Deep Concept-wise Temporal Convolutional Networks for Action Localization

Xin Li¹, Tianwei Lin¹, Xiao Liu¹, Chuang Gan², Wangmeng Zuo³, Chao Li¹, Xiang Long¹, Dongliang He¹, Fu Li¹, Shilei Wen¹
Department of Computer Vision Technology (VIS), Baidu Inc.¹
MIT-Watson AI Lab.² Harbin Institute of Technology.³

Abstract

Existing action localization approaches adopt shallow temporal convolutional networks (i.e., TCN) on 1D feature map extracted from video frames. In this paper, we empirically find that stacking more conventional temporal convolution layers actually deteriorates action classification performance, possibly ascribing to that all channels of 1D feature map, which generally are highly abstract and can be regarded as latent concepts, are excessively recombined in temporal convolution. To address this issue, we introduce a novel concept-wise temporal convolution (CTC) layer as an alternative to conventional temporal convolution layer for training deeper action localization networks. Instead of recombining latent concepts, CTC layer deploys a number of temporal filters to each concept separately with shared filter parameters across concepts. Thus can capture common temporal patterns of different concepts and significantly enrich representation ability. Via stacking CTC layers, we proposed a deep concept-wise temporal convolutional network (C-TCN), which boosts the state-of-the-art action localization performance on THUMOS'14 from 42.8 to 52.1 in terms of mAP(%), achieving a relative improvement of 21.7%. Favorable result is also obtained on ActivityNet.

1. Introduction

Video content analysis methods have been extensively adopted in many real-world applications, such as entertainment, visual surveillance, and robotics. A critical problem in video content analysis is action recognition for understanding human behavior and intent, but action recognition of manually trimmed video clip is unrealistic in practice, where the start and end time of each action instance are usually not annotated for real-world videos. To address this issue, intensive studies have been gradually given to temporal action localization, and considerable progress has been made in the detection and localization of action instances in untrimmed video [5, 29, 62].

Temporal action localization and spatial object detection

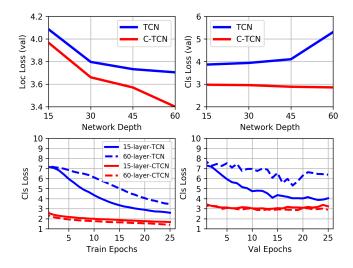


Figure 1. Localization losses (top left) and classification losses (top right) of TCNs and C-TCNs with different network depths on the ActivityNet validation set. Along with the increase of network depth, the classification loss of TCN becomes unexpectedly larger. The training and testing curves (bottom left/bottom right) indicate that stacking more conventional temporal convolution layers deteriorates category modeling rather than improving it.

share similar task paradigms. In particular, object detection aims to find objects in a 2D image in the form of the bounding box positions and object categories. Analogously, the goal of temporal action localization is to detect action instances in a 1D sequence of frames and output the temporal boundaries as well as action categories. Motivated by such similarity, one can transfer the existing 2D spatial convolutional network (CNN) into 1D temporal convolutional network (TCN) for temporal action localization [27, 26, 5, 29, 31]. For example, TAL-Net [5] and SSAD [29] can be regarded as the temporal version of Faster-RCNN [37] and SSD [32], respectively.

However, the transfer from 2D spatial CNN to 1D TCN is not always straightforward and proper architecture design of TCN is still required. Concretely, increasing the depth

of backbone network [15] generally is helpful in improving object detection performance [37, 32, 28, 25]. In contrast, simply stacking more layers in conventional TCN may even cause performance degradation in temporal action localization. To illustrate this point, we train four TCNs on ActivityNet [16] under the same architecture except that the network depths are different (*i.e.*, 15, 30, 45, 60). All models are optimally tuned on the training set and tested on the validation set. As shown in Fig. 1 (a) and (b), the localization performance can be consistently improved along with the increase of network depth, while the classification performance begins to drop when the network depth is higher than 15. Fig. 1(c) and (d) further indicate that the degradation of classification performance cannot be simply explained by over-fitting to training data.

Why performance of conventional TCN degraded with deeper network? To begin with, the input feature sequence **F** of TCN can be denoted as $c \times t$, where t and c are number of snippets and number of feature channels separately. Benefited from deep appearance and motion feature extraction, channels of F generally are highly abstract and can be regarded as disentangled concepts. In a conventional temporal convolution layer, each temporal filter is operated on all c concepts and output the weighted combination. This operation contains two possible drawbacks: (1) the concepts may be excessively recombined, making concepts in deep feature maps are not discriminate enough for action classification; (2) each concept is generated by one temporal 1D convolution filter. If we term the number of filters used in TCN for generating a concept as "Potential", it will be 1 for ordinary temporal 1D. Small Potential can suffer from inferior capability. These drawbacks may explain the performance degradation of deeper TCN, and make deep TCN not a proper choice to boost performance of action localization. A straightforward way to reduce the recombination of concepts is group convolution, which divides the input **F** into c groups and each group is convolved with its own set of k temporal filters. Thus, the shape of the convolved feature map \mathbf{F}' will be $(kc) \times t$, where the filters of different groups have independent parameters. However, our experiments in Sec. 4.2 show that naively utilizing group convolutions provides little benefit in improving the performance of TCN.

Considering that the extracted feature sequence can be viewed as the response sequence of different concepts, we intuitively expect *common temporal patterns of different concepts can be captured*. To achieve this, we propose a novel **concept-wise temporal convolution (CTC) layer**, which deploys a number of temporal filters to each concept separately but shares the filter parameters across concepts. Thus, CTC layer can (1) reduce the concepts recombination, (2) enrich concept context via expanding a concept from one potential to multiple potentials, and (3) capture

common temporal patterns via sharing the filter parameters across concepts. We implement CTC layer in a natural way, via expanding the 1D snippet-level representation into a 2D map, whose two dimensions are potential and concept respectively. In such representation, CTC can be directly achieved and the shape of the convolved feature map will be $k \times c \times t$. Based on stacked CTC layers, we proposed the **Concept-wise Temporal Convolution Network (C-TCN)**, which is an anchor-based temporal action localization network with deep backbone.

