.

**Two-stream Fusion** In recent works on temporal action detection (Chao et al. 2018), fusing the appearance and optic flow information by two-stream architectures has shown significantly helpful to achieving good performance due to their complementarity. Similarly, we also investigate the two-stream fusion in our framework. Specifically, we train our proposed networks for two streams separately, and average the predicted values from both streams in the test phase.

**Post-processing** For each anchor, the fused confidence score and three-step regressed boundaries are used for evaluation. Non-maximum suppression (NMS) is conducted to reduce redundant results. We assemble the detection results of the clips belonging to the same video.

# Experiments

## Dataset and Setup

**Dataset** *THUMOS'14* (Jiang et al. 2014) contains 200 untrimmed videos (including $3,007$ action instances) in the validation set and 213 untrimmed videos (including $3,358$ action instances) in the test set which are widely used for temporal action detection from 20 action categories. We use the validation set for training and the test set for evaluation. *ActivityNet v1.3* (Caba Heilbron et al. 2015) contains $10,024$, $4,926$, and $5,044$ videos from 200 classes in the training, validation, and test sets, respectively. We evaluate our model on the validation set, as in previous works (Shou et al. 2017; Gao, Chen, and Nevatia 2018; Xie et al. 2018).

**Evaluation Metrics** For evaluation, we report the mean average precision (mAP) using multiple IoU thresholds. On THUMOS'14, mAP is computed by its official toolkit (Jiang et al. 2014). On ActivityNet, following the protocol provided by the dataset (Caba Heilbron et al. 2015), the IoU thresholds are set as $[0.5 : 0.05 : 0.95]$.

**Experimental Settings** For training, adjacent clips are allowed to be temporally overlapped to increase the amount of training data. We only keep the clips containing at least one complete ground-truth action instance. In the inference stage, the input video is split into clips without temporal overlap. All clips are kept for evaluation. On THUMOS'14, we sample both RGB and optic flow frames at 10 frames per second (fps). The length of each clip $L$ is set as 256 frames (*i.e.*, about 25.6 seconds), which is greater than that of $99.7\%$ of action instances in the dataset. Since ActivityNet is much larger-scale than THUMOS'14, we sample frames at only 3 fps on ActivityNet. Accordingly, $L$ is set as 768 (*i.e.*, covering 256 seconds of a video). Such a temporal length is longer than the duration of over $99.9\%$ of action instances in the training set. We set 6 feature layers in the two pyramids. In addition, considering the fact that one video only contains a single class of action instances for most videos in ActivityNet, we first conduct binary classification in different modules, which indicates whether each anchor contains an action instance, and then assign each detected anchor the top-1 video-level label predicted by (Wang and Tao 2016), by following the protocol in (Lin et al. 2018; 2019). Our backbone is pretrained by (Carreira and Zisserman 2017) on the ImageNet and Kinetics datasets. The batch size is set as 1, thus we freeze all batch normalization layers.

## Action Detection Performance

Table 1 gives the detection performance comparison of our proposed PBRNet with state-of-the-art methods, where the results after two-stream fusion are reported. It can be seen that PBRNet outperforms previous state-of-the-art methods with remarkable mAP gaps for all used IoU thresholds. Particularly, PBRNet achieves the improvement of $8.5\%$ at tIoU $= 0.5$ (from $42.8\%$ to $51.3\%$) compared with TALNet, which is the state-of-the-art two-stage detector in THUMOS'14. Such results imply that one-stage detectors have

