One-shot Action Localization by Learning Sequence Matching Network

Hongtao Yang
The Australian National University
u5226028@anu.edu.au

Xuming He
ShanghaiTech University
hexm@shanghaitech.edu.cn

Fatih Porikli
The Australian National University
fatih.porikli@anu.edu.au

Abstract

*Learning based temporal action localization methods require vast amounts of training data. However, such large-scale video datasets, which are expected to capture the dynamics of every action category, are not only very expensive to acquire but are also not practical simply because there exists an uncountable number of action classes. This poses a critical restriction to the current methods when the training samples are few and rare (*e.g. *when the target action classes are not present in the current publicly available datasets). To address this challenge, we conceptualize a new example-based action detection problem where only a few examples are provided, and the goal is to find the occurrences of these examples in an untrimmed video sequence. Towards this objective, we introduce a novel one-shot action localization method that alleviates the need for large amounts of training samples. Our solution adopts the one-shot learning technique of Matching Network and utilizes correlations to mine and localize actions of previously unseen classes. We evaluate our one-shot action localization method on the THUMOS14 and ActivityNet datasets, of which we modified the configuration to fit our one-shot problem setup.*

1. Introduction

Temporal action localization, which jointly classifies action instances and localizes them in an untrimmed video, is a key task in video understanding. Due to a wide range of applications such as video surveillance and video analytics [31, 16], it has been an active research topic in recent years [25, 30, 38, 49, 35, 48, 15, 50, 51, 34, 23, 14, 28, 47, 8]. While early attempts relied on handcrafted video features [31, 19, 44, 25], deep neural network based approaches have made significant progress [9, 20, 35, 48, 49, 34] and reported the state-of-the-art performance due to data-driven spatiotemporal feature representations, effective end-to-end learning strategies, and the availability of large-scale action/activity datasets [40, 18, 10].

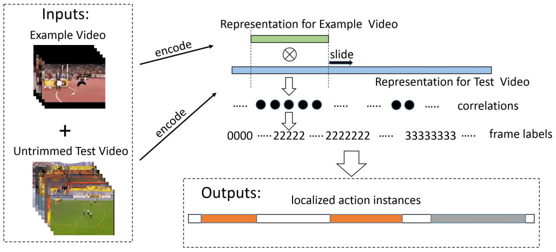
Inputs:
Example Video
+
Untrimmed Test Video
encode
encode
Representation for Example Video
⊗
slide
Representation for Test Video
····· ● ● ● ● ● ····· ····· ● ● ····· correlations
0000 ·····22222 ·····2222222 ····· 33333333 ····· frame labels
Outputs:
localized action instances

Figure 1. Example-based action localization problem setup. Inputs are a few example videos and a untrimmed test video, after encoding and computing correlation, frame labeling are obtained by comparing correlations. Finally, frame labeling are combined into action instances.

Most existing deep network based action localization methods, however, adopt a strongly supervised learning strategy that relies on a large amount of annotated video data, which are costly to collect [20, 21]. While transfer learning or model pretraining may mitigate this problem to some extent [34, 6, 48], it remains challenging to handle novel action classes and adapt learned network models to a new scenario with high data efficiency. By contrast, human beings are capable of learning new action classes and localizing their instances from only a few examples of each class [13, 7]. Therefore, it is much desirable to incorporate such an efficient learning strategy into action localization for more flexibility and better generalization when annotations are scarce.

To this end, in this paper, we consider a one(few)-shot learning scenario of action localization; given one (or a few) example of new action classes, typically one per class, our goal is to detect all occurrences of each class in an untrimmed video. While one-shot learning has been extensively studied in the context of visual recognition [1, 24, 32, 33, 43, 12, 29], few studies address the challenge of learning from a few instances to *detect* spatiotemporal patterns, such as actions, in videos. To tackle this problem, we develop a novel meta-learning strategy that integrates the task-level prior knowledge on matching video sequences into learning action localization. The key ideas of our one-shot learning strategy are an intuitive, structured representation of action videos suitable for matching (partial) sequences and a similarity metric that allows us to transfer the labeling of action examples to action proposals in the untrimmed video. Figure 1 shows an example of our one-shot action localization pipeline.

