Weakly Supervised Temporal Action Localization Using Deep Metric Learning

Ashraful Islam
Rensselaer Polytechnic Institute
islama6@rpi.edu

Richard J. Radke
Rensselaer Polytechnic Institute
rjradke@ecse.rpi.edu

Abstract

Temporal action localization is an important step towards video understanding. Most current action localization methods depend on untrimmed videos with full temporal annotations of action instances. However, it is expensive and time-consuming to annotate both action labels and temporal boundaries of videos. To this end, we propose a weakly supervised temporal action localization method that only requires video-level action instances as supervision during training. We propose a classification module to generate action labels for each segment in the video, and a deep metric learning module to learn the similarity between different action instances. We jointly optimize a balanced binary cross-entropy loss and a metric loss using a standard backpropagation algorithm. Extensive experiments demonstrate the effectiveness of both of these components in temporal localization. We evaluate our algorithm on two challenging untrimmed video datasets: THUMOS14 and ActivityNet1.2. Our approach improves the current state-of-the-art result for THUMOS14 by 6.5% mAP at IoU threshold 0.5, and achieves competitive performance for ActivityNet1.2.

1. Introduction

Video action recognition and action localization are active areas of research. There are already impressive results in the literature for classifying action categories in trimmed videos [5, 41, 40], and important contributions have been made in action localization in untrimmed videos [50, 43, 6]. Temporal action localization is a much harder task than action recognition due to the lack of properly labelled datasets for this task and the ambiguity of temporal extents of actions [29]. Most current temporal action localization methods are fully supervised, i.e., the temporal boundaries of action instances must be known during training. However, it is very challenging to create large-scale video datasets with such temporal annotations. On the other hand, it is much easier to label video datasets with only action instances, since billions of internet videos already have some kind of

weak labels attached. Hence, it is important to develop algorithms that can localize actions in videos with minimum supervision, i.e., only using video-level labels or other weak tags.

In this paper, we propose a novel deep learning approach to temporally localize actions in videos in a weakly-supervised manner. Only the video-level action instances are available during training, and our task is to learn a model that can both classify and localize action categories given an untrimmed video. To achieve this goal, we propose a novel classification module and a metric learning module. Specifically, given an untrimmed video, we first extract equal-length segments from the video, and obtain segment-level features by passing them through a feature extraction module. We feed these features into a classification module that measures segment-level class scores. To calculate the classification score of the whole video, we divide the video into several equal-length blocks, combine the block-level classification scores to get the video-level score, and then apply a balanced binary cross-entropy loss to learn the parameters. To facilitate the learning, we also incorporate a metric learning module. We propose a novel metric function to make frames containing the same action instance closer in the metric space, and frames containing different classes to be farther apart. We jointly optimize the parameters of both of these modules using the Adam optimizer [21]. An overview of our model is shown in Fig. 1.

The proposed method exhibits outstanding performance on the THUMOS14 dataset [18], outperforming the current state of the art by 6.5% mAP at IoU threshold 0.5, and showing comparable results even to some fully-supervised methods. Our method also achieves competitive results on the ActivityNet1.2 [4] dataset.

2. Related Work

Video Action Analysis. There has been significant progress in the field of action recognition and detection, particularly due to the introduction of large-scale datasets [18, 4, 34, 22, 13, 37] and the development of deep learning models. For example, two-stream networks [35], 3D convolutional networks (C3D) [39] and recently I3D networks [5]
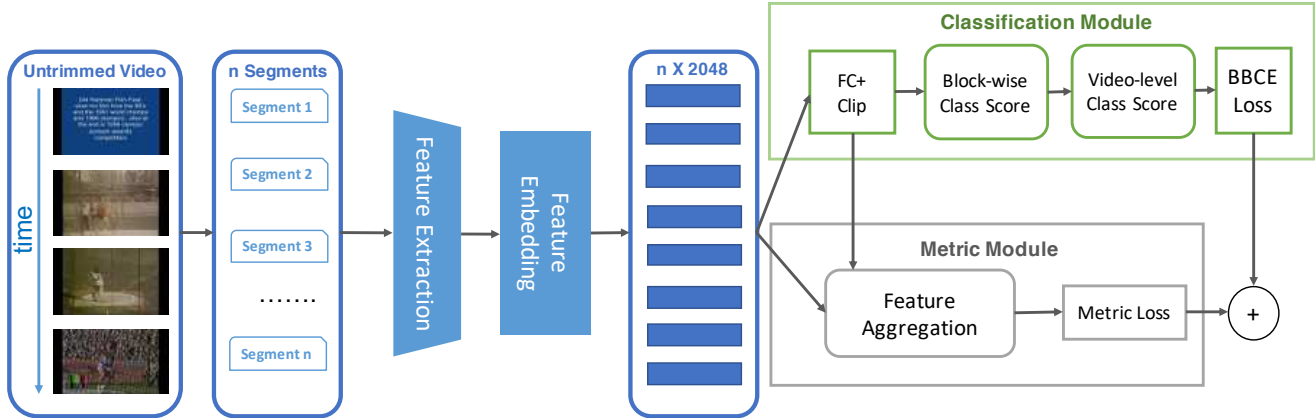
Figure 1: Our algorithm extracts features from video segments and feeds them into classification and metric learning modules. We optimize these jointly to learn the network weights.

have been extensively applied to learn video representations and have achieved convincing performance. For temporal action localization, various deep learning based methods include temporal segment networks [43], structured segment networks [50], predictive-corrective networks [9], and TAL-Net [6]. Most of these techniques use temporal annotations during training, while we aim to use only video-level labels for action localization.