| tIoU | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
|---|---|---|---|---|---|
| Wang *et al.* (Wang, Qiao, and Tang 2014) | 14.6 | 12.1 | 8.5 | 4.7 | 1.5 |
| FTP (Caba Heilbron, Carlos Niebles, and Ghanem 2016) | – | – | 13.5 | – | – |
| DAP (Escorcia et al. 2016) | – | – | 13.9 | – | – |
| Oneata *et al.* (Oneata, Verbeek, and Schmid 2014) | 28.8 | 21.8 | 15.0 | 8.5 | 3.2 |
| Richard and Gall (Richard and Gall 2016) | 30.0 | 23.2 | 15.2 | – | – |
| Yeung *et al.* (Yeung et al. 2016) | 36.0 | 26.4 | 17.1 | – | – |
| SMS (Yuan et al. 2016) | 36.5 | 27.8 | 17.8 | – | – |
| Yuan *et al.* (Yuan et al. 2017) | 33.6 | 26.1 | 18.8 | – | – |
| S-CNN (Shou, Wang, and Chang 2016) | 36.3 | 28.7 | 19.0 | 10.3 | 5.3 |
| SST (Buch et al. 2017b) | 37.8 | – | 23.0 | – | – |
| CDC (Shou et al. 2017) | 40.1 | 29.4 | 23.3 | 13.1 | 7.9 |
| SSAD (Lin, Zhao, and Shou 2017a) | 43.0 | 35.0 | 24.6 | – | – |
| TCN (Dai et al. 2017) | – | 33.3 | 25.6 | 15.9 | 9.0 |
| TURN (Gao et al. 2017) | 44.1 | 34.9 | 25.6 | – | – |
| Xiong *et al.* (Xiong et al. 2017) | 48.7 | 39.8 | 28.2 | – | – |
| R-C3D (Xu, Das, and Saenko 2017) | 44.8 | 35.6 | 28.9 | – | – |
| SS-TAD (Buch et al. 2017a) | 45.7 | – | 29.2 | – | 9.6 |
| SSN (Zhao et al. 2017) | 51.9 | 41.0 | 29.8 | – | – |
| CTAP (Gao, Chen, and Nevatia 2018) | – | – | 29.9 | – | – |
| CBR (Gao, Yang, and Nevatia 2017) | 50.1 | 41.3 | 31.0 | 19.1 | 9.9 |
| ETP (Qiu et al. 2018) | 48.2 | 42.4 | 34.2 | 23.4 | 13.9 |
| BSN (Lin et al. 2018) | 53.5 | 45.0 | 36.9 | 28.4 | 20.0 |
| MGG (Liu et al. 2019) | 53.9 | 46.8 | 37.4 | 29.5 | 21.3 |
| GTAN (Long et al. 2019) | 57.8 | 47.2 | 38.8 | – | – |
| BMN (Lin et al. 2019) | 56.0 | 47.4 | 38.8 | 29.7 | 20.5 |
| CMS-RC3D (Bai et al. 2018) | 54.7 | 48.2 | 40.0 | – | – |
| TAL-Net (Chao et al. 2018) | 53.2 | 48.5 | 42.8 | 33.8 | 20.8 |
| PBRNet | **58.5** | **54.6** | **51.3** | **41.8** | **29.5** |

Table 1: Temporal action detection mAP (%) on THUMOS'14.

great potential to beat the two-stage detectors. Table 2 gives the action detection results on ActivityNet. Similarly, PBRNet surpasses other state-of-the-art methods for all listed IoU thresholds, although we sample frames of ActivityNet at only 3 fps. Some qualitative examples are shown in Figure 4.

## Inference Speed

We present the comparison of action detection speed between our PBRNet and other state-of-art methods, as shown in Table 3. Here our model is evaluated on a Nvidia GeFore GTX 1080Ti GPU. It can be seen that PBRNet is able to process frames at a speed of 1488 fps, and thus is one of most competitive methods. The high efficiency of our PBRNet may mainly from the following two facts. First, PBRNet belongs to the family of one-stage detectors and thus eliminates the additional proposal stage used in S-CNN, DAP and R-C3D. Second, PBRNet adopts the fully convolutional operations to process frames rather than the time-consuming recurrent architectures used in DAP and SS-TAD.

## Ablation Study

In order to investigate the effect of key components and settings in our proposed PBRNet, we conduct ablation study on THUMOS'14. Here we only use the threshold of 0.5 to evaluate the methods without specification, since different thresholds have similar performance trends. The results of RGB stream and optical stream are reported separately.

