ActionBytes: Learning from Trimmed Videos to Localize Actions

Mihir Jain[1*], Amir Ghodrati[1*], Cees G. M. Snoek[2]
[1]Qualcomm AI Research[†], Qualcomm Technologies Netherlands B.V.
[2]QUVA Lab, University of Amsterdam
mijain@qti.qualcomm.com ghodrati@qti.qualcomm.com cgmsnoek@uva.nl

Abstract

This paper tackles the problem of localizing actions in long untrimmed videos. Different from existing works, which all use annotated untrimmed videos during training, we learn only from short trimmed videos. This enables learning from large-scale datasets originally designed for action classification. We propose a method to train an action localization network that segments a video into interpretable fragments, we call ActionBytes. Our method jointly learns to cluster ActionBytes and trains the localization network using the cluster assignments as pseudo-labels. By doing so, we train on short trimmed videos that become untrimmed for ActionBytes. In isolation, or when merged, the ActionBytes also serve as effective action proposals. Experiments demonstrate that our boundary-guided training generalizes to unknown action classes and localizes actions in long videos of Thumos14, MultiThumos, and ActivityNet1.2. Furthermore, we show the advantage of ActionBytes for zero-shot localization as well as traditional weakly supervised localization, that train on long videos, to achieve state-of-the-art results.

1. Introduction

The goal of this paper is to determine the start, the end and the class of each action instance in a long untrimmed video. State-of-the-art approaches for action localization slide a trained model over an untrimmed video to produce classification score sequences over time [5, 18, 40]. They depend on start, end, and action class labels at training time. Weakly-supervised approaches [28, 32, 34] have demonstrated this even works when the long untrimmed training videos come with action class labels only. Different from all these works, we will localize action instances in long untrimmed videos by learning from short trimmed videos labeled with just their action class.

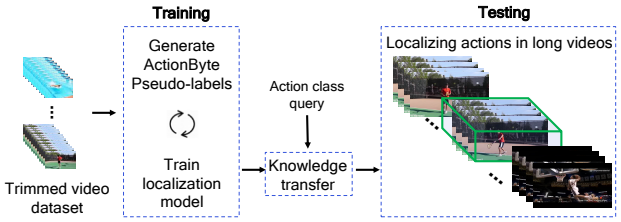

Figure 1: **From short, trimmed videos we learn to localize actions in long, untrimmed video.** During training, our method jointly learns to generate pseudo-labels from ActionBytes, and to localize them in the short video. During testing, our localization model detects the instances of the query action class in the untrimmed video.

Short trimmed videos are highly popular and easy to access for action classification. Datasets in this domain come with a large number of samples and labels [3, 4, 8, 24]. Kinetics-700 [3], for example, has nearly 650k short trimmed video clips categorized into as many as 700 action classes. In this work, we leverage datasets commonly used for action classification, the *what*, for tackling the task of action localization, the *when*. This opens up opportunities for 1) learning from larger datasets with more action classes, and 2) localizing unknown classes by transferring knowledge between trimmed and untrimmed video datasets.

However, given just short trimmed videos during training provides virtually no scope to learn about action boundaries. To overcome this limitation, we adopt a self-supervised approach to regularize our network to learn boundary-aware models. Specifically, we use intermediate layers of a CNN model to decompose a trimmed video into multiple atomic actions called *ActionBytes*. From these we generate pseudo-labels to train a CNN to localize ActionBytes within videos. This model can be used to extract a new set of ActionBytes, so we iterate between updating ActionBytes and training the localization model using new pseudo-labels. Given a long test video, we slide our trained model over it to generate a classification score sequence for the query action, and thus localize its instances, see Figure 1.

[*] equal contribution
[†] Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc

We make three contributions in this paper. First, we define *What2When*, the task of localizing actions in long untrimmed videos using short trimmed videos commonly used for action classification. Second, we introduce ActionBytes: interpretable, temporally scale-invariant fragments of videos capable of spotting parts of an action. Third, we propose an iterative approach for training boundary-aware models from short videos. We experimentally show the effectiveness of our method on Thumos14 [10], MultiThumos [38] and AcitivityNet [1]. Since our approach transfers action class knowledge from trimmed videos to untrimmed videos with unseen classes, it is a natural fit for zero-shot applications. We evaluate our model in a zero-shot scenario where the label set of the short trimmed training videos and the long untrimmed test videos are disjoint. Finally, we conduct experiments on the task of weakly supervised action localization. Although our method is not designed for learning from long videos, we show the benefit of ActionBytes as action proposals in obtaining favorable performance compared to the state-of-the-art.

2. Related work

The problem of learning from short videos to localize action in long videos relates to multiple recognition tasks in videos.

Mid-level representation. Several works have proposed methods to automatically discover mid-level representations by segmenting an action into atomic actions [9, 15, 33]. Lan *et al*. [15] discover mid-level action elements by clustering spatio-temporal segments. This is done on a per-class basis. In [6, 9] the authors automatically obtain meaningful action fragments but they require temporal action annotations to do so. Alternatively, [39] also uses parts of actions but exploits their ordered fashion. Unlike all the above methods, our ActionBytes are class-agnostic, which makes them suitable to enable knowledge transfer to videos of unseen classes.