In summary, our work has three main contributions:

- We systematically analyze why performance of conventional TCN degraded with deeper network, and reveal that the key solutions are reducing concept combination and capturing common temporal patterns of different concepts to enrich representation ability.
- We propose an novel and effective Concept-wise Temporal Convolution (CTC) layer that allows training deeper model for action localization, and propose a deep Concept-wise Temporal Convolution Network (C-TCN) based on CTC layer.
- Extensive experiments show that C-TCN can outperforms TCN consistently with increasing network depth, and can achieve state-of-the-art performance on both ActivityNet-1.3 and THUMOS-14 datasets.

2. Related Work

2.1. Action Recognition

Action recognition is a fundamental task in video content analysis. The goal of action recognition is to classify a trimmed video with only one action instance. Early approaches proposed many hand-crafted features as spatial-temporal representations [24, 41, 7, 49]. Thanks to the progress made by deep convolution neural network, ConvNet based methods achieved superior accuracy to conventional hand-crafted features. Pre-trained ConvNets are used in [9, 42, 8, 1, 14, 33] to extract convolutional features from video frames, which are encoded into a global representation for classification. Recent methods [45, 23, 11, 53, 48, 4, 36, 50, 55, 56] also explored to design network architectures to exploit the spatial-temporal patterns of actions.

2.2. Temporal Action Localization

Early works apply hand-crafted features and sliding windows [47, 61] for temporal action localization. *E.g.*, Yuan et al. [61] used SVM to encode the iDT features [49] at each position and each resolution, followed by sliding window to localize the actions. In order to combine multiple hand-crafted features, Tang et al. [47] introduced a hierarchical graph method for feature selection and learned the structure

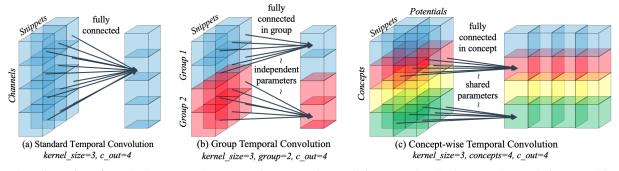


Figure 2. Illustration of standard, group and concept-wise temporal convolutions. (a) Standard temporal convolution recombines all concepts. (b) Group temporal convolution recombines concepts in each group separately with independent parameters. (c) Concept-wise temporal convolution recombines potentials in each concept separately with shared parameters. Thus the discrimination of snippet-level representation is best preserved and the common temporal patterns can be captured.

of the graph using score-based structure learning. Similarly, Heilbron et al. [17] learned sparse dictionaries from multiple features to represent and retrieve activity proposals.

Snippet-level deep features such as two-stream CNN [45], C3D [48] and I3D [4] have achieved both higher efficiency and better performance than handcrafted features. LSTM is used to generate action proposals [10, 59] and detection segments [34, 3], while 1D-temporal convolutional architectures [27, 26, 29] show better performance than LSTM when modeling long range temporal structure of actions. Inspired by state-of-the-art region-based object detectors [40, 39, 37, 32, 28, 25], twostage [43, 5, 44, 18, 13, 6, 12, 52, 64] and one-stage [29, 30] action detectors were proposed for the temporal domain. Xu et al. [58] encoded video streams with a trainable C3D network to generate proposals and classified the selected proposals into specific activities, such that the entire model is trained end-to-end from video frames. Some other works firstly generate snippet-level action scores and then use the labels for contextual reasoning. A few of them [57, 31, 62] aimed to find the start and end boundaries of action proposals, while [38, 19] jointly optimized the appearance, temporal structure and action duration for action detection with structure modeling.

3. Proposed Method

3.1. Snippet-level Feature Coding

Following [27, 26, 29], our model is built upon deep appearance and motion features extracted from raw video frames. We first uniformly divide the video into several small consecutive snippets and then extract visual features within each snippet. In our implementation, the output scores of the "pool5" layer of a pre-trained two-stream model [45] are concatenated as the snippet-level representation. Thus, the shape of the visual feature sequence ${\bf F}$ is $c \times t$, where c is the dimension of the snippet-level representation, and t is the number of snippets.

3.2. Concept-wise Temporal Convolution Layer

The channels of **F** can be regarded as disentangled concepts. Motivated by our analysis of conventional TCN, we expect the concept-wise temporal convolution (CTC) layer to meet three demands: (i) context of each concept should be enriched; (ii) each concept should be convolved separately to avoid the recombination of concepts; (iii) fiter parameters should be shared across concepts to capture common temporal patterns.

In CTC layer, we view the snippet-level feature as an $1 \times c$ tensor such that the shape of **F** is $1 \times c \times t$. Considering the reshaped F as an 1-channel map whose height and width are c and t, we can use $k \times 1 \times 3$ (3 could be other size) filters to convolve the map in temporal dimension and obtain a new convolved feature map \mathbf{F}' whose shape is $k \times c \times t$. As shown in Fig. 2, rectangular filters are utilized such that the concepts are not mixed together. Meanwhile, the filter parameters are shared by all concepts. Thus, CTC layer expands the original concept × snippet feature sequence to the new **potential** \times **concept** \times **snippet** feature map. The potential number of the input data is 1, and the potential number of a hidden layer decided by the filter kernel number of the previous CTC layer. The CTC layer can also be achieved with group convolution and weight sharing. But our implementation is more natural and concise.

3.3. Concept-wise TCN

Based on CTC layer, it is available to build a deep Concept-wise Convolution Network (C-TCN). In this paper, we transfer existing object detection framework - feature pyramid network (FPN) [28] to build C-TCN as an anchorbased action localization network. However, the transfer is not restricted to specific framework. The backbone and FPN adopted in our C-TCN is illustrated in Fig. 3.