**Main Detection Modules** Anchors are progressively revised in three modules. Here we investigate the performances after using different numbers of modules, where the multiplication is applied to fuse the confidence scores from all used modules. The results are shown in Table 4. It can be seen that the detection performance is boosted gradually as higher-level modules are used to refine action anchors, validating the progressive design of our PBRNet.

**Frame-level Classification Branches** In FGD, three frame-level classification branches are used to enrich the frame-level feature and generate ultimate confidence score. Here the performance are evaluated when the frame-level classification branches are all removed. The second row in Table 5 shows the frame-level classification branches are helpful to boost the action detection performances.

**Progressive Matching Strategy** We adopt the increasing IoU thresholds in PBRNet to match action anchors in the detection modules. Here we conduct another experiment with the same IoU threshold of 0.5 as a comparison. As shown in

| tIoU | 0.5 | 0.75 | 0.95 | Average |
|---|---|---|---|---|
| R-C3D (Xu, Das, and Saenko 2017) | 26.80 | – | – | 12.70 |
| CMS-RC3D (Bai et al. 2018) | 32.92 | 18.36 | 1.13 | 18.46 |
| TCN (Dai et al. 2017) | 36.17 | 21.12 | 3.89 | – |
| TAL-Net (Chao et al. 2018) | 38.23 | 18.30 | 1.30 | 20.22 |
| CDC (Shou et al. 2017) | 43.83 | 25.88 | 0.21 | 22.77 |
| Xiong *et al.* (Xiong et al. 2017) | 39.12 | 23.48 | 5.49 | 23.98 |
| Lin *et al.* (Lin, Zhao, and Shou 2017b) | 44.39 | 29.65 | 7.09 | 29.17 |
| BSN (Lin et al. 2018) | 46.45 | 29.96 | 8.02 | 30.03 |
| BMN (Lin et al. 2019) | 50.07 | 34.78 | 8.29 | 33.85 |
| GTAN (Long et al. 2019) | 52.61 | 34.14 | 8.91 | 34.31 |
| PBRNet | **53.96** | **34.97** | **8.98** | **35.01** |

Table 2: Temporal action detection mAP (%) on ActivityNet v1.3 (val)

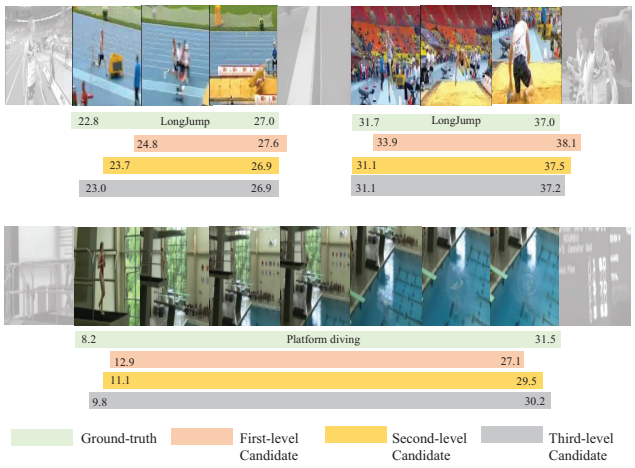


Figure 4: Qualitative examples from THUMOS'14 and ActivityNet. Here we compare the predictions from different levels with ground truth. All the temporal boundaries are showed in seconds. Best viewed in color.

| Methods | FPS |
|---|---|
| S-CNN (Shou, Wang, and Chang 2016) | 60 |
| DAP (Escorcia et al. 2016) | 134 |
| CDC (Shou et al. 2017) | 500 |
| SS-TAD (Buch et al. 2017a) (Titan Xm) | 701 |
| R-C3D (Xu, Das, and Saenko 2017) (Titan Xm) | 569 |
| R-C3D (Xu, Das, and Saenko 2017) (Titan Xp) | 1030 |
| PBRNet (1080Ti) | **1488** |

Table 3: Comparison on action detection speed for test.

| Module | | | mAP@0.5 | |
|---|---|---|---|---|
| CPD | RPD | FGD | RGB | Flow |
| ✓ | | | 36.8 | 37.3 |
| ✓ | ✓ | | 39.7 | 40.6 |
| ✓ | ✓ | ✓ | **42.2** | **42.4** |

Table 4: Ablation study of different modules in PBRNet.

the third row of Table 5, the proposed setting is positive to the performance of action detection.