Deep Metric Learning. The objective of metric learning is to learn a good distance metric such that the distance between the same type of data is reduced and the distance between different types of data is enlarged. Traditional metric learning approaches rely on linear mapping to learn the distance metric, which may not capture non-linear manifolds in complex tasks like face recognition, activity recognition, and image classification. To solve this problem, kernel tricks are usually adopted [47, 24]. However, these methods cannot explicitly obtain nonlinear mappings, and also suffer from scalability problems. With the advent of deep learning, deep neural network-based approaches have been used to learn non-linear mappings in metric learning. For example, Hu *et al.* [16] trained a deep neural network to learn hierarchical non-linear mappings for face verification. Bell and Bala [1] learned visual similarity using contrastive embedding [14]. Schroff *et al.* [30] used triplet embedding [45] on faces for face verification and clustering.

Weakly-Supervised Temporal Localization. Weakly supervised deep learning methods have been widely studied in object detection [2, 8, 26], semantic segmentation [15, 20], visual tracking [51], and video summarization [15]. However, there are only a few weakly supervised methods in temporal action localization that rely only on video-level labels during training. It should be noted that there are different types of weak supervision for the temporal localization task. For example, some works use movie scripts or subtitles as weak supervision [3, 10], whereas others use the temporal order of actions during training [28, 17]. We do not use any information about temporal ordering in our model. Our approach only uses a set of action classes for each video during training.

Wang *et al.* [42] proposed a model named Untrimmed-Nets consisting of a classification module that predicts the classification scores for each video clip and a selection module that detects important video segments. The algorithm uses a Softmax function to generate action proposals, which is not ideal for distinguishing multiple action classes. It is also based on a temporal segments network [43] that considers a fixed number of video segments, which is not effective for variable-length video datasets. Nguyen *et al.* [25] added a sparsity-based loss function and class-specific action proposals (contrary to class-agnostic proposals in UntrimmedNets). However, the sparsity constraint for attention weights that they propose would hurt localization performance in videos that contain very few background activities.

Shou *et al.* [32] introduced Outer-Inner-Contrastive Loss to automatically predict the temporal boundaries of each action instance. Paul *et al.* [27] proposed techniques that combine Multiple Instance Learning Loss with Co-activity Similarity Loss to learn the network weights. Our proposed method is similar to this work with novel contributions in several important areas. In particular, we adopt a block-based processing strategy to obtain a video-level classification score, and propose a novel metric function as a similarity measure between activity portions of the videos. Su *et al.* [38] proposed shot-based sampling instead of uniform sampling and designed a multi-stage temporal pooling network for action localization. Zeng *et al.* [49] proposed an iterative training strategy to use not only the most discriminative action instances but also the less discriminative ones.

Liu *et al.* [23] recently proposed a multi-branch architecture to model the completeness of actions, where each branch is enforced to discover distinctive action parts. They also used temporal attention similar to [25] to learn the importance of video segments, showing a minor performance improvement over [27].

3. Proposed Algorithm

In this section, we introduce the detailed pipeline of our proposed algorithm. We first describe the data processing and feature extraction modules. We then present the classification and deep metric learning modules and introduce loss functions to jointly optimize them [1].

Problem Formulation. We consider an untrimmed video as a collection of segments, where each segment contains an equal number of frames. Let a video $\mathbf{V}$ be represented as a collection of segments $\{c_i\}_{i=1}^n$, where n is the total segment length, and an associated activity class set with n_c unique activity instances represented as $\mathbf{a} = \{a^k\}_{k=1}^{n_c}$, where $a^k \in \mathcal{A}$, the set of all action classes in the dataset. The training data set contains N videos $\{\mathbf{V}_i\}_{i=1}^N$ with their associated labels $\{\mathbf{a}_i\}_{i=1}^N$. The length and activity instances in the video can vary significantly, and we only have video-level labels during the training period. Given a test video, the model will predict a set of action labels with corresponding start time, end time and confidence score.

3.1. Feature Extraction

We extract segment-level features $\{\mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional feature vector, and n is the segment length of the video. Two-stream networks have become common for action recognition and detection [5, 11]. Following [25], we use the I3D network [5] pretrained on the Kinetics dataset [19] to extract features from each video segment. Both the RGB and optical flow streams are used for feature extraction, and we fuse them together to get a single feature vector for each video segment. We use the TV-L1 algorithm [44] to extract the flow. We do not use any fine-tuning on this feature extractor network.

3.2. Feature Embedding

Given a feature representation of a video $\mathbf{V}$ as $\{\mathbf{x}_i\}_{i=1}^n$, we feed the features to a module consisting of a fully connected layer followed by a ReLU and a dropout layer. This module modifies the original features extracted from the pre-trained feature extraction module into task-specific embedded features. We keep the dimension of the embedded features the same as the dimension of the extracted features. The embedded features are denoted by $\{\mathbf{u}_i\}_{i=1}^n$, where $\mathbf{u}_i \in \mathbb{R}^d$.

---

[1]Code accompanying this paper is available at https://github.com/asrafulashiq/wsad.git

3.3. Classification Module

Next, we learn a linear mapping $\mathbf{W}_f \in \mathbb{R}^{C \times d}$ and bias $\mathbf{b} \in \mathbb{R}^C$ followed by a clipping function $\varphi_\kappa(\cdot)$ to obtain class-specific activations $s_i \in \mathbb{R}^C$ for each segment, where C is the total number of class labels, i.e.,

$$s_i = \varphi_\kappa(\mathbf{W}_f \mathbf{x_i} + \mathbf{b}) \tag{1}$$

where $\varphi_\kappa(\cdot)$ is defined by

$$\varphi_\kappa(x) = \begin{cases} \kappa & \text{if } x > \kappa \\ -\kappa & \text{if } x < -\kappa \\ x & \text{otherwise} \end{cases}$$

The necessity of using a clipping function is discussed in Sec. 3.4.