Backbone. We build a feature pyramid on the top of the ResNet architecture [15]. Our feature pyramid consists of 8 temporal scales, from P_2 to P_9 , where the resolution of P_l is $1/2^l$ of the input. The resolution of the finest scale P_2 is

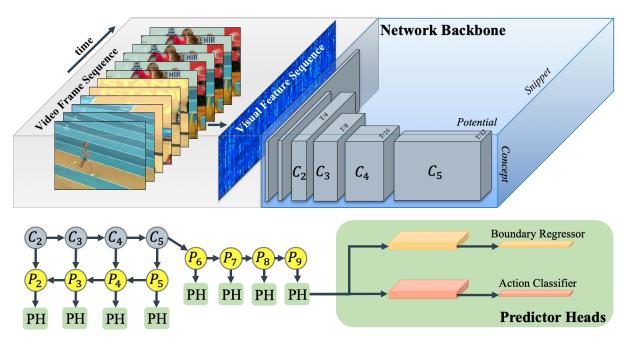


Figure 3. The backbone and feature pyramid of our C-TCN. The backbone is a ResNet with 4 stages of CTC residual blocks, from C_2 to C_5 . We then build feature pyramid of 8 scales (only four are shown here), from P_2 to P_9 , on the top of the residual blocks. Two predictor heads for temporal segment classifier and regressor are then adopted for each scale of feature pyramid.

1/4 of the original input, while the resolution of the coarsest scale P_9 is 1/512 of the original input. Specifically, the snippet number of the input feature sequence is 512, and we use two 1×7 CTC layers with 1×2 stride to decrease its temporal length to 128. The feature map is then passed to a ResNet with 4 stages of CTC residual blocks. The activation outputs of the four residual stages are denoted from C_2 to C_5 . P_6 is computed via a 1×3 CTC layer with 1×2 stride on C_5 . P_7 to P_9 are obtained by applying ReLU followed by a 1×3 CTC layer with 1×2 stride on the previous scale correspondingly. We then compute P_2 to P_5 based on C_2 to C_5 using top-down and lateral connections as in [28]. The architecture and the parameter setting of our CTC are tabulated in supplementary material.

Predictor Head. We use two predictor heads for temporal segment classifier and regressor separately. Each predictor head has a hidden layer followed by a prediction layer. The hidden layer is a CTC layer that decreases the potential number. The prediction layer is a $c \times 1$ convolutional layer that combines information from all concepts. It produces a (A+1)M-channel 2^{9-l} -length score map for P_l , where A+1 is the number of action categories plus one background class, l is the pyramid scale and M is the number of anchor segments. Hence, the temporal segment classifier produces classification scores for M anchor segments at each temporal location and scale of pyramid. Similarly, the prediction layer of the regressor produces a 2M-channel 2^{9-l} -length output for P_l . The location and size offsets are predicted for each anchor segment at each temporal location

and scale of pyramid.

Default Size and Position. Different scales of pyramid have different basic segment sizes. The basic segment size of P_l is set as:

$$s_l = \frac{2^l}{512}T, \quad l = 2\dots 9,$$
 (1)

such that the coarsest scale P_9 has the basic segment size of T, which is the length of the input video. For each scale of pyramid, we use M anchor segments to adjust the basic segment size subtly. Denote the default size of the m^{th} anchor segment of P_l as $s_{l,m}$, we have

$$s_{l,m} = \frac{2}{3}s_l + \frac{2(m-1)}{3M}s_l, \quad m = 1\dots M,$$
 (2)

for the pyramid scale from P_2 to P_8 . It means that P_l occupies a length range from $\frac{2}{3}s_l$ to $\frac{4}{3}s_l$, and the range is divided into M equal portions associated to different anchor sizes. The coarsest scale P_9 occupies the length range from $\frac{2}{3}s_9$ to s_9 , because the length of an action can not be longer than the length of the video.

Regarding that P_l has 2^{9-l} cells, and each one is associated with a default center location of action segment. The default center location of the j^{th} cell in P_l is

$$b_{l,j} = \frac{(2j-1)}{2^{10-l}}T. (3)$$

Classification and Regression. For simplicity, we denote $prop_i^j$ as a temporal proposal of video V_i with index j, and

 $\hat{w}_{i,j}$ and $\hat{c}_{i,j}$ are the default length and center temporal location of the proposal obtained by (1), (2) and (3) according to its pyramid scale, the cell index and the anchor index.

For action classification, we reshape the output of the classifier in each cell to a $(A+1)\times M$ matrix, where each column of the matrix (o^0,o^1,\ldots,o^A) is related to the scores of A+1 categories, and each row is related to an anchor. A softmax layer is adopted to transform the scores to class confidences, and we denote

$$\hat{y}_{i,j}^{k} = \frac{\exp o_{i,j}^{k}}{\sum_{k'} \exp(o_{i,j}^{k'})} \tag{4}$$

as the k^{th} class confidence of $prop_i^j$.

The output of the regressor in each cell is reshaped to a $2 \times M$ matrix, where the first element in each column is the length offset and the second element in each column is the location offset. The regressed length of $prop_i^j$ is $\exp(\hat{\beta}_{i,j})\hat{w}_{i,j}$ and the regressed center temporal location of $prop_i^j$ is $\hat{c}_{i,j} + \hat{w}_{i,j}\hat{\gamma}_{i,j}$, where $\hat{\beta}_{i,j}$ and $\hat{\gamma}_{i,j}$ are length and location offsets of $prop_i^j$ correspondingly.

Matching Strategy and Training Objective. We need to match the temporal proposals to ground-truth actions to train the network. Given $\hat{w}_{i,j}$ and $\hat{c}_{i,j}$, we can calculate the temporal Intersection-over-Union (tIoU) between $prop_i^j$ and the ground-truth actions in V_i . Each temporal proposal is matched to the ground-truth action with the largest tIoU, and is regarded as a positive sample if its largest tIoU is larger than 0.5, otherwise a negative sample.

Our training objective function combines classification loss and localization loss in a multi-task framework. The classification loss of a positive temporal proposal is a cross-entropy form:

$$L_{cla}^{pos}(prop_i^j) = -\log(\hat{y}_{i,j}^{y_{i,\zeta(i,j)}}), \tag{5}$$

where $\zeta(i,j)$ is the index of the matched ground-truth of $prop_i^j$. Similarly, the classification loss of a negative temporal proposal is

$$L_{cla}^{neg}(prop_i^j) = -\log(\hat{y}_{i,j}^0). \tag{6}$$

The localization loss is only applied to positive samples. Inspired by SSD [32] and Faster-RCNN [37], we regress the parametric offsets instead of the start and end boundaries:

$$L_{loc}(prop_i^j) = \operatorname{smooth}_{L1}(\hat{\beta}_{i,j} - \beta_{i,j}) + \operatorname{smooth}_{L1}(\hat{\gamma}_{i,j} - \gamma_{i,j})$$
(7)

where
$$\beta_{i,j}=\log(\frac{w_{i,\zeta(i,j)}}{\hat{w}_{i,j}})$$
 and $\gamma_{i,j}=(c_{i,\zeta(i,j)}-\hat{c}_{i,j})/\hat{w}_{i,j}.$

We thus have the overall training objective as follows:

$$L(\tau, \theta) = \sum_{i} \left(\sum_{j \in \Psi_{i}} L_{cla}^{neg}(prop_{i}^{j}) + \sum_{j \in \Omega_{i}} \left(L_{cla}^{pos}(prop_{i}^{j}) + L_{loc}(prop_{i}^{j}) \right) \right), \tag{8}$$

where Ω_i and Ψ_i denote the positive sample set and negative sample set of V_i respectively, and θ indicates learnable parameters.