Specifically, we propose a new Matching Network architecture that takes as input a few examples per action class and predicts a dense temporal labeling of a test video in terms of action instances. In contrast to the classification network in [43], our localization network first generates a sequence of action proposals in a sliding window and predicts their action labels through three network components, as follows. The first component, *a video encoder network*, computes a segment-based action representation for each action proposal and reference action, which maintains the temporal structure of actions and allows for accurate localization. The action proposals are then compared with every reference action by the second component, *a similarity network*, which generates a set of correlation scores at each time step. Based on the correlation scores within a time window, the third component, *a labeling network*, predicts the action class label (as one of the foreground classes or the background) for the proposal in each time step.

Since all the network components are differentiable, our localization network can be trained in an end-to-end fashion. We evaluate our approach on the Thumos14 and the ActivityNet dataset and report significantly better performance than the state-of-the-art methods while using only a small training set. Our ablative studies also demonstrate the efficacy of every model components in our architecture.

Our main contributions in this work are three-fold:

- We introduce an one(few)-shot action localization problem that addresses the novel task of detecting action instances given only one (or a few) annotated video examples per class;
- We propose a meta-learning approach to the action localization problem based on the Matching Network framework, which is capable of capturing task-level prior knowledge;
- We develop a structured representation of action videos for matching that encodes the temporal ordering information and produces more accurate localization results.

2. Related Work

One-shot Learning Modern techniques for one-shot learning tend to utilize the meta-learning framework. Meta-learning techniques are applied on a set of datasets, with the goal of learning transferable knowledge among datasets

Some approaches train a meta-learner that learns how to update the parameters of the learners' model, usually a deep neural network [1, 24, 32]. The meta-learner learns a update rule that will guide the learner to quickly adapt to each individual tasks. MAML [12] combines the meta-learner and the learner into one, by directly computing gradient with respect to the meta-learning objective. In [33], a memory-augmented neural network is trained to learn how to store and retrieve memories. Also tries to retrieve past memories, TCML [29] treats each individual dataset as a sequential input, and use temporal convolution to automatically discover learning strategies.

Metric learning methods are also employed by many one-shot learning algorithms that generate good results. Deep siamese networks [22] train a convolutional network to embed examples so that samples in the same class are close while samples in different classes are far away. [43, 36, 39] refine this idea by introducing recurrence and attention mechanisms.

In this paper, we follow the framework of matching network [43], using correlation to classify given videos, and adapt it for the example based action detection problem.

Action localization Fully supervised action localization problem has been extensively studied [25, 30, 38, 49, 35, 48, 15, 50, 51, 34, 23, 14, 28, 47, 8]. To capture temporal dynamics, LSTM networks have been widely used in representing action instances. [38, 49, 28] use LSTM in addition to CNN to better model the temporal dynamics each proposal. In this work, we use LSTM to encode finer temporal details into the encoding vector of the video. 3D CNN and temporal convolution have proven to be effective in capturing spatial temporal features, and is reported to achieve state-of-the-art results for both action recognition and action localization tasks. [42, 35, 48, 34, 6, 23]. However, 3D network is hard to train and require a large amount of data [6]. Other methods use structured representations to model the details of action instances. Yuan *et al*. [50, 51] improves localization accuracy by explicitly finding the start, middle and end stage of an action. In this paper, we also employ a structured representation of the video, but we do not explicitly model different stages of an action.