To obtain the video-level classification score, we use a block-based processing strategy. Specifically, since the total segment length n of a video $\mathbf{V}$ can vary, we divide the video into blocks, where each block is a set of an equal number of consecutive segments, i.e., $\mathbf{V} = \{B_i\}_{i=1}^{n_B(\mathbf{V})}$, where $n_B(\mathbf{V}) = \left\lfloor \frac{n}{l_w} \right\rfloor$ is the total number of blocks, and l_w is the number of segments in each block. We empirically chose the value of l_w (discussed in Sec. 4.4).

We calculate $P(c \mid \mathbf{V})$, the probability of the video $\mathbf{V}$ containing particular class c, as

$$P(c \mid \mathbf{V}) = P\left(c \mid \{B_i\}_{i=1}^{n_B(\mathbf{V})}\right) \tag{2}$$

$$= 1 - \prod_{i=1}^{n_B(\mathbf{V})} (1 - P(c \mid B_i)) \tag{3}$$

where $P(c \mid B_i)$ is the probability that the i-th block contains class c. One approach to obtain this probability is to pick the highest class activation in that block. However, an activity would likely cover several video segments. Hence, following [27], we compute the average of the k-max class activation scores in the block as

$$P(c \mid B_i) = \sigma \left(\frac{1}{k} \max_{l \subset \mathcal{I}_i} \sum_{j=1}^{k} s_{l_j}^c \right) \tag{4}$$

where $\mathcal{I}_i$ contains the segment indices for the i-th block, $\sigma(\cdot)$ is the sigmoid activation function, and $s_{l_j}^c$ is the class activation score for the l_j-th segment.

We compute $P(c \mid \mathbf{V})$ for each class $c \in \{1, 2, \ldots, C\}$. As a video can contain multiple activities, this is a multi-label classification problem. Hence, the binary cross-entropy loss (BCE) is an obvious choice. However, we found through experiments that the standard BCE loss performs poorly in this case, mainly due to the class-imbalance

problem. Xie and Tu [46] first introduced a class-balancing weight to offset the class-imbalance problem in binary cross entropy. Similar to them, we introduce a balanced binary cross-entropy loss, which produces better results in practice. We calculate the balanced binary cross-entropy (BBCE) loss as

$$\mathcal{L}_{\text{BBCE}} = \frac{\sum_{c=1}^{C} y_c \log P(c \mid \mathbf{V})}{\sum_{c=1}^{C} y_c} \tag{5}$$

$$+ \frac{\sum_{c=1}^{C}(1 - y_c) \log(1 - P(c \mid \mathbf{V}))}{\sum_{c=1}^{C}(1 - y_c)} \tag{6}$$

Here, y_c is set to 1 if the video contains class c, otherwise it is set to 0. The effectiveness of $\mathcal{L}_{\text{BBCE}}$ is demonstrated in Sec. 4.

3.4. Metric Learning Module

Here, we first give a brief review of distance metric learning, and how it is incorporated in our algorithm.

Distance Metric Learning. The goal of metric learning is to learn a feature embedding to measure the similarity between input pairs. Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}$ be input features and $\mathbf{Y} = \{y_i\}_{i=1}^{n}$ be corresponding labels. We want to learn a distance function $D(\mathbf{x}_i, \mathbf{x}_j) = f(\theta; \mathbf{x}_i, \mathbf{x}_j)$, where f is the metric function and θ is a learnable parameter. Various loss functions have been proposed to learn this metric. Contrastive loss [7, 14] aims to minimize the distance between similar pairs and penalize the negative pairs that have distance less than margin α:

$$\mathcal{L}_{\text{contrastive}}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{1}(y_i = y_j)D^2(\mathbf{x}_i, \mathbf{x}_j) + \tag{7}$$

$$\mathbf{1}(y_i \neq y_j)[\alpha - D^2(\mathbf{x}_i, \mathbf{x}_j)]_+ \tag{8}$$

where $[\cdot]_+$ indicates the hinge function $\max(0, \cdot)$.

On the other hand, triplet loss [45] aims to make the distance of a negative pair larger than the distance of a corresponding positive pair by a certain margin α. Let $\{\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n\}$ be a triplet pair such that $\mathbf{x}_i^a$ and $\mathbf{x}_i^p$ have the same label and $\mathbf{x}_i^a$ and $\mathbf{x}_i^n$ have different labels. The triplet loss is defined as:

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n) = [D^2(\mathbf{x}_i^a, \mathbf{x}_i^p) - D^2(\mathbf{x}_i^a, \mathbf{x}_i^n) + \alpha]_+ \tag{9}$$

Motivation. A set of videos that have similar activity instances should have similar feature representations in the portions of the videos where that activity occurs. On the other hand, portions of videos that have different activity instances should have different feature representations. We incorporate the metric learning module to apply this characteristic in our model.