Hard Negative Mining and Data Augmentation. Following the spirit of modern object detection approaches [37, 32, 28, 25], For hard negative mining, we apply hard negative mining and data augmentation during training. Instead of using all the negative samples, we pick the negative samples with higher classification loss, and keep the ratio between the negatives and positives samples as 3:1.

Data augmentation is useful for preventing overfitting, but is not adopted in temporal action localization methods yet. In this work, we propose two data augmentation strategies: temporal **random move** and **random crop**. In random move, we first remove all action segments from the video, and concatenate the left background parts to a new video. We then randomly insert action segments to the concatenated video. In random crop, we randomly crop a temporal segment from the video and resize it the size of the original video. At least 50% of a ground-truth action is guaranteed to be left after random crop.

4. Experimental Results

To evaluate the proposed approach, we conduct experiments on two video action localization datasets, *i.e.* ActivityNet-1.3 [16] and THUMOS'14 [21]. The impact of different components of our algorithm is investigated by ablation studies. The comparison between C-TCN with other state-of-the-art methods are also reported.

4.1. Experimental Settings

Dataset. ActivityNet-1.3 consists of more than 648 hours of untrimmed sequences from a total of 20K videos. It contains 200 different daily activities such as "walking the dog", "long jump" and "vacuuming floor". The numbers of videos for training, validation and testing are 10,024, 4,926, and 5,044. We report results on its validation set because the labels of testing set are not released. THUMOS'14 has 20 action classes with 200 untrimmed videos in validation set and 213 videos in testing set. We use its validation set to train our model and report results on its testing set.

Evaluation metrics. We follow conventional evaluation metrics of action localization task: mean Average Precision (mAP) with different tIoU thresholds. On ActivityNet-1.3,

mean of all mAP values computed with tIoU thresholds between 0.5 and 0.95 (inclusive) with a step size of 0.05 is used. On THUMOS'14, mAPs with tIoU thresholds 0.1, 0.2, 0.3, 0.4, 0.5 are adopted. We use the AR-AN (average recall with average number of proposals) curve to evaluate the quality of generated temporal proposals.

Features. To extract snippet-level video feature, we first sample frames from each video at 5fps on both ActivityNet-1.3 and THUMOS'14 datasets, and we then apply a TV-L1 [63] algorithm to get the optical flow of each frame. For ActivityNet-1.3, two TSN [53] models with Senet152 [20] backbones are separately trained on RGB and stacked optical flow, and the two trained models are used to extract two-stream features. For THUMOS'14, we use the TSN models pre-trained on ActivityNet-1.3 to extract two-stream features. An open source I3D model [4] pre-trained on Kinetics is also employed to extract the I3D features. We use an additional linear layer to reduce the dimension of snippet-level representation to 256.

Implementation Details. In the training phase, we train the models using Stochastic Gradient Descent (SGD) with momentum of 0.9, weight decay of 0.0001 and a minibatch size of 16. Additional dropout layers with an ratio of 0.5 are added after the feature maps of all pyramid scales. On THUMOS'14, we set the initial learning rate at 0.001 and reduce once with a ratio of 0.1 after 200 epochs. On ActivityNet-1.3, we set the initial learning rate at 0.0005 and reduce once with a ratio of 0.1 after 20 epochs. The model is trained from scratch and an early-stop strategy is also used to prevent model overfitting. In the testing phase, we apply Soft-NMS for each action class separately and merge the outputs of all classes as the final results.

4.2. Ablation Studies

C-TCN vs. TCN. We trained C-TCNs and conventional TCNs with different numbers of layers on THUMOS'14 and the results are summarized in Table 1. As can be seen, C-TCNs can gain accuracy from increased depth but TCNs get degraded results when the layer number is larger than 15. Furthermore, we see that C-TCNs perform better than TCNs consistently with the same network depth. In Table 1, we also demonstrate the performance of TCN with different groups, which indicates that solely avoid concept recombination in TCN is not enough, but sharing filter parameters is necessary for boosting detection performance.

Feature Setting. The comparison of performance between using TSN and I3D features are summarized in Table 2. As can be seen, the I3D features achieve better performance than the TSN features. As expected, combining the RGB and flow features obtains much higher mAP than using a single modality, demonstrating that the two modalities are heavily complementary for this task. We also investigate the effect of different fusion methods and find early fusion

Table 1. Comparison of C-TCN vs. conventional TCN on THU-MOS'14 in terms of mAP (%). L is the number of layers in network, and G is the group number of TCN.

	L	G	0.1	0.2	0.3	0.4	0.5
TCN	15	1	60.0	58.5	53.9	49.0	41.0
TCN	30	1	63.2	61.5	57.9	51.6	42.4
TCN	45	1	58.0	55.3	51.7	45.6	38.1
TCN	60	1	56.4	53.8	50.4	43.8	36.3
TCN	60	4	57.9	55.5	51.8	46.0	36.0
TCN	60	16	59.6	57.2	52.7	46.1	36.8
TCN	60	64	62.6	59.8	55.2	47.9	36.1
TCN	60	256	39.5	36.4	31.5	24.8	18.1
C-TCN	15	-	70.9	69.4	66.6	59.7	49.0
C-TCN	30	-	71.2	69.9	67.2	60.3	50.5
C-TCN	45	-	71.4	70.3	67.9	60.9	51.4
C-TCN	60	-	72.2	71.4	68.0	62.3	52.1

Table 2. Study of different feature settings on THUMOS'14 in terms of mAP(%)@tIoU.

tIoU	0.1	0.2	0.3	0.4	0.5
TSN RGB	58.9	56.5	51.4	44.0	35.0
TSN flow	47.0	44.9	42.3	37.6	31.1
TSN R. + F. late	63.0	61.6	57.8	50.7	41.7
TSN R. + F. early	66.0	64.5	59.8	52.6	42.5
I3D RGB	60.0	58.1	54.6	47.2	37.6
I3D flow	67.4	66.5	63.5	57.3	49.7
I3D R. + F. late	70.7	69.3	66.4	61.2	51.5
I3D R. + F. early	72.2	71.4	68.0	62.3	52.1

achieving slight better results than late fusion.