Pseudo-labeling. Recently self-supervised approaches have been proposed pseudo-labeling data in representation learning [2], label propagation for semi-supervised learning [11] and semantic segmentation [16]. This line of work relies on clustering to create pseudo-labels from unlabelled data. We also generate pseudo-labels per video during training, but for a different purpose, we use them to regularize our localization model to be sensitive to boundaries.

Self-training. Our approach can also be considered as a self-training procedure applied to the video domain, and adapted for localization in the What2When task. It differs from other self-training approaches [17, 29, 42] in many ways, but mainly because the pseudo-labels are generated at the sub-video level and are regularized for localization.

Weakly supervised. In recent times, there has been increased interest in developing models that can be trained with weaker forms of supervision, such as video-level labels. UntrimmedNets [34] and STPN [26] formulated weakly supervised action localization as a multiple instance learning problem along with attention to locate the actions in videos. AutoLoc [32] introduced a boundary predictor built on an Outer-Inner-Constrastive loss. W-TALC [28] introduced a co-activity similarity loss that looks for similar temporal regions in a pair of videos containing a common action class. Nguyen *et al*. [27] proposes to model both foreground and background, while [39] exploits temporal relations among video segments. All these methods depend on the presence of multiple actions in long videos to learn to discriminate foreground action from background. Differently, we propose a method to learn action boundaries from short videos through our ActionByte mining.

Zero-shot learning. Many approaches for zero-shot and few shot learning focus on intra-dataset splits between seen and unseen classes [7, 14, 19, 37]. While others attempt cross-dataset action recognition [12, 20, 41] and some of those learn only from the image domain to recognize actions in videos [12, 20]. To avoid the use of common classes across datasets, Roitberg *et al*. [30] present an evaluation by filtering very similar classes between source and target. The common practice in zero-shot learning is to transfer action class knowledge through a semantic embedding space, such as attributes, word vectors or visual features. Among these, word vectors have been preferred as only category names are required for constructing the semantic embedding space. In this paper, we also employ word embeddings to map source classes to target classes while precisely following the zero-shot regime.

3. Method

In this section, we explain our proposed method that learns from short trimmed videos to temporally localize actions in long untrimmed videos. We first formally define the problem of What2When action localization. Then, we explain our method illustrated in Figure 2 and its components. We start by introducing ActionBytes, the basic building block of our method and give an explanation on how to extract them from videos. Next, we explain our two-step iterative pipeline that leverages ActionBytes to train localization models on short videos in a self-training fashion. Finally, we discuss the potential of ActionBytes by itself as action proposals in the video localization context.

Problem statement. Given a long test video, we aim to predict a set of action categories present in that video, together with their start and end time. During training, a set of n short, single-action videos $\chi^{short} = \{x_i\}_{i=1}^{n}$ is given where each video x has a single label c, belonging to label set $C_{short} = \{c_i\}_{i=1}^{n_c}$. During testing, a set of long untrimmed videos $\chi^{long} = \{x'_i\}_{i=1}^{n'}$ is given, where for each video x', the goal is to find the boundary of all action in-

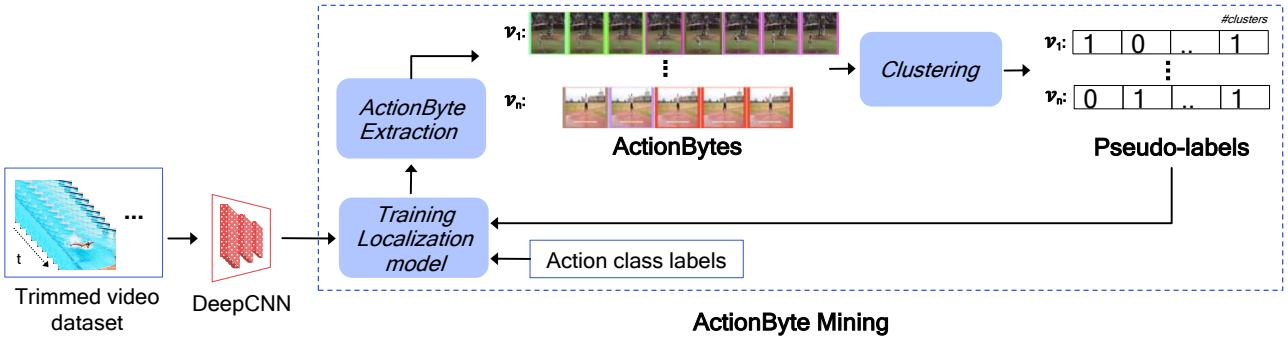

Figure 2: **The proposed mining pipeline** segments a video into ActionBytes. These are then clustered and assigned pseudo-labels, which are used as a supervision signal to train the localization model. Action classes labels are from C_{short}.

stances and predict their category labels, c', from the label set $C_{long} = \{c'_i\}_{i=1}^{n'_c}$. In this paper, unless explicitly otherwise stated, we train on χ^{short} and evaluate on χ^{long}.

3.1. ActionBytes

It is well-known that high-level features of consecutive frames, extracted from a CNN, usually vary smoothly over time [13, 35]. Therefore, any abrupt change in feature space can represent a high-level change in the pixel space. We leverage this property to segment videos into interpretable fragments, we call ActionBytes.