Our Approach. We use embedded features and class-activation scores to calculate the aggregated feature for a particular class. Let $\mathcal{B}_c = \{\mathbf{V}_k\}_{k=1}^{N}$ be a batch of

videos containing a common class c. After feeding the video segments to our model, we extract embedded features $\{\mathbf{u}_{k,i}\}_{i=1}^{n_k}$ and class activation scores $\{s_{k,i}\}_{i=1}^{n_k}$ for the k-th video, where n_k is the length of the video. Following [27], we calculate the aggregated feature vector for class c from video $\mathbf{V}_k$ as follows:

$$\mathbf{z}_k^c = \sum_{i=1}^{n_k} \pi_{k,i}^c \mathbf{u}_{k,i} \quad \text{and} \quad \mathbf{z}_k^{\neg c} = \sum_{i=1}^{n_k} \frac{1 - \pi_{k,i}^c}{n_k - 1} \mathbf{u}_{k,i}$$

where $\pi_{k,i}^c = \frac{\exp(s_{k,i}^c)}{\sum_{i'=1}^{n_k} \exp(s_{k,i'}^c)}$. Here, $s_{k,i}^c$ is the class activation of the i-th segment for class c in video $\mathbf{V}_k$. Hence, $\mathbf{z}_k^c$ is aggregated from feature vectors that have high probability of containing class c, and $\mathbf{z}_k^{\neg c}$ is aggregated from feature vectors that have low probability of containing class c. We normalize these aggregated features to a d-dimensional hypersphere to calculate $\tilde{\mathbf{z}}_k^c$ and $\tilde{\mathbf{z}}_k^{\neg c}$, i.e. $||\tilde{\mathbf{z}}_k^c||_2 = 1$ and $||\tilde{\mathbf{z}}_k^{\neg c}||_2 = 1$. Here, we can see the motivation behind applying a clipping function in Eqn. 1. If the clipping function is not applied, there might be a segment i_h with a very high class score s_{k,i_h}^c, and the value of π_{k,i_h}^c, which is the output of a Softmax function, will be close to 1 for that segment and close to 0 for other segments. Hence, the aggregated features will be calculated mostly from the segment with maximum class score, even though there are other segments that can have high class score for a particular class. Therefore, we apply a clipping function to limit the class score to have a certain maximum and minimum value.

Next, the average distances for positive and negative pairs from a batch of videos with common class c are calculated as

$$d^{+,c} = \frac{1}{n_k(n_k - 1)} \sum_{\substack{1 \leq j,j' \leq n_k \\ j \neq j'}} D_c^2(\tilde{\mathbf{z}}_j^c, \tilde{\mathbf{z}}_{j'}^c),$$

$$d^{-,c} = \frac{1}{n_k(n_k - 1)} \sum_{\substack{1 \leq j,j' \leq n_k \\ j \neq j'}} D_c^2(\tilde{\mathbf{z}}_j^c, \tilde{\mathbf{z}}_{j'}^{\neg c})$$

Instead of using cosine distance as the distance function, our intuition is that D_c should be different for different classes, and hence we define $D_c(\mathbf{u}, \mathbf{v}) = ||\mathbf{W}_f^c(\mathbf{u} - \mathbf{v})||_2$, where $\mathbf{W}_f^c \in \mathbb{R}^{1 \times d}$ is the c-th row of the weight matrix of the final fully-connected layer of our model. To clarify why this is a proper distance function in this case, we can write $D_c(\cdot, \cdot)$ as:

$$D_c(\mathbf{u}, \mathbf{v}) = \sqrt{(\mathbf{u} - \mathbf{v})^\top (\mathbf{W}_f^c)^\top \mathbf{W}_f^c (\mathbf{u} - \mathbf{v})} \tag{10}$$

$$= \sqrt{(\mathbf{u} - \mathbf{v})^\top \mathbf{M}^c (\mathbf{u} - \mathbf{v})} \tag{11}$$

where $\mathbf{M}^c = (\mathbf{W}_f^c)^\top \mathbf{W}_f^c$ is a symmetric positive semi-definite matrix. Hence, Eqn. 10 is actually a Mahalanobis

type distance function, where the metric $\mathbf{M}^c$ is calculated from the weights of a neural network. Additionally, the class score for class c is calculated from the weight $\mathbf{W}_f^c$; hence $\mathbf{M}^c$ is a metric that can be used in the distance measure only for class c. We show in the ablation studies that our proposed distance function is a better metric in this setting.

Finally, we calculate either the triplet loss $\mathcal{L}_{\text{triplet}}^c = [d^{+,c} - d^{-,c} + \alpha]_+$ or contrastive loss $\mathcal{L}_{\text{contrastive}}^c = d^{+,c} + [\alpha - d^{-,c}]_+$ as the metric loss function. We found through experiments that triplet loss performs slightly better than contrastive loss. Hence, we use triplet loss unless stated otherwise.

3.5. Temporal Localization

Given an input test video, we obtain the segment level class score $y_i^c = \sigma(s_i^c)$ where $\sigma(\cdot)$ is the sigmoid function, and calculate the video-level class score $\bar{y}^c$ for each class c following Eqn. 2. For temporal localization, we detect action instances for each class in a video separately. Given class scores y_i^c for the i-th segment and class c, we first discard all segments that have class score less than threshold 0.5. The one-dimensional connected components of the remaining segments denote the action instances of the video. Specifically, each action instance is represented by (i_s, i_e, c, q) where i_s is the start index, i_e is the end index, c is the action class, and q is the class score calculated as $q = \max(\{y_i^c\}_{i=i_s}^{i_e}) + \gamma \bar{y}^c$, where γ is set to 0.7.

4. Experiments

In this section, we first describe the benchmark datasets and evaluation setup. Then, we discuss implementation details and comparisons of our results with state-of-the-art methods. Finally, we analyze different components in our algorithm.

4.1. Datasets and Evaluation

We evaluate our method on two popular action localization datasets, namely THUMOS14 [18] and ActivityNet1.2 [4], both of which contain untrimmed videos (i.e., there are many frames in the videos that do not contain any action).

The THUMOS14 dataset has 101 classes for action recognition and 20 classes for temporal localization. As in the literature [25, 32, 27], we use 200 videos in the validation set for training and 213 videos in the testing set for evaluation. Though this dataset is smaller than ActivityNet1.2, it is challenging since some videos are relatively long, and it has on average around 15.5 activity segments per video. The length of activity also varies significantly, ranging from less than a second to minutes.