Anchor Setting. We try using different number of anchor segments in our experiments, and the results are summarized in Table 3. One can see that using 7 anchors achieves the best performance with the tIoU thresholds of 0.2, 0.3, 0.4 and 0.5, and using 11 anchors achieves the best performance with the tIoU threshold of 0.1. Using more anchors not always obtains better results because more redundant detections are involved. We also compare the performance of using the anchor setting of SSD [32] with 6 anchors. All its mAPs with different tIoUs are lower than our setting with 7 anchors. We can conclude that our new setting is better than the SSD anchor setting for this task.

Data Augmentation Setting. The comparison of different data augmentation settings is summarized in Table 4. It can be seen that data augmentations play critical roles in preventing model overfitting. Without data augmentation, the model converges very fast and the mAP drops seriously. We also find that random move and random crop are highly complementary.

Network Architecture Settings. We also conduct ablation

Table 3. Study of different anchor settings on THUMOS'14 in terms of mAP(%)@tIoU.

tIoU	0.1	0.2	0.3	0.4	0.5
SSD anchor setting	71.5	69.9	64.9	58.7	47.4
3 anchors	68.8	67.9	64.6	59.1	49.3
5 anchors	70.1	68.9	65.0	58.4	47.7
7 anchors	72.2	71.4	68.0	62.3	52.1
9 anchors	69.2	67.8	64.7	58.4	48.9
11 anchors	73.2	71.2	67.9	58.7	46.6
13 anchors	68.4	66.7	63.7	57.2	47.2

Table 4. Study of different data augmentation settings on THU-MOS'14 in terms of mAP(%)@tIoU.

tIoU	0.1	0.2	0.3	0.4	0.5
no random crop	68.6	67.2	63.9	56.2	45.1
no random move	63.3	62.1	59.4	54.8	44.2
no augmentation	60.2	58.2	54.3	47.7	37.4
both augmentations	72.2	71.4	68.0	62.3	52.1

Table 5. The ablation studies of network architecture settings on THUMOS'14 in terms of mAP(%) with tIoU 0.5.

Settings	mAP (%)
remove regression	41.4
kernel size changed to square	45.4
stride size changed to square	39.9
reduce scale number	44.5
remove top-down connections	47.7

studies for other network architecture settings, and the results are summarized in Table 5. Five different settings are conducted to demonstrate the superiorities of the components used in our method. (1) With the regression step during testing removed, the mAP with the threshold of 0.5 drops from 52.1 to 41.4. (2) With the kernel size from 1×3 changed to 3×3 , which means that we use square 2D convolutions instead of concept-wise convolutions, the mAP drops to 45.4. (3) With the stride size from 1×2 changed to 2×2 , which means that the conceptual dimension is downsampled at higher pyramid scale, the mAP drops to 39.9. (4) With the pyramid scale number reduced from 8 to 7 and the finest scale removed, the mAP drops to 44.5. (5) With the top-down and lateral connections removed, the mAP drops to 47.7. The results together prove that the network architecture of C-TCN has been optimally tuned.

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

THUMOS'14. We compare C-TCN with other state-of-the-art methods on THUMOS'14 and summarize the results in Table 6. As our results are significantly better than oth-

Table 6. Action localization mAP(%) on THUMOS'14.								
tIoU	0.1	0.2	0.3	0.4	0.5			
Karaman et al. [22]	4.6	3.4	2.4	1.4	0.9			
Oneata et al. [35]	36.6	33.6	27.0	20.8	14.4			
Wang et al. [51]	18.2	17.0	14.0	11.7	8.3			
Richard et al [38]	39.7	35.7	30.0	23.2	15.2			
Shou et al. [44]	47.7	43.5	36.3	28.7	19.0			
Yeung et al. [60]	48.9	44.0	36.0	26.4	17.1			
Yuan et al. [61]	51.4	42.6	33.6	26.1	18.8			
Shou et al. [43]	-	-	37.8	-	23.0			
Buch et al. [2]	-	-	45.7	-	29.2			
Gao et al. [12]	60.1	56.7	50.1	41.3	31.0			
Dai et al. [6]	-	-	-	33.3	25.6			
Gao et al. [13]	54.0	50.9	44.1	34.9	25.6			
Xu et al. [58]	54.5	51.5	44.8	35.6	28.9			
Zhao et al. [64]	66.0	59.4	51.0	41.0	29.8			
Chao et al. [5]	59.8	57.1	53.2	48.5	42.8			
Lin et al. [31]	-	-	53.5	45.0	36.9			
Ours	72.2	71.4	68.0	62.3	52.1			

ers, we compare our method with others by evaluating the quality of generating action proposals for further analysis. Figure 4 shows the AR-AN curves for action proposals. We find that our method achieves higher AR than others in the low AN region, but is not the best when AN is higher than 40. On the one hand, the curve in low AN region is more closely related to the mAP metrics used in action localization. On the other hand, boundary information is potentially highly complementary with our anchor-based method.

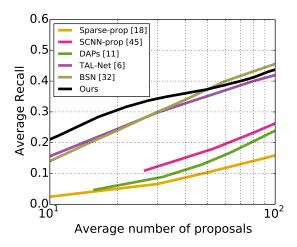


Figure 4. Comparisons in the AR-AN (%) on THUMOS'14.