Suppose $F = \{\boldsymbol{f_t}\}_{t=1}^{T}$ are d-dimensional features extracted using a deep model for each time instant t, where T is the temporal sequence length. We learn to map these features to a latent space using a latent projection module. The output of the latent projection module, $L \in \mathbf{R}^{l \times T}$, keeps the affinity to l latent concepts for each time instant (Figure 3). For a given video, we find ActionByte boundaries B by looking for time instants where affinities to latent concepts change abruptly compared to the previous time instant:

$$B = \{t | t : \sum_{i=1}^{l} |L[i,t] - L[i,t-1]| > \tau\} \qquad (1)$$

where τ is set to the p^{th} percentile, so the number of ActionBytes in a video is directly proportional to its length T. In general, the p^{th} percentile leads to $T \times \frac{100-p}{100}$ ActionBytes. The length of each of them varies with the video content, with average length equal to $\frac{100}{100-p}$.

Each boundary in the set B starts an ActionByte, $A_i = (B_i, B_{i+1} - 1)$, resulting in $|B| - 1$ ActionBytes. Such boundaries are obtained in a class-agnostic way, but they segment a video into interpretable fragments. These ActionBytes are temporally scale-invariant as their lengths are adapted to the video content. For example, a single ActionByte can capture an atomic action regardless of the action speed. Some ActionBytes examples are shown in Figure 4.

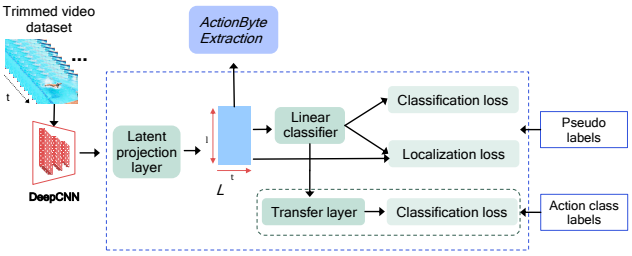

Figure 3: **Localization model and ActionByte extraction**. The localization model is trained with classification and localization losses on pseudo-labels. The latent output L is used to extract ActionBytes. Classes labels are from C_{short}.

3.2. Mining ActionBytes

Next, we discuss how we learn a model from short videos. One can train a classification model on short videos and slide it on long test videos. However, such a model is agnostic to boundaries within the short videos, and might not be able to generate good class activation scores for localization. Here, we leverage ActionBytes, to train a discriminative, boundary-aware model from short videos. This is done by decomposing a video into multiple ActionBytes, from which we generate pseudo-labels to train our model.

The proposed pipeline for mining ActionBytes is shown in Figure 2. It has two steps that iterates between *Generating pseudo-labels from ActionBytes* and *Training the localization model with pseudo-labels*. For the creation of the pseudo-labels we take inspiration from Caron *et al.* [2]. We first extract N ActionBytes from a set of training videos and represent each of them by averaging latent features within its boundaries. Next, we group all the ActionBytes into K clusters using the k-means algorithm by solving

$$\min_{C \in \mathbb{R}^{l \times K}} \frac{1}{N} \sum_{n=1}^{N} \min_{y_n \in \{0,1\}^K} \|a_n - C y_n\|_2^2$$

Figure 4: **Extracted ActionBytes**, highlighted in different colors, for two examples of *Baseball Pitch*. The ActionBytes capture the action in four parts that are interpretable as (1) 'get into wind-up position' (red), (2) 'loading to deliver' (blue), (3) 'delivery' (pink) and (4) 'follow-through' (green). ActionBytes are scale-invariant and can adapt to varying temporal scale, *e.g.*, the 'follow-through' extends to different number of snippets in the two examples.

where a_n is feature vector obtained from ActionByte n. Solving this problem provides a centroid matrix C that is used to assign a cluster id to each ActionByte in a video. Finally, the pseudo-label vector for a video is defined as all the cluster ids assigned to ActionBytes of that video.

Having obtained the multiple pseudo-labels for each training video, we update the parameters of the localization network in the second step for classifying and localizing ActionBytes in the video (shown in Figure 3). Such training leads to a better representation of latent concepts, L, of the model that in turn result in a better set of ActionBytes. Therefore, we iterate over these two steps of extracting ActionBytes and training the localization model. This approach can be seen as a regularization technique. By training the model with pseudo-labels, we avoid the risk of overfitting the model to class labels.

Localization model. Our full localization model, used in the second step of our pipeline, is shown in Figure 3. The role of this model is to learn to classify and localize ActionBytes into the assigned pseudo-labels. This is reminiscent of a model for weakly-supervised temporal localization, where each video has multiple instances of actions and temporal annotations are not available. With this motivation, we now describe our localization.

We first extract features $F = \{\boldsymbol{f_t}\}_{t=1}^{T}$ from a pretrained deep network where d is the feature dimension and T is the temporal sequence length. We pass extracted features to a latent projection module to map the features to a set of latent concepts, from which we extract ActionBytes. For the latent projection module, we simply use a fully connected layer followed by ReLU [25].

$$L = ReLU(W_{proj}F)$$

where $W_{proj} \in \mathbf{R}^{l \times d}$ is the latent projection matrix and l is number of latent concepts. The output of the latent projection layer, L, is passed through a linear classifier to obtain activation scores over time for pseudo-classes. On these activation sequences, following [28], we apply k-max multiple-instance learning loss for classification and co-activity similarity loss for localization. For k-max MIL loss, the prediction score corresponding to a class is computed as the average of its k-max activations over the temporal dimension. The co-activity similarity loss is computed over class activation sequences and L. For a given video and a class, a vector of similarities between class activation sequence and each row of L (l^{th} latent concept) is computed. A pair of videos with a common class label will have higher similarities with the same latent concepts. This is what is enforced by this loss, which makes it a suitable localization loss in our method.