The ActivityNet1.2 dataset has 100 activity classes consisting of 4,819 videos for training, 2,383 videos for validation, and 2,480 videos for testing (whose labels are withheld). Following [42], we train our model on the training set and test on the validation set.

We use the standard evaluation metric based on mean Average Precision (mAP) at different intersection over union (IoU) thresholds for temporal localization. Specifically, given the testing videos, our model outputs a ranked list of localization predictions, each of which consists of an activity category, start time, end time, and confidence score for that activity. If a prediction has correct activity class and significant overlap with a ground truth segment (based on the IoU threshold), then the prediction is considered to be correct; otherwise, it is regarded as a false positive.

4.2. Implementation Details

We first sample a maximum of 300 segments of a video, where each segment contains 16 frames with no overlap. If the video contains more than 300 segments, we sample 300 segments from the video randomly. Following [25], we use a two-stream I3D network to extract features from each stream (RGB and flow), and obtain 2048-dimensional feature vectors by concatenating both streams. The total loss function in our model is:

$$\mathcal{L} = \mathcal{L}_{\text{BBCE}} + \lambda \mathcal{L}_{\text{metric}} \tag{12}$$

We set $\lambda = 1$. We use $\alpha = 3$ in the metric loss function, block size $l_w = 60$, and $k = 10$ (Section 3.3). For the videos that have total segment length less than 60, we set l_w to be equal to the total segment length and k to be $\min(10, l_w)$. We use batch size 20 with 4 different activity instances per batch such that at least 5 videos have the same activity. The network is trained using the Adam optimizer [21] with learning rate 10^{-4}.

4.3. Comparisons with State-of-the-Art

We compare our result with state-of-the-art fully-supervised and weakly-supervised action localization methods on the THUMOS14 dataset in Table 1. Our method outperforms other approaches by a significant margin. In particular, it achieves 6.5% more mAP than the current best result at IoU threshold 0.5, and consistently performs better at other thresholds as well. Our approach even outperforms several fully-supervised methods, though we are not using any temporal information during training.

Table 2 shows our result on the ActivityNet1.2 validation set. Here, we see the performance is comparable with the state-of-the-art. We achieve state-of-the-art performance on IoU 0.1 and 0.3, and the results on other IoUs are very close to the current best results. Due to the significant difference between these two datasets, our algorithm does not produce as impressive results for ActivityNet1.2 as it does for THUMOS14 at all IoU thresholds. However, the THUMOS14 dataset has a large number of activity instances per

Table 1: Comparison of our algorithm with other state-of-the-art methods on the THUMOS14 dataset for temporal action localization.

Supervision	Method	IoU			
		0.1	0.3	0.5	0.7
Full	S-CNN [33]	47.7	36.3	19.0	5.3
	CDC [31]	-	40.1	23.3	7.9
	R-C3D [48]	54.5	44.8	28.9	-
	CBR-TS [12]	60.1	50.1	**31.0**	**9.9**
	SSN [50]	**60.3**	**50.6**	29.1	-
Weak	Hide-and-Seek [36]	36.4	19.5	6.8	-
	UntrimmedNets [42]	44.4	28.2	13.7	-
	STPN [25]	52.0	35.5	16.9	4.3
	AutoLoc [32]	-	35.8	21.2	5.8
	W-TALC [27]	55.2	40.1	22.8	7.6
	Su et al. [38]	44.8	29.1	14.0	-
	Liu et al. [23]	57.4	41.2	23.1	7.0
	Zeng et al. [49]	57.6	38.9	20.5	-
	Ours	**62.3**	**46.8**	**29.6**	**9.7**

video (around 15 instances per video) compared to ActivityNet1.2 which has only 1.5 instances per video. Moreover, THUMOS14 contains around 71% background activity per video (compared to 36% in ActivityNet1.2). Due to the high concentration of activity instances and large background activity, we think THUMOS14 is a better dataset for evaluating the performance of weakly supervised action detection. Therefore, we will concentrate mostly on THUMOS14 for evaluating our algorithm.

Table 2: Comparison of our algorithm with other state-of-the-art methods on the ActivityNet1.2 validation set for temporal action localization.

Supervision	Method	IoU			
		0.1	0.3	0.5	0.7
Full	SSN [50]	-	-	41.3	30.4
Weak	UntrimmedNets [42]	-	-	7.4	3.9
	AutoLoc [32]	-	-	27.3	**17.5**
	W-TALC [27]	53.9	45.5	**37.0**	14.6
	Liu et al. [23]	-	-	36.8	-
	Ours	**60.5**	**48.4**	35.2	16.3

4.4. Ablation Study

In this section, we present ablation studies of several components of our algorithm. We use different values of hyperparameters that give the best result for each architectural change. We perform all the studies in this section using the THUMOS14 [18] dataset.

Choice of classification loss function. As discussed in Sec. 3.3, we use the balanced binary cross-entropy (BBCE) loss instead of binary cross-entropy (BCE) and softmax loss. Figure 2 presents the effectiveness of BBCE loss over other choices. The same block-based processing strategy for the classification module is also included in the experiment. Our intuition is that the BBCE loss gives equal im-

portance to both foreground activities and background activities, so it can solve the class imbalance problem in a video more accurately.
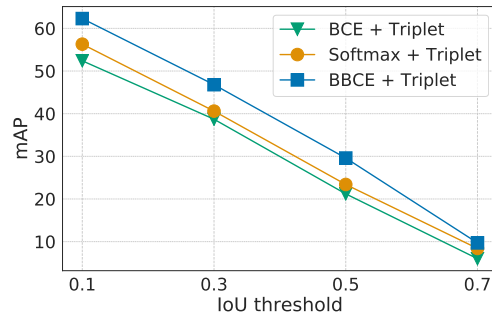

Figure 2: The mAP performance at different IoU thresholds on the THUMOS14 dataset for different classification loss functions. For the same metric loss function, BBCE performs better than BCE and Softmax loss. Here the Softmax loss is calculated according to the multiple-instance learning loss in [27].