**Preliminary Anchor Discarding** This is used to filter out the background anchors with high confidence scores for subsequent detection modules, before applying hard example mining. We also conduct an ablation experiment where only hard example mining is used. The results in the fourth row of Table 5 show that the proposed strategy is useful to our network.

| Techniques in PBRNet | mAP@0.5 | |
|---|---|---|
| | RGB | Flow |
| *w/o* Frame-level classification branches | 41.2 | 40.3 |
| *w/o* Progressive matching strategy | 41.3 | 41.5 |
| *w/o* Preliminary anchor discarding | 42.0 | 41.7 |
| *full* | **42.2** | **42.4** |

Table 5: Effect of different techniques in PBRNet, where the last row of results is the baseline.

## Conclusion

In this paper, we proposed a novel progressive boundary refinement network (PBRNet) for temporal action detection, which progressively refine action anchors under the one-stage detection framework. Particularly, we introduced two inter-connected pyramids to perform the anchor-based detection and one refinement module to accurately localize the action boundaries in a fine granularity. Benefited from the well-designed architecture, end-to-end training, and specific learning methods, PBRNet achieves the state-of-the-art performance on THUMOS'14 and ActivityNet.

## Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant 61673362 and 61836008, Youth Innovation Promotion Association CAS (2017496).

## References

Alwassel, H.; Caba Heilbron, F.; Escorcia, V.; and Ghanem, B. 2018. Diagnosing error in temporal action detectors. In *ECCV*.

Bai, Y.; Xu, H.; Saenko, K.; and Ghanem, B. 2018. Contextual multi-scale region convolutional 3d network for activity detection. *arXiv preprint arXiv:1801.09184*.

Buch, S.; Escorcia, V.; Ghanem, B.; Fei-Fei, L.; and Niebles, J. 2017a. End-to-end, single-stream temporal action detection in untrimmed videos. In *BMVC*.

Buch, S.; Escorcia, V.; Shen, C.; Ghanem, B.; and Niebles, J. C. 2017b. Sst: Single-stream temporal action proposals. In *CVPR*.

Caba Heilbron, F.; Escorcia, V.; Ghanem, B.; and Carlos Niebles, J. 2015. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*.

Caba Heilbron, F.; Carlos Niebles, J.; and Ghanem, B. 2016. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *CVPR*.

Cai, Z., and Vasconcelos, N. 2018. Cascade r-cnn: Delving into high quality object detection. In *CVPR*.

Carreira, J., and Zisserman, A. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*.

Chao, Y.-W.; Vijayanarasimhan, S.; Seybold, B.; Ross, D. A.; Deng, J.; and Sukthankar, R. 2018. Rethinking the faster r-cnn architecture for temporal action localization. In *CVPR*.

Dai, X.; Singh, B.; Zhang, G.; Davis, L. S.; and Chen, Y. Q. 2017. Temporal context network for activity localization in videos. In *ICCV*.

Escorcia, V.; Heilbron, F. C.; Niebles, J. C.; and Ghanem, B. 2016. Daps: Deep action proposals for action understanding. In *ECCV*.

Gao, J.; Yang, Z.; Sun, C.; Chen, K.; and Nevatia, R. 2017. Turn tap: Temporal unit regression network for temporal action proposals. In *ICCV*.

Gao, J.; Chen, K.; and Nevatia, R. 2018. Ctap: Complementary temporal action proposal generation. In *ECCV*.

Gao, J.; Yang, Z.; and Nevatia, R. 2017. Cascaded boundary regression for temporal action detection. In *BMVC*.

Gavrilyuk, K.; Ghodrati, A.; Li, Z.; and Snoek, C. G. 2018. Actor and action video segmentation from a sentence. In *CVPR*.