Using this model in our mining, we get predictions for the pseudo-labels. In order to translate this into predictions for the training classes, C_{short}, we add a *transfer layer* on top of the linear classifier. This is an FC layer learned again with a k-max MIL loss, but using class labels (see Figure 3). For localization at test time, we follow the two-stage thresholding scheme of [28] on the output of the transfer layer.

Knowledge transfer. In cross-dataset evaluation, the label set of seen short videos, C_{short} can be different from the label set of unseen long videos, C_{long}. For knowledge transfer in such cases, we follow Objects2Action [12]. We employ the skip-gram model of word2vec [21, 22] as a semantic embedding function to embed each word of a given class label as a vector. For multi-word class labels, we take the average vector of the embedded words [12, 23] to represent the label. The affinities between class labels from C_{short} and C_{long} are computed by cosine similarity between their embeddings. Thus, the class activation score for C_{short} is transferred to that for C_{long}.

The two sets of class-labels, though different, may have some overlap. To evaluate in a pure zero-shot localization set-up, we also conduct an experiment where training is done on a subset of C_{short}, such that this subset does not overlap with test label set C_{long}.

3.3. Action proposals from ActionBytes

Segmenting video into ActionBytes is critical to learn a reliable localization model from short videos. In addition

to this, ActionByte by itself is also suited for action localization as an informative action unit. We show how they can be used to form action proposals in long videos during testing. Consequently, we also demonstrate the utility of ActionBytes is not limited to the What2When set-up but also extends to the weakly-supervised set-up.

Since an ActionByte represents an interpretable part of an action, one or more ActionBytes together form a good action proposal. For a given test video, we generate action proposals, P_{AB}, by merging $m \in M$ ActionBytes, where set M contains the numbers of ActionBytes to be merged.

$$P_{AB} = \bigcup_{m \in M} \bigcup_{i=1}^{|B|-m} (B_i, B_{i+m} - 1) \tag{2}$$

where B_i is the start of ActionByte i. $(B_i, B_{i+m} - 1)$ is an action proposal from B_i to $B_{i+m} - 1$. Each of these proposals is temporally jittered to include up to one neighboring time-step. This is to make sure the immediate neighborhood of boundaries is included in the action proposals.

ActionBytes for weakly-supervised localization. Weakly-supervised action localization is a popular task where training and testing are done on long videos *i.e.* $L_{short} = L_{long}$. The ActionByte mining explained in Section 3.2 is critical to learn from short videos. But, when learning on long videos in a weakly-supervised set-up, generating pseudo-labels is not needed, as the long videos are already untrimmed w.r.t. the actual action labels. Therefore, only the localization model, without the transfer layer, is enough to learn good quality classification score sequences and ActionBytes.

4. Experiments

In this section, we first explain the datasets we train and evaluate our proposed method on, following the implementation details. Then we present an ablation study of our method, and next we compare our model with baselines in the What2When setup. We also conduct an experiment in a zero-shot setup and compare our model with the state-of-the-art models in the weakly-supervised regime.

Datasets. We use the validation set of **Kinetics-400** [4] for training our model. It contains $17,281$ single trimmed action videos belonging to 400 action classes with a maximum length of 10 seconds. For evaluation, we report on the untrimmed **Thumos14** [10], **MultiThumos** [38] and **ActivityNet1.2** [1]. Thumos14 contains 200 validation videos and 212 test videos with temporal annotations belonging to 20 action classes, with about 15.5 action instances per video on average. The length of the videos in this dataset is on average 212 seconds. MultiThumos has the same set of videos as in Thumos14, but it extends the latter from 20 action classes with 0.3 labels per frame to 65 classes with 1.5 labels per frame. Also, the average number of distinct action classes in a video is 10.5 (compared to 1.1 in Thumos14), making it a more challenging multi-label dataset. ActivityNet1.2 has $4,819$ videos for training and $2,383$ videos for validation, which in the literature is used for evaluation. It has 100 classes, with on an average 1.5 action instances per video. The average length of the videos in this dataset is 115 seconds.

Implementation details. As a base network we use I3D [4] pretrained on Kinetics-400. We extract RGB and flow features from the last average-pooled layer (1024 dimensions for each stream). We use TVL1 to compute optical flow. Features are extracted from non-overlapping 16-frame chunks of video. We do not finetune the feature extractors. The network is implemented in PyTorch and trained with Adam optimizer with a learning rate of 0.001. We initialize the localization model by training on the validation set of Kinetics-400 dataset. For k-max MIL loss, we set k to $1/8$ of the length of the video. In all the experiments, we iterate over our pipeline for 3 iterations. The value of the p percentile (sets τ in Eq. 1) determines how many ActionBytes are extracted from a given video. For Thumos14 and MultiThumos we set $p = 50$, and for ActivityNet1.2 we use $p \in \{92, 95, 97.5, 99, 99.5\}$. In all the experiments we set $M = \{1, 2\}$ in Eq. 2. We report the commonly used mean Average Precision (mAP) metric on snippet-level granularity for evaluating detections. For the weakly-supervised setup, experiment settings are kept similar to [28].