Effect of metric learning module. To clarify, the goal of using a distance function here is to introduce an extra supervising target, which is especially useful in the weakly-supervised setting. In Table 3, we show the performance of our model without any metric loss, with contrastive metric loss, and with triplet loss, respectively. We see significant increases in the overall performance when metric loss is applied. In particular, the average mAP increases by 13.17% when the contrastive metric loss is applied and 13.32% when the triplet loss is applied.

Table 3: Experiments to show the effect of metric function on the THUMOS14 testing set for different IoU thresholds. Here, 'Avg' denotes the average mAP over IoU thresholds 0.1, 0.3, 0.5 and 0.7.

Method	IoU				
	0.1	0.3	0.5	0.7	Avg
Ours, $\mathcal{L}_{\text{BBCE}}$	48.7	29.3	14.0	3.1	23.78
Ours, $\mathcal{L}_{\text{BBCE}} + \mathcal{L}_{\text{Contrastive}}$	61.7	46.6	28.4	9.3	36.95
Ours, $\mathcal{L}_{\text{BBCE}} + \mathcal{L}_{\text{Triplet}}$	**62.3**	**46.8**	**29.6**	**9.7**	**37.10**

To validate the effectiveness of our proposed metric over other metric functions, we perform experiments by replacing our distance function with cosine distance, Euclidean distance, and a custom learnable distance function. For the custom distance function, we propose a learnable parameter $M \in \mathbb{R}^{C \times d \times d}$, which is updated through back-propagation, where C is the total number of classes, and set the metric $\mathbf{M}^c = M(c, :, :)$ in Eq. 11. Recall that when $\mathbf{M}^c = \mathbf{I}_d$, where $\mathbf{I}_d$ is the d-dimensional identity matrix, the metric function becomes the Euclidean distance function. In Fig. 3, we present the results for different distance functions. From

the figure, we see that the performances of cosine distance and Euclidean distance are quite similar, and the custom distance performs better than both of them since it has learnable parameters. However, our distance metric consistently performs the best at all IoU thresholds. In our algorithm, we are using a Mahalanobis type distance function, and the metric in the distance function comes from the weights of the classification module. Although the custom metric has the capability, at least in theory, to learn the same metric as our proposed distance function, the direct coupling between the classification module and the metric learning module creates an extra boost in our algorithm that improves the performance.
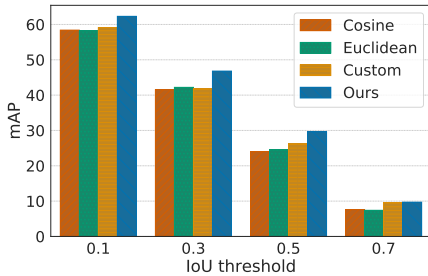

Figure 3: Performance comparison on the same dataset for different distance functions. Our metric performs better than the cosine distance, Euclidean distance, and a custom learnable distance.

Effect of block-based processing. We adopt a block-based processing strategy in the classification module to compute the classification score. In Table 4, we show the performance without block-based processing, i.e., when there is only one block for the whole video. From the experiment, we infer that block-based processing can handle variable length video more effectively. We still achieve superior performance compared to the current state-of-the-art without any block-based processing, mostly due to the metric learning module.

Table 4: The mAP performance at different IoU thresholds on the THUMOS14 dataset without any block-based processing in the classification module.

IoU	0.1	0.3	0.5	0.7
mAP	59.0	43.2	25.5	7.9

Effect of block size and k value. The block size l_w and value of k for k-max class activation are important parameters in our model (see Sec. 3.3). The value of k determines how many segments should be considered in each block to calculate the class score. From Fig. 4a, we see that at $k = 10$ for block size 60, we get the highest average mAP.

As k increases or decreases, the performance degrades. The reason is that at lower k, noisy segments can corrupt the classification score, and at higher k, the model cannot detect very short-range action instances properly. Fig. 4b illustrates the effect of block size l_w on the final performance. Here, we again see that there is a trade-off for the value of l_w, and we get the best performance at around $l_w = 60$.
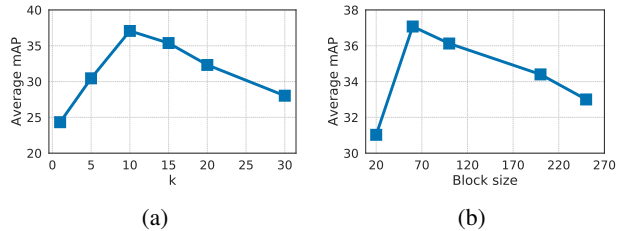

(a)   (b)

Figure 4: (a) The effect of k for a fixed block size 60 on average mAP. (b) Variations of average mAP for different values of block size (here, k is 6% of the block size). The average mAP is calculated by averaging the mAPs for IoU thresholds 0.1, 0.3, 0.5, and 0.7.

Ablation on clipping threshold. Through experiments, we found that applying a clipping function $\varphi_\kappa(\cdot)$ increases the performance. In Table 5, we show the mAP performance for different values of clipping thresholds κ, where 'w/o clip' denotes the model where no clipping function $\varphi_\kappa(\cdot)$ is applied (or the threshold κ is set to infinity). In particular, we obtain 2.5% mAP improvement at IoU threshold 0.5 for $\kappa = 4$ over no clipping.