Jiang, Y.; Liu, J.; Zamir, A. R.; Toderici, G.; Laptev, I.; Shah, M.; and Sukthankar, R. 2014. Thumos challenge: Action recognition with a large number of classes.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature* 521(7553):436.

Lin, T.; Zhao, X.; Su, H.; Wang, C.; and Yang, M. 2018. Bsn: Boundary sensitive network for temporal action proposal generation. In *ECCV*.

Lin, T.; Liu, X.; Li, X.; Ding, E.; and Wen, S. 2019. Bmn: Boundary-matching network for temporal action proposal generation. *arXiv preprint arXiv:1907.09702*.

Lin, T.; Zhao, X.; and Shou, Z. 2017a. Single shot temporal action detection. In *ACMMM*.

Lin, T.; Zhao, X.; and Shou, Z. 2017b. Temporal convolution based action proposal: Submission to activitynet 2017. *arXiv preprint arXiv:1707.06750*.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. Ssd: Single shot multibox detector. In *ECCV*.

Liu, Y.; Ma, L.; Zhang, Y.; Liu, W.; and Chang, S.-F. 2019. Multi-granularity generator for temporal action proposal. In *CVPR*.

Long, F.; Yao, T.; Qiu, Z.; Tian, X.; Luo, J.; and Mei, T. 2019. Gaussian temporal awareness networks for action localization. In *CVPR*.

Oneata, D.; Verbeek, J.; and Schmid, C. 2014. The lear submission at thumos 2014. *THUMOS14 Action Recognition Challenge*.

Qiu, H.; Zheng, Y.; Ye, H.; Lu, Y.; Wang, F.; and He, L. 2018. Precise temporal action localization by evolving temporal proposals. In *ICMR*.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*.

Richard, A., and Gall, J. 2016. Temporal action detection using a statistical language model. In *CVPR*.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*.

Shou, Z.; Chan, J.; Zareian, A.; Miyazawa, K.; and Chang, S.-F. 2017. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *CVPR*.

Shou, Z.; Wang, D.; and Chang, S.-F. 2016. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*.

Wang, R., and Tao, D. 2016. Uts at activitynet 2016. *AcitivityNet Large Scale Activity Recognition Challenge* 2016:8.

Wang, L.; Qiao, Y.; and Tang, X. 2014. Action recognition and detection by combining motion and appearance features. *THUMOS14 Action Recognition Challenge* 1(2):2.

Xie, S.; Sun, C.; Huang, J.; Tu, Z.; and Murphy, K. 2018. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*.

Xiong, Y.; Zhao, Y.; Wang, L.; Lin, D.; and Tang, X. 2017. A pursuit of temporal accuracy in general activity detection. In *CVPR*.

Xu, H.; Das, A.; and Saenko, K. 2017. R-c3d: region convolutional 3d network for temporal activity detection. In *ICCV*.

Yan, Y.; Xu, C.; Cai, D.; and Corso, J. 2017. Weakly supervised actor-action segmentation via robust multi-task ranking. *language*.

Yang, K.; Qiao, P.; Li, D.; Lv, S.; and Dou, Y. 2018. Exploring temporal preservation networks for precise temporal action localization. In *AAAI*.

Yeung, S.; Russakovsky, O.; Mori, G.; and Fei-Fei, L. 2016. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*.

Yuan, J.; Ni, B.; Yang, X.; and Kassim, A. A. 2016. Temporal action localization with pyramid of score distribution features. In *CVPR*.

Yuan, Z.-H.; Stroud, J. C.; Lu, T.; and Deng, J. 2017. Temporal action localization by structured maximal sums. In *CVPR*.

Zhao, Y.; Xiong, Y.; Wang, L.; Wu, Z.; Tang, X.; and Lin, D. 2017. Temporal action detection with structured segment networks. In *ICCV*.

Zhou, Y.; Sun, X.; Zha, Z.-J.; and Zeng, W. 2018. Mict: Mixed 3d/2d convolutional tube for human action recognition. In *CVPR*.