Localization at test time. For localization at test time, we use our trained model to generate class-activation sequences over the untrimmed test video. We follow the two-stage thresholding scheme of [28] for localizing actions. The first threshold is applied to filter out classes that have confidence score less than the mean confidence score. The second threshold is applied along the temporal axis to obtain the detections. When ActionByte proposals are added, non-maximum suppression is also applied.

4.1. Ablation study

In the ablation, we test on untrimmed Thumos14, and train on the validation set of trimmed Kinetics-400 dataset.

Fixed length versus scale-invariant ActionBytes. First, we evaluate the effect of ActionBytes. We run two setups: the first uses fixed-size segments, uniformly sampled along the video, and the second uses our automatically-extracted ActionByte boundaries. For the first setup we uniformly segment the video into chunks of two snippets, in order to make it comparable with the average length of ActionBytes. The final localization performance at $IoU = 0.5$ is 14.1% for fixed-size segments and 15.5% for ActionBytes. Automatically extracted ActionByte boundaries are preferred over uniformly sampled boundaries.

Influence of number of clusters. Next, we evaluate the

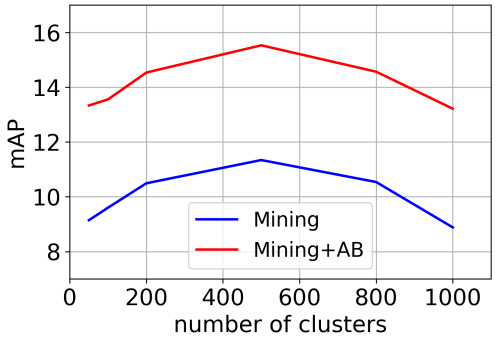

Figure 5: **Influence of the number of clusters** on localization performance. The performance increases up to 500 and decreases afterward, as over-granular clusters might not be able to represent a single ActionByte.

influence of the number of clusters for generating pseudo-labels on the final localization performance. Figure 5 shows that the performance increases by increasing the number of clusters up to 500 and then decreases. This makes sense as with a large number of clusters, an ActionByte might not be represented by a single cluster centroid. Therefore, during all the experiments, we fix the number of clusters to 500.

Number of mining iterations. In Figure 6 *(Left)*, we show how performance changes over training iterations. It increases up to a point, and then decreases slightly. This is mainly because, after few epochs, our iterative mining reaches an equilibrium point where the clustering loss stops decreasing (see Figure 6 *(Right)*) and the model converges to an optimum.

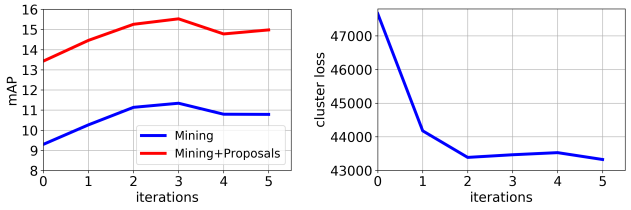

Figure 6: **Iterative mining.** *(Left)* Action localization mAP over mining iterations. Performance increases as long as the clustering loss *(Right)* decreases, then both get saturated.

ActionByte as proposals. As explained in Section 3.3, ActionBytes, when merged together, can act as action proposals. In this ablation, we show how the number of merged ActionBytes influences localization performance. As shown in Figure 7, using single ActionByte proposals ($M = \{1\}$) can improve the performance by more than 3% compared to not using ActionByte proposals. This shows the effectiveness of ActionBytes as proposals. Merging up to 4 ActionBytes ($M = \{1, 2, 3, 4\}$) can improve localization performance further. However, it comes with the cost

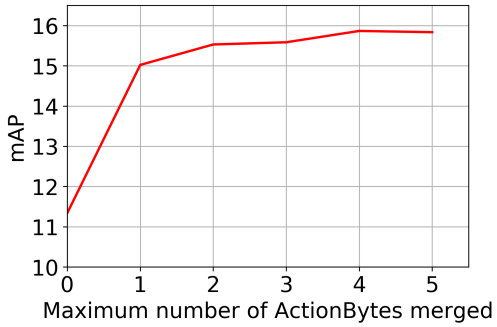

Figure 7: **ActionByte as proposals** for localization. Single ActionByte proposals ($M = \{1\}$) improve mAP compared to not using ActionByte proposals. We set $M = \{1, 2\}$ in all the experiments as adding more proposals increases the computational cost while bringing marginal improvement.

of processing more proposals. To keep the balance between computational cost and performance, we set $M = \{1, 2\}$ in the remaining experiments. Since the ActionBytes vary in length, the proposal length also varies. This is reminiscent of commonly used anchor lengths [32]. The proposal length, for chosen M and p, ranges from 1 to 70 for Thumos14/MultiThumos and from 6 to 369 for ActivityNet.

4.2. What2When action localization

In the What2When action localization experiments, we show the benefit of our mined ActionBytes compared to the baseline. For training, we use the validation set of Kinetics-400 dataset. For evaluation, we follow the common protocol from the literature and evaluate on the test sets of Thumos14 and MultiThumos, and validation set of Activitynet1.2. **Baseline** is the localization model trained on the Kinetics-400 validation set, without ActionBytes and iterative training. This model generates confidence scores for 400 classes over untrimmed long videos. Then we transfer the class scores to target classes as explained in Section 3.2, and localize actions using the two-stage thresholding. **Ours** is our proposed deep mining method, that is similar to the baseline (and trained on the same dataset) except that we use pseudo-labels during training to regularize the model. To have a fair comparison, we keep all the hyper-parameters fixed during evaluation. Finally, for **Ours (+ Proposals)** we add ActionByte proposals to the pool of proposals during localization.