Table 5: Experiments on clipping value κ

Clipping value κ	IoU			
	0.1	0.3	0.5	0.7
w/o clip	60.3	45.0	27.1	9.2
2	60.5	45.4	26.8	9.3
3	61.8	46.2	28.7	9.4
4	**62.3**	**46.8**	**29.6**	**9.7**
5	61.1	46.3	28.0	9.4
10	62.1	46.1	27.6	8.7

Qualitative results. Figure 5 represents qualitative results on some videos from THUMOS14. In Fig. 5a, there are many occurrences of the Hammer Throw activity, and due to the variation in background scene in the same video, it is quite challenging to localize all the actions. We see that our method still performs quite well in this scenario. In Fig. 5b, the video contains several instances of the Long Jump activity. Our method can localize most of them effectively. Our method also localizes most activities in Fig. 5c fairly well. Fig. 5d shows an example where our algorithm performs poorly. In Fig. 5d, there are several cases where the person swings the golf club or prepares to swing, but does not hit the ball. It is very challenging to differentiate

(a) Hammer Throw

(b) Long Jump

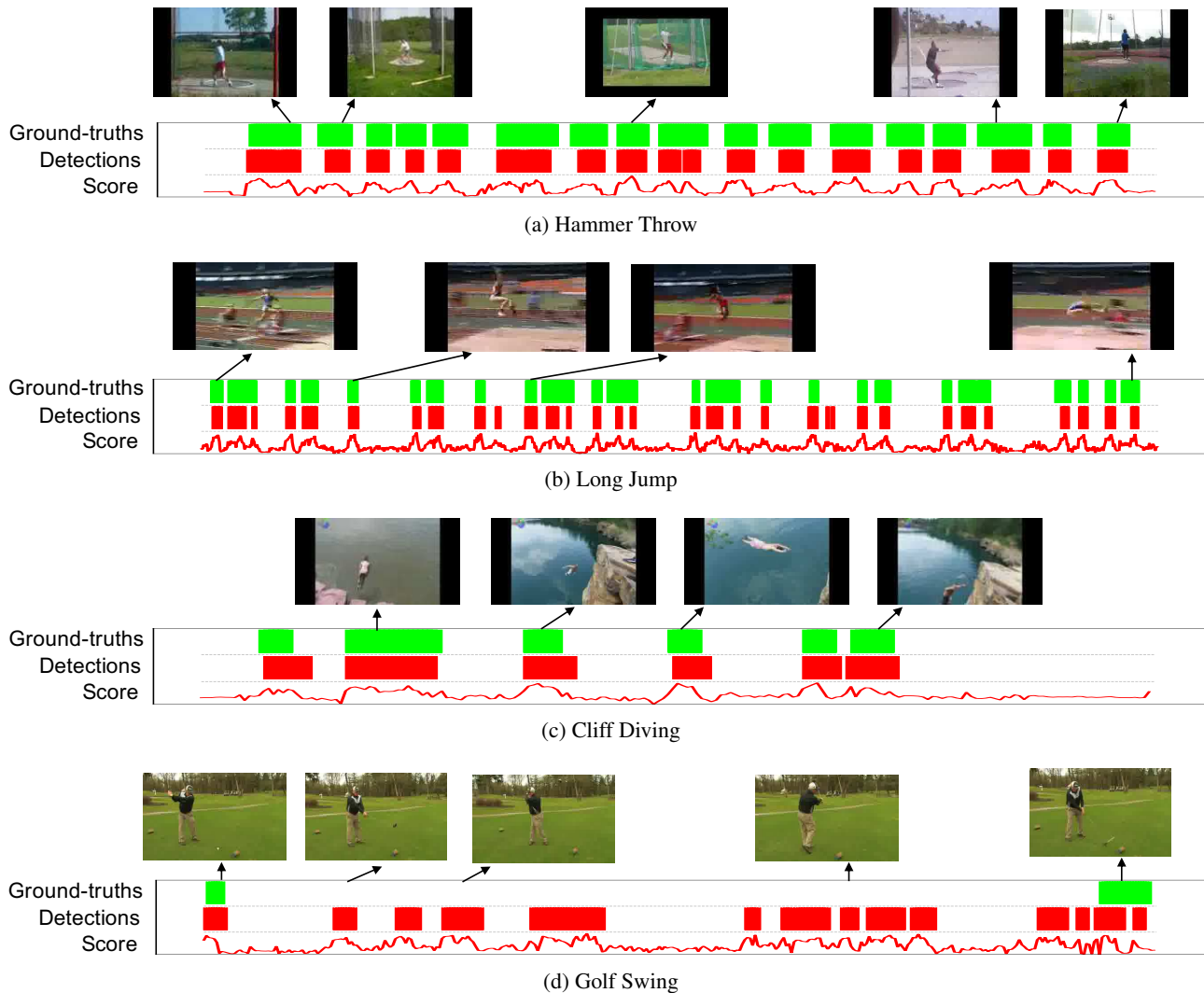(c) Cliff Diving

(d) Golf Swing

Figure 5: Qualitative results on THUMOS14. The horizontal axis denotes time. On the vertical axis, we sequentially plot the ground truth detection, detection score after post-processing, and class activation score for a particular activity. (d) represents a failure case for our method. In (d), there are several false alarms where the person actually swings the golf club, but does not hit the ball.

actual Golf Swing and fake Golf Swing without any ground truth localization information. Despite several false alarms, our model still detects the relevant time-stamps in the video.

rectly into the end-to-end model.