ActivityNet-1.3. Our comparisons on ActivityNet-1.3 are summarized in Table 7. We can see that the proposed method again achieves the best results among all competing methods. BSN [31] has a higher mAP than our C-TCN with

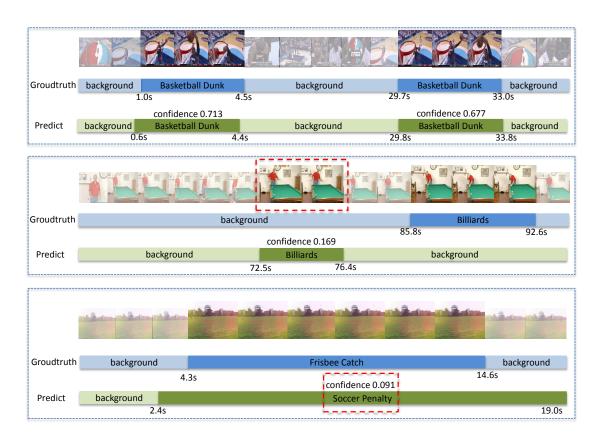


Figure 5. Qualitative results on THUMOS'14 with a good case on the top row and two bad cases on the middle and the bottom rows.

the threshold of 0.95, however, it uses additional video classification results of [65] which are fused by multiple models, while our model only uses a two-stream model.

Table 7. Action localization mAP(%) on ActivityNet v1.3 (val). * means additional video-level classification results are fused.

tIoU	0.5	0.75	0.95	Ave.
Singh et al. [46]	34.4	-	-	-
Wang et al. [54]	43.6	-	-	-
Heilbron et al. [18]	40.0	17.9	4.7	21.7
Shou et al. [43]	45.3	26.0	0.2	23.8
Dai et al. [6]	36.4	21.2	3.9	-
Xu et al. [58]	26.8	-	-	12.7
Chao et al. [5]	38.2	18.2	1.3	20.2
Lin et al. [29]*	44.3	29.6	7.0	29.1
Lin et al. [31]*	46.4	29.9	8.0	30.0
Ours	47.6	31.9	6.2	31.1

4.4. Qualitative Results

Qualitative results on THUMOS'14 with a good case on the top row and two bad cases on the middle and the bottom rows are shown in Figure 5. In the first bad case, a groundtruth billiard action is undetected and a wipe cue stick action is wrongly detected as the billiard action. In the second bad case, our algorithm successfully localize the period that the action happens, but misclassified the frisbee catch action to soccer penalty.

5. Conclusion

In this paper, we studied the challenging task of temporal action localization in video. Delving into the performance degradation along with the increase of network depth, we empirically reveal that degradation of classification performance may ascribe to that all concepts are recombined in temporal convolution. In this work, we present a novel concept-wise temporal convolutional network (C-TCN) to improve temporal action localization. The C-TCN can transfer the popular architecture from spatial 2D CNN and achieves accuracy gains consistently by increasing network depth. Experimental results show that C-TCN brings substantial improvements over the conventional TCN and achieves state-of-the-art performance in ActivityNet and THUMOS'14. In future we will exploit C-TCNs for video classification. Codes are available at PaddleVideo.

References

- R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In CVPR, 2016.
- [2] S. Buch, V. Escorcia, B. Ghanem, L. Fei-Fei, and J. Niebles. End-to-end, single-stream temporal action detection in untrimmed videos. In *BMVC*, 2017. 7
- [3] S. Buch, V. Escorcia, B. Ghanem, L. Fei-Fei, and J. C. Niebles. End-to-end, single-stream temporal action detection in untrimmed videos. In *BMVC* 2017, 2017. 3
- [4] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In CVPR, 2017. 2, 3,
- [5] Y. Chao, S. Vijayanarasimhan, B. Seybold, D. Ross, J. Deng, and R. Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In CVPR, 2018. 1, 3, 7, 8
- [6] X. Dai, B. Singh, G. Zhang, L. S. Davis, and Y. Q. Chen. Temporal context network for activity localization in videos. In *ICCV*, 2017. 3, 7, 8
- [7] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In ECCV, 2006.
- [8] A. Diba, V. Sharma, and L. Van Gool. Deep temporal linear encoding networks. In CVPR, 2017. 2
- [9] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In CVPR, 2015. 2
- [10] V. Escorcia, F. Heilbron, J. Niebles, and B. Ghanem. Daps: Deep action proposals for action understanding. In ECCV, 2016. 3
- [11] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In NIPS, 2016.
- [12] J. Gao, Z. Yang, and R. Nevatia. Cascaded boundary regression for temporal action detection. In BMVC, 2017. 3, 7
- [13] J. Gao, Z. Yang, C. Sun, K. Chen, and R. Nevatia. Turn tap: Temporal unit regression network for temporal action proposals. In *ICCV*, 2017. 3, 7
- [14] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In CVPR, 2017. 2
- [15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016. 2, 3
- [16] F. Heilbron, V. Escorcia, B. Ghanem, and J. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2012. 2, 5
- [17] F. Heilbron, J. Niebles, and B. Chanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In CVPR, 2016. 3
- [18] F. C. Heilbron, W. Barrios, V. Escorcia, and B. Ghanem. Scc: Semantic context cascade for efficient action detection. In CVPR, 2017. 3, 8
- [19] R. Hou, R. Sukthankar, and M. Shah. Real-time temporal action localization in untrimmed videos by sub-action discovery. In *BMVC*, 2017. 3