As shown in Table 1, the baseline performance on Thumos14 dataset for $IoU = 0.5$ is 8.4% which shows the difficulty of the task. Using our model, the performance increases to 11.3%. This is interesting, considering that the state-of-the-art performance for this dataset for the weakly-supervised regime where training and test is done on the same dataset is just 26.5% [27] (see Table 3). Finally, by

Table 1: **What2When action localization** performance on Thumos14, ActivityNet1.2 and MultiThumos.

	Thumos14				**ActivityNet1.2**				**MultiThumos**			
	0.3	0.4	0.5	0.7	0.3	0.4	0.5	0.7	0.3	0.4	0.5	0.7
Baseline	18.8	12.7	8.4	1.7	24.0	21.7	19.4	8.0	7.5	4.9	3.2	0.6
Ours	21.1	15.6	11.3	2.8	24.4	22.4	20.1	8.2	8.1	5.7	4.1	1.0
Ours (+ Proposals)	**26.1**	**20.3**	**15.5**	**3.7**	**24.7**	**22.7**	**20.3**	**8.3**	**10.8**	**8.1**	**6.1**	**1.4**

Table 2: **Zero-shot action localization** performance on Thumos14 and MultiThumos in What2When setup.

	0.1	0.2	0.3	0.4	0.5
Thumos14					
Baseline	13.8	11.1	7.1	4.7	3.1
Ours	14.9	12.6	8.5	6.1	4.1
Ours (+ Proposals)	**17.8**	**15.5**	**11.3**	**8.7**	**6.3**
MultiThumos					
Baseline	6.4	5.14	3.1	2.0	1.3
Ours	7.0	5.7	3.7	2.5	1.7
Ours (+ Proposals)	**9.4**	**8.0**	**5.6**	**4.1**	**3.0**

adding ActionByte proposals, the performance increases to 15.5% *i.e.* an 84% relative improvement overall. This also shows the effectiveness of our ActionBytes as proposals, which is mainly due to their complementary nature to the baseline proposals. The improvements are obtained across the *IoUs*, especially for the higher ones.

For ActivityNet1.2 the baseline obtains an mAP of 19.4% at $IoU=0.5$, while our full model gets to 20.3%. The gains are less compared to Thumos14 but consistent across the *IoUs*. The reduced gains can be attributed to the nature of temporal annotations, which merge several nearby action instances and in-between pauses into one instance. This meant extra false-positives, as ActionByte proposals do well at separating actions from temporal context.

For results on MultiThumos the trend is similar to Thumos14, mining and then ActionByte proposals consistently improve performance across the *IoU* thresholds. It is promising that the proposed method maintains its gain on this more challenging multi-label dataset.

4.3. Zero-shot action localization

For this set of experiments, we have a similar setup to the previous What2When experiment, except that we adhere to a zero-shot premise and exclude common classes between the source Kinetics-400 dataset and the target datasets. Thus, during training, we exclude 18 classes of Kinetics-400 for Thumos14/MultiThumos. Similarly, 72 classes of Kinetics-400 are excluded for ActivityNet1.2, which leaves classes that are semantically very different from those of ActivityNet1.2. The remaining classes are semantically very different from those of ActivityNet1.2, resulting in a much lower baseline mAP of 2.6% at $IoU=0.3$ compared to 24.0% in the What2When experiment. As ActivityNet1.2 is not suitable for zero-shot transfer from Kinetics-400, we evaluate on the other two datasets in Table 2. Compared to the What2When results there is a drop in performance, which is expected, considering the difficulty of the task. However, the same trend is maintained: our mining model performs better than the baseline and adding ActionByte proposals further adds to the localization performance. Again, we observe considerable gains over the baseline for both Thumos14 and MultiThumos, leading to consistent improvement across the *IoUs*. We believe that these are the first zero-shot temporal localization results reported on Thumos14 and MultiThumos.

Table 3: **Weakly-supervised localization** on Thumos14 dataset. (*) indicates I3D features.

	0.3	0.4	0.5	0.7
Strong supervision				
Shou *et al.* [31]	40.1	29.4	23.3	7.9
Xu *et al.* [36]	44.8	35.6	28.9	-
Zhao *et al.* [40]	50.6	40.8	29.1	-
Chao *et al.* * [5]	53.2	48.5	42.8	20.8
Weak supervision				
Nguyen *et al.* * [26]	35.5	25.8	16.9	4.3
Shou *et al.* [32]	35.8	29.0	21.2	5.8
Paul *et al.* * [28]	40.1	31.1	22.8	7.6
Yu *et al.* * [39]	39.5	-	24.5	7.1
Nguyen *et al.* * [27]	**46.6**	**37.5**	26.5	9.0
*Ours** (Proposals)	43.0	35.8	**29.0**	**9.5**