5. Conclusions and Future Work

We presented a weakly-supervised temporal action localization algorithm that predicts action boundaries in a video without any temporal annotation during training. Our approach achieves state-of-the-art results on THUMOS14, and competitive performance on ActivityNet1.2. For action boundary prediction, we currently rely on thresholding in the post-processing step. In the future, we would like to extend our work to incorporate the post-processing step di-

6. Acknowledgement

References

[1] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Trans. Graph.*, 34:98:1–98:10, 2015.

[2] H. Bilen and A. Vedaldi. Weakly supervised deep detection networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2846–2854, 2016.

[3] P. Bojanowski, F. R. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Finding actors and actions in movies. *2013 IEEE International Conference on Computer Vision*, pages 2280–2287, 2013.

[4] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.

[5] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the Kinetics dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.

[6] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar. Rethinking the faster R-CNN architecture for temporal action localization. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1130–1139, 2018.

[7] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1:539–546 vol. 1, 2005.

[8] R. G. Cinbis, J. J. Verbeek, and C. Schmid. Weakly supervised object localization with multi-fold multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:189–203, 2017.

[9] A. Dave, O. Russakovsky, and D. Ramanan. Predictive-corrective networks for action detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2067–2076, 2017.

[10] O. Duchenne, I. Laptev, J. Sivic, F. R. Bach, and J. Ponce. Automatic annotation of human actions in video. *2009 IEEE 12th International Conference on Computer Vision*, pages 1491–1498, 2009.

[11] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016.

[12] J. Gao, Z. Yang, and R. Nevatia. Cascaded boundary regression for temporal action detection. In *British Machine Vision Conference (BMVC)*, 2017.

[13] C. Gu, C. Sun, S. Vijayanarasimhan, C. Pantofaru, D. A. Ross, G. Toderici, Y. Li, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6047–6056, 2018.

[14] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2:1735–1742, 2006.

[15] S. Hong, D. Yeo, S. Kwak, H. Lee, and B. Han. Weakly supervised semantic segmentation using web-crawled videos. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2224–2232, 2017.

[16] J. Hu, J. Lu, and Y.-P. Tan. Discriminative deep metric learning for face verification in the wild. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1875–1882, 2014.

[17] D.-A. Huang, L. Fei-Fei, and J. C. Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *ECCV*, 2016.

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

[19] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, A. Natsev, M. Suleyman, and A. Zisserman. The Kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[20] A. Khoreva, R. Benenson, J. H. Hosang, M. Hein, and B. Schiele. Simple does it: Weakly supervised instance and semantic segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1665–1674, 2017.

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[22] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[23] D. Liu, T. Jiang, and Y. Wang. Completeness modeling and context separation for weakly supervised temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1298–1307, 2019.

[24] J. Lu, G. Wang, and P. Moulin. Image set classification using holistic multiple order statistics features and localized multi-kernel metric learning. *2013 IEEE International Conference on Computer Vision*, pages 329–336, 2013.

[25] P. Nguyen, T. Liu, G. Prasad, and B. Han. Weakly supervised action localization by sparse temporal pooling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6752–6761, 2018.

[26] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? - weakly-supervised learning with convolutional neural networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 685–694, 2015.

[27] S. Paul, S. Roy, and A. K. Roy-Chowdhury. W-TALC: Weakly-supervised temporal activity localization and classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 563–579, 2018.

[28] A. Richard, H. Kuehne, and J. Gall. Weakly supervised action learning with RNN based fine-to-coarse modeling. *2017*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1273–1282, 2017.

[29] K. Schindler and L. V. Gool. Action snippets: How many frames does human action recognition require? *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[30] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, June 2015.

[31] 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. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1417–1426, 2017.

[32] Z. Shou, H. Gao, L. Zhang, K. Miyazawa, and S.-F. Chang. AutoLoc: Weakly-supervised temporal action localization in untrimmed videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 154–171, 2018.

[33] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage CNNs. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1049–1058, 2016.

[34] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, 2016.

[35] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pages 568–576, 2014.

[36] K. K. Singh and Y. J. Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3544–3553, 2017.

[37] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[38] H. Su, X. Zhao, T. Lin, and H. Fei. Weakly supervised temporal action detection with shot-based temporal pooling network. In *International Conference on Neural Information Processing*, pages 426–436. Springer, 2018.

[39] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[40] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[41] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:1510–1517, 2018.

[42] L. Wang, Y. Xiong, D. Lin, and L. Van Gool. Untrimmed-Nets for weakly supervised action recognition and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4325–4334, 2017.

[43] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[44] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers. An improved algorithm for TV-L1 optical flow. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, 2008.

[45] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[46] S. Xie and Z. Tu. Holistically-nested edge detection. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1395–1403, 2015.

[47] F. Xiong, M. Gou, O. I. Camps, and M. Sznaier. Person re-identification using kernel-based metric learning methods. In *ECCV*, 2014.

[48] H. Xu, A. Das, and K. Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5794–5803, 2017.

[49] R. Zeng, C. Gan, P. Chen, W. Huang, Q. Wu, and M. Tan. Breaking winner-takes-all: Iterative-winners-out networks for weakly supervised temporal action localization. *IEEE Transactions on Image Processing*, 28(12):5797–5808, 2019.

[50] Y. S. Zhao, Y. Xiong, L. Wang, Z. Wu, D. Lin, and X. Tang. Temporal action detection with structured segment networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2933–2942, 2017.

[51] B. Zhong, H. Yao, S. Chen, R. Ji, X.-T. Yuan, S. Liu, and W. Gao. Visual tracking via weakly supervised learning from multiple imperfect oracles. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1323–1330, 2010.