- [20] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-andexcitation networks. In CVPR, 2018. 6
- [21] Y. Jiang, J. Liu, A. Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. Thumos challenge: Action recognition with a large number of classes. In http://crcv.ucf.edu/THUMOS14/, 2014. 5
- [22] S. Karaman, L. Seidenari, and A. Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. In http://crcv.ucf.edu/THUMOS14/, 2014. 7
- [23] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In CVPR, 2014. 2
- [24] A. Klaser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In BMVC, 2008. 2
- [25] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In ECCV, 2018. 2, 3, 5
- [26] C. Lea, M. Flynn, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks for action segmentation and detection. In CVPR, 2017. 1, 3
- [27] C. Lea, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks: A unified approach to action segmentation. In ECCV, 2016. 1, 3
- [28] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *ECCV*, 2018. 2, 3, 4, 5
- [29] T. Lin, X. Zhao, and Z. Shou. Single shot temporal action detection. In ACM Multimedia, 2017. 1, 3, 8
- [30] T. Lin, X. Zhao, and Z. Shou. Temporal convolution based action proposal: Submission to activitynet 2017. arXiv preprint arXiv:1707.06750, 2017. 3
- [31] T. Lin, X. Zhao, H. Su, C. Wang, and M. Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In ECCV, 2018. 1, 3, 7, 8
- [32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 1, 2, 3, 5, 6
- [33] X. Long, C. Gan, G. de Melo, J. Wu, X. Liu, and S. Wen. Attention clusters: Purely attention based local feature integration for video classification. In CVPR, 2018. 2
- [34] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In CVPR, 2016. 3
- [35] D. Oneata, J. Verbeek, and C. Schmid. The lear submission at thumos 2014. In http://crcv.ucf.edu/THUMOS14/, 2014. 7
- [36] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017.
- [37] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn. In NIPS, 2015. 1, 2, 3, 5
- [38] A. Richard and J. Gall. Temporal action detection using a statistical language model. In CVPR, 2016. 3, 7
- [39] G. Rirshick. Fast r-cnn. In ICCV, 2015. 3
- [40] G. Rirshick, J. Donashue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014. 3
- [41] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In ACM Multimedia, 2007. 2

- [42] Y. Shi, Y. Tian, Y. Wang, W. Zeng, and T. Huang. Learning long-term dependencies for action recognition with a biologically-inspired deep network. In *ICCV*, 2017. 2
- [43] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang. Cdc: convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In CVPR, 2017. 3, 7, 8
- [44] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In CVPR, 2016. 3, 7
- [45] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In NIPS, 2014. 2, 3
- [46] G. Singh and F. Cuzzolin. Untrimmed video classification for activity detection: submission to activitynet challenge. In ActivityNet Large Scale Activity Recognition Challenge, 2016. 8
- [47] K. Tang, B. Yao, L. Fei-Fei, and D. Koller. Combining the right features for complex event recognition. In CVPR, 2013.
- [48] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 2, 3
- [49] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 2
- [50] L. Wang, W. Li, W. Li, and L. Van Gool. Appearance-andrelation networks for video classification. In CVPR, 2018.
- [51] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. In http://crcv.ucf.edu/THUMOS14/, 2014. 7
- [52] L. Wang, Y. Xiong, D. Lin, and L. Van Gool. Untrimmednets for weakly supervised action recognition and detection. In *CVPR*, 2017. 3
- [53] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, and X. Tang. Temporal segment networks: Towards good practices for deep action recognition. In ECCV, 2016. 2, 6
- [54] R. Wang and D. Tao. Uts at activitynet 2016. In ActivityNet Large Scale Activity Recognition Challenge, 2016. 8
- [55] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In CVPR, 2018. 2
- [56] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning for video understanding. In ECCV, 2018. 2
- [57] Y. Xiong, Y. Zhao, L. Wang, D. Lin, and X. Tang. A pursuit of temporal accuracy in general activity detection. In CVPR, 2017. 3
- [58] H. Xu, A. Das, and K. Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017. 3, 7, 8
- [59] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In CVPR, 2016. 3
- [60] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. Endto-end learning of action detection from frame glimpses in videos. In CVPR, 2016. 7

- [61] J. Yuan, B. Ni, X. Yang, and A. Kssim. Temporal action localization with pyramid of score distribution features. In CVPR, 2016. 2, 7
- [62] Z. Yuan, J. C. Stroud, T. Lu, and J. Deng. Temporal action localization by structured maximal sums. In CVPR, 2017. 1, 3
- [63] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In *Proceedings of the 29th* DAGM Conference on Pattern Recognition, 2007. 6
- [64] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin. Temporal action detection with structured segment networks. In *ICCV*, 2017. 3, 7
- [65] Y. Zhao, B. Zhang, Z. Wu, S. Yang, L. Zhou, S. Yan, L. Wang, Y. Xiong, D. Lin, Y. Qiao, and X. Tang. Cuhk & ethz & siat submission to activitynet challenge 2017. arXiv preprint arXiv:1710.08011, 2017. 8

Deep Concept-wise Temporal Convolutional Networks for Action Localization (Supplementary Material)

Xin Li¹, Tianwei Lin¹, Xiao Liu¹, Chuang Gan², Wangmeng Zuo³, Chao Li¹, Xiang Long¹, Dongliang He¹, Fu Li¹, Shilei Wen¹ Department of Computer Vision Technology (VIS), Baidu Inc.¹ MIT-Watson AI Lab.² Harbin Institute of Technology.³

A. Content

The content of this supplementary material involves:

- The architecture of C-TCN.
- Additional results about how group convolution and weight sharing contribute to the performance improvement of C-TCN.

B. Architecture Details

Our C-TCN is comprised of three components, *i.e.*, the backbone, the feature pyramid and the predictor head.

The backbone architecture of C-TCN is shown in Table A. The shape of an input sample is $1\times256\times512$. We use two CTC layers with stride to make C_1 , who is then passed to a ResNet with 4 stages of CTC residual blocks to produce C_2 to C_5 . Batch normalization and ReLU are added to the end of each CTC layer.

The feature pyramid architecture of C-TCN is shown in Table B. P_6 is computed via a 1×3 CTC layer with 1×2 stride on C_5 . P_7 to P_9 are obtained by applying ReLU followed by a 1×3 CTC layer with 1×2 stride on the previous scale correspondingly. We then compute P_2 to P_5 based on C_2 to C_5 using top-down and lateral connections.