4.4. Comparison with the state-of-the-art

Here, we demonstrate the effectiveness of our ActionByte proposals in a weakly-supervised setup as explained in Section 3.3. We employ the off-the-shelf model of Paul *et al.* [28] as baseline and add ActionBytes proposals on top of it. For the Thumos14 dataset, we train the model on the validation set and evaluate on the test set. Similar as before, we use *IoU* between detections and ground-truth as the evalua-

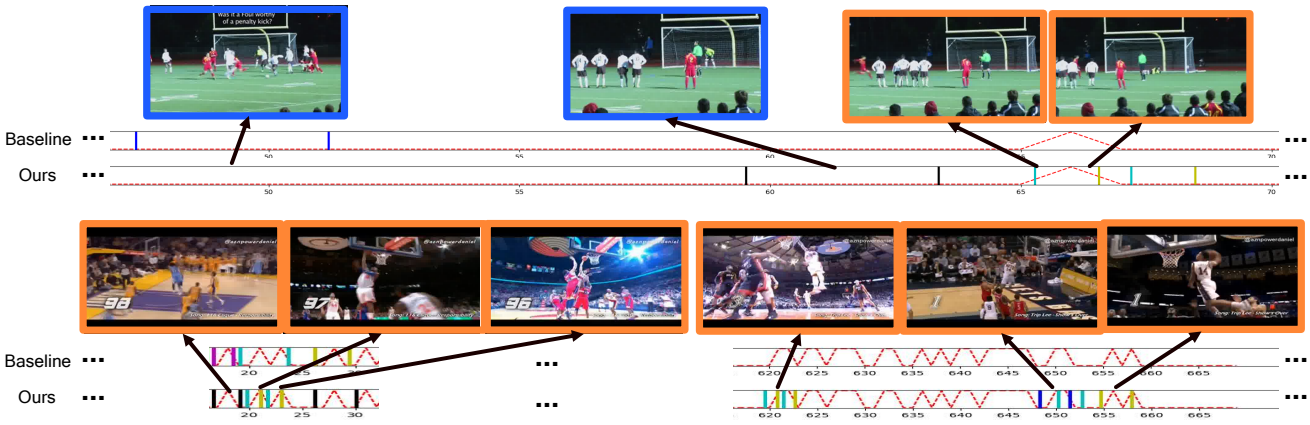

Figure 8: **Qualitative results** showing top localizations on sample videos from *Soccer Penalty* and *Basketball Dunk*. Frames representing action instances are highlighted by the orange boxes and the ones for the background are in blue boxes. Below these frames, ground-truth is plotted in red against time in seconds. Localization boundaries are shown in other colors for the baseline detections as well as the detections using the ActionByte proposals. In *Soccer Penalty* example, there is only one true-positive which is missed by the baseline, while it is populated by our proposals, one of which detects it. Both methods have false positives. The second example of *Basketball Dunk* is a video longer than 10 minutes, with many action instances. Out of shown 16 instances, our approach could localize 6 while getting 3 false-positives at $IoU = 0.5$. Two of these false-positives are duplicate detections (in cyan near 620s and 650s). The baseline could localize two action instances with one false-positive. There are a few false-positives and missed detection by our approach, but it could localize some very difficult action instances. Figure best viewed in color.

tion metric. As shown in Table 3, our method outperforms the state-of-the-art for higher overlap thresholds. Our improvement is particularly notable at $IoU = 0.5$, where we improve the state-of-the-art by a margin of 2.4%. It validates that our ActionByte proposals are suitable for both What2When and weakly supervised tasks. In Table 4, results on ActivityNet1.2 are reported. We outperform state-of-the-art for all *IoUs* except 0.7. In Table 5, we report results for MultiThumos. To our knowledge, the only video-level localization results reported on MultiThumos is by Yeung *et al*. [38]. While they report 32.4% at $IoU = 0.1$, with frame-level supervision, we reach this mAP with weak supervision only. To the best of our knowledge, this is the first weakly-supervised evaluation on MultiThumos. We also evaluate our baseline [28] on this dataset and consistently improve it over the *IoU* thresholds. In summary, our method could improve over the baselines and achieve promising results on all three datsets. This shows the effectiveness of the ActionByte proposals. We show some qualitative results of our detections in Figure 8.

Table 4: **Weakly-supervised localization** on ActivityNet1.2 dataset. (*) indicates I3D features.

	0.3	0.4	0.5	0.7
Wang *et al*. [34]	-	-	7.4	3.9
Shou *et al*. [32]	-	-	27.3	17.5
Paul *et al*. * [28]	45.5	41.6	37.0	14.6
Yu *et al*. * [39]	-	-	28.3	**18.9**
*Ours** (Proposals)	**47.8**	**44.0**	**39.4**	15.4

Table 5: **Weakly-supervised localization** on MultiThumos dataset. (*) indicates I3D features. †Our evaluation of [28].

	0.1	0.2	0.3	0.4	0.5
Strong supervision					
Yeung *et al*. [38]	32.4	-	-	-	-
Weak supervision					
Paul *et al*. *†	30.7	24.0	17.1	12.6	8.9
*Ours** (Proposals)	**32.4**	**26.8**	**20.5**	**15.7**	**12.1**

5. Conclusions

We introduced the new task of learning from short trimmed videos to localize actions in long untrimmed videos. To tackle the new task, our proposed pipeline is jointly trained to segment the videos into ActionBytes and localize them in the short video. Our method can be considered as a technique to regularize action boundaries during training. Experiments on the three datasets show the effectiveness of our method not only for the proposed task, but also for zero-shot action localization and weakly supervised action localization. This demonstrates the adaptability of the models trained by our method, as we considerably improve over the baselines and achieve state-of-the-art results.