Table A: The backbone architecture of C-TCN. The shape of an input sample is $1 \times 256 \times 512$. We use two CTC layers with stride to make C_1 , who is then passed to a ResNet with 4 stages of CTC residual blocks to produce C_2 to C_5 . Batch normalization and ReLU are added to the end of each CTC layer.

backbone stage	output	architecture
C_1	$64 \times 256 \times 128$	1×7 conv, stride 1×2 , 32 1×7 conv, stride 1×2 , 64
C_2	$256 \times 256 \times 128$	$\begin{bmatrix} 1 \times 1 & \text{conv}, 128 \\ 1 \times 3 & \text{conv}, 128 \\ 1 \times 1 & \text{conv}, 512 \end{bmatrix} \times 3$
C_3	$512 \times 256 \times 64$	$\begin{bmatrix} 1 \times 1 & \text{conv}, 256 \\ 1 \times 3 & \text{conv}, 256 \\ 1 \times 1 & \text{conv}, 1024 \end{bmatrix} \times 4$
C_4	$1024 \times 256 \times 32$	$\begin{bmatrix} 1 \times 1 & \text{conv}, 512 \\ 1 \times 3 & \text{conv}, 512 \\ 1 \times 1 & \text{conv}, 2048 \end{bmatrix} \times 6$
C_5	$2048 \times 256 \times 16$	$\begin{bmatrix} 1 \times 1 & \text{conv}, 1024 \\ 1 \times 3 & \text{conv}, 1024 \\ 1 \times 1 & \text{conv}, 4096 \end{bmatrix} \times 3$

The predictor head architecture of C-TCN is shown in Table C. We use two predictor heads for temporal segment classifier and regressor separately. Each predictor head has a hidden layer followed by a prediction layer. The hidden layer is a 1×3 CTC layer that decreases the potential number to 256. The prediction layer is a 256×1 convolutional layer that combines information from all concepts. It produces a (A+1)M-channel 2^{9-l} -length score map for P_l , where A+1 is the number of action categories plus one

background class, l is the pyramid scale and M is the number of anchor segments. Hence, the temporal segment classifier produces classification scores for M anchor segments at each temporal location and scale of pyramid. Similarly, the prediction layer of the regressor produces a 2M-channel 2^{9-l} -length output for P_l . The location and size offsets are predicted for each anchor segment at each temporal location and scale of pyramid.

Table B: The feature pyramid architecture of C-TCN. P_6 is computed via a 1×3 CTC layer with 1×2 stride on C_5 . P_7 to P_9 are obtained by applying ReLU followed by a 1×3 CTC layer with 1×2 stride on the previous scale correspondingly. We then compute P_2 to P_5 based on C_2 to C_5 using top-down and lateral connections.

feature pyramid	output	architecture		
		$1 \times 1 \text{ conv}, 512 (C_2)$		
P_2	$\begin{bmatrix} 512 \times 256 \times 128 \end{bmatrix}$	upsample (P_3)		
1 2	312 × 230 × 126	element-wise add		
		1×3 conv, 512		
		$1 \times 1 \text{ conv}, 512 (C_3)$		
P_3	$512 \times 256 \times 64$	upsample (P_4)		
Г3		element-wise add		
		1×3 conv, 512		
		$1 \times 1 \text{ conv}, 512 (C_4)$		
P_{A}	$512 \times 256 \times 32$	upsample (P_5) element-wise add		
Γ_4				
		1×3 conv, 512		
P_5	$512 \times 256 \times 16$	$1 \times 1 \text{ conv}, 512 (C_5)$		
P_6	$512 \times 256 \times 8$	1×3 conv, stride 1×2 , 512 (C_5)		
P_7	$512 \times 256 \times 4$	1×3 conv, stride 1×2 , 512 (P_6)		
P_8	$512 \times 256 \times 2$	1×3 conv, stride 1×2 , 512 (P_7)		
P_9	$512 \times 256 \times 1$	1×3 conv, stride 1×2 , 512 (P_8)		

Table C: The predictor head architecture of C-TCN. We use two predictor heads for temporal segment classifier and regressor separately. Each predictor head has a hidden layer followed by a prediction layer. The hidden layer is a 1×3 CTC layer that decreases the potential number to 256. The prediction layer is a 256×1 convolutional layer that combines information from all concepts. It produces a (A+1)M-channel 2^{9-l} -length score map for P_l , where A+1 is the number of action categories plus one background class, l is the pyramid scale and M is the number of anchor segments. Hence, the temporal segment classifier produces classification scores for M anchor segments at each temporal location and scale of pyramid. Similarly, the prediction layer of the regressor produces a 2M-channel 2^{9-l} -length output for P_l .

predictor head	output	architecture		
	$256 \times 256 \times 2^{9-l}$	1×3 conv, 256		
classifier of P_l	$(A+1)M \times 1 \times 2^{9-l}$	$256 \times 1 \text{ conv}, (A+1)M$		
	$(A+1)M \times 2^{9-l}$	squeeze		
	$256 \times 256 \times 2^{9-l}$	1×3 conv, 256		
regressor of P_l	$2M \times 1 \times 2^{9-l}$	256×1 conv, $2M$		
	$2M \times 2^{9-l}$	squeeze		

Table D: Ablation study on how group convolution and weight sharing contribute to the performance improvement of C-TCN. The results in terms of mAP(%) on the testing set of THUMOS'14 are reported.

	without weight sharing				with weight sharing					
tIoU	0.1	0.2	0.3	0.4	0.5	0.1	0.2	0.3	0.4	0.5
1 group	56.4	53.8	50.4	43.8	36.3	56.4	53.8	50.4	43.8	36.3
4 groups	57.9	55.5	51.8	46.0	36.0	58.6	57.2	52.8	46.2	38.2
16 groups	59.6	57.2	52.7	46.1	36.8	62.3	60.7	56.9	50.4	40.9
64 groups	62.6	59.8	55.2	47.9	36.1	66.6	65.2	61.8	53.6	43.1
256 groups	39.5	36.4	31.5	24.8	18.1	72.2	71.4	68.0	62.3	52.1

C. Experiments on Group Convolution and Weight-Sharing

We conduct an additional experiment on THUMOS'14 to investigate how group convolution and weight sharing contribute to the performance improvement of C-TCN. The left part of Table D summarizes the results of five TCNs (60 layers) with different number of group convolutions. The right part of Table D summarizes the results of five TCNs with different number of group convolutions, but the filter parameters are shared by all groups. All models are optimally tuned on the validation set of THUMOS'14 and tested on the testing set. We see that: 1) solely using group convolution to avoid concept recombination is not enough for boosting the performance, because more groups cannot bring performance gain. 2) Using the same number of group, weight-sharing consistently improves the performance. 3) With weight-sharing, the performance can be consistently improved by increasing group numbers. Note that when the group number of TCN is 1 and parameters are not shared, it defines a standard TCN. And when the group number of TCN is 256 and the parameters are shared, it equals to a C-TCN. Thus, both group convolution and weight sharing in C-TCN are necessary for boosting the performance of temporal action localization.