References

[1] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015. 2, 5

[2] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018. 2, 3

[3] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019. 1

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

[5] Yu-Wei Chao, Sudheendra Vijayanarasimhan, Bryan Seybold, David A Ross, Jia Deng, and Rahul Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *CVPR*, 2018. 1, 7

[6] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Temporal localization of actions with actoms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2782–2795, 2013. 2

[7] Chuang Gan, Tianbao Yang, and Boqing Gong. Learning attributes equals multi-source domain generalization. In *CVPR*, 2016. 2

[8] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017. 1

[9] Rui Hou, Mubarak Shah, and Rahul Sukthankar. Real-time temporal action localization in untrimmed videos by sub-action discovery. In *BMVC*, 2017. 2

[10] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, 2017. 2, 5

[11] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *CVPR*, 2019. 2

[12] Mihir Jain, Jan C van Gemert, Thomas Mensink, and Cees GM Snoek. Objects2action: Classifying and localizing actions without any video example. In *ICCV*, 2015. 2, 4

[13] Dinesh Jayaraman and Kristen Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *CVPR*, 2016. 3

[14] Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. Unsupervised domain adaptation for zero-shot learning. In *ICCV*, 2015. 2

[15] Tian Lan, Yuke Zhu, Amir Roshan Zamir, and Silvio Savarese. Action recognition by hierarchical mid-level action elements. In *ICCV*, 2015. 2

[16] Mans Larsson, Erik Stenborg, Carl Toft, Lars Hammarstrand, Torsten Sattler, and Fredrik Kahl. Fine-grained segmentation networks: Self-supervised segmentation for improved long-term visual localization. In *ICCV*, 2019. 2

[17] D. Lee. Pseudo-label: the simple and efficient semi-supervised learning method for deep neural networks. In *ICML*, 2013. 2

[18] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *ECCV*, 2018. 1

[19] Jingen Liu, Benjamin Kuipers, and Silvio Savarese. Recognizing human actions by attributes. In *CVPR*, 2011. 2

[20] Pascal Mettes and Cees GM Snoek. Spatial-aware object embeddings for zero-shot localization and classification of actions. In *ICCV*, 2017. 2

[21] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013. 4

[22] Tomas Mikolov, Ilia Sutskever, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013. 4

[23] Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. Evaluating neural word representations in tensor-based compositional settings. In *EMNLP*, 2014. 4

[24] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Yan Yan, Lisa Brown, Quanfu Fan, Dan Gutfreund, and Carl Vondrick. Moments in time dataset: one million videos for event understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 1

[25] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 4

[26] Phuc Nguyen, Ting Liu, Gautam Prasad, and Bohyung Han. Weakly supervised action localization by sparse temporal pooling network. In *CVPR*, 2018. 2, 7

[27] Phuc Xuan Nguyen, Deva Ramanan, and Charless C Fowlkes. Weakly-supervised action localization with background modeling. In *ICCV*, 2019. 2, 6, 7

[28] Sujoy Paul, Sourya Roy, and Amit K Roy-Chowdhury. W-talc: Weakly-supervised temporal activity localization and classification. In *ECCV*, 2018. 1, 2, 4, 5, 7, 8

[29] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *ICLR*, 2015. 2

[30] Alina Roitberg, Manuel Martinez, Monica Haurilet, and Rainer Stiefelhagen. Towards a fair evaluation of zero-shot action recognition using external data. In *ECCV*, 2018. 2

[31] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. CDC: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *CVPR*, 2017. 7

[32] Zheng Shou, Hang Gao, Lei Zhang, Kazuyuki Miyazawa, and Shih-Fu Chang. AutoLoc: Weakly-supervised temporal action localization in untrimmed videos. In *ECCV*, 2018. 1, 2, 6, 7, 8

[33] Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012. 2

[34] Limin Wang, Yuanjun Xiong, Dahua Lin, and Luc Van Gool. Untrimmednets for weakly supervised action recognition and detection. In *CVPR*, 2017. 1, 2, 8

[35] Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002. 3

[36] Huijuan Xu, Abir Das, and Kate Saenko. R-C3D: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017. 7

[37] Xun Xu, Timothy M Hospedales, and Shaogang Gong. Multi-task zero-shot action recognition with prioritised data augmentation. In *ECCV*, 2016. 2

[38] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 126(2-4):375–389, 2018. 2, 5, 8

[39] Tan Yu, Zhou Ren, Yuncheng Li, Enxu Yan, Ning Xu, and Junsong Yuan. Temporal structure mining for weakly supervised action detection. In *ICCV*, 2019. 2, 7, 8

[40] Yue Zhao, Yuanjun Xiong, Limin Wang, Zhirong Wu, Xiaoou Tang, and Dahua Lin. Temporal action detection with structured segment networks. In *ICCV*, 2017. 1, 7

[41] Yi Zhu, Yang Long, Yu Guan, Shawn Newsam, and Ling Shao. Towards universal representation for unseen action recognition. In *CVPR*, 2018. 2

[42] Y. Zou, Z. Yu, X. Liu, B.V.K. V. Kumar, and J. Wang. Confidence regularized self-training. In *ICCV*, 2019. 2