Action localization has been studied in other non-fully supervised setup by a few works. [3, 17] proposed some weakly supervised action labeling methods using only the action order information. UntrimmedNets [45] employs a different type of weak supervision by using only untrimmed videos without annotations during training, and a attention based modeling method is proposed under such weak supervision. [41] uses no supervision at all and formulate the unsupervised action detection problem as a Knapsack problem. They propose a supervoxel-based pipeline that

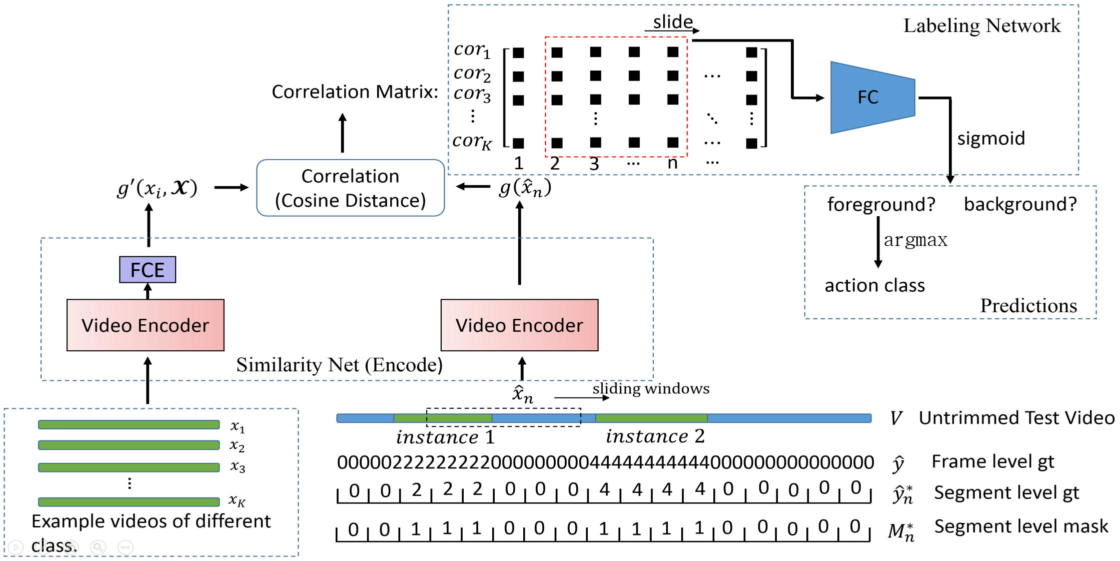

Figure 2. Overview of the one-shot localization System. We use sliding window to swipe over the untrimmed test video with the stride equal to the segment length. For each window, we compute correlations with all reference examples. By concatenating correlations with every example at every time step, a correlation matrix is obtained. In the end, a FC network is applied on the correlation matrix over a certain time span to make segment level classification of fore/background. On the places of foreground, action classification is done through comparing correlations. We use different window sizes for multi-scale predictions.

first discover actions and then spatial-temporally localize them. Unlike these problem setups, we aim to address the example-based action detection problem, which can be formulated as one-shot action localization. The one-shot aspect of the problem restrict us from using any pre-trained networks.

3. One-shot Localization System Overview

Given a few typical examples from a set of new action classes, our goal is to locate all the instances of these classes in an untrimmed video. To this end, we propose a meta-learning strategy to learn a structured representation of action instances and a matching similarity metric of actions that enable us to classify every action candidate in the untrimmed video based on the action examples and their class labels. An overview of our one-shot action localization net is shown in Figure 2.

Specifically, we develop an one-shot localization neural network based on the Matching Network framework [43], which takes as input the examples of new action classes and predicts a dense labeling of the test video for localizing action instances of these classes. Our network adopts a sliding-window strategy to generate a sequence of action proposals, and assign a class label to each of these proposals through three main network modules as follows. The first network module is a video encoder network that generates a fixed-length feature representation consisting of multiple segments for action proposal and examples. At each time step, a correlation score between each pair of the proposal and the reference examples is computed by the second network module, a similarity network. The correlation scores at every time step are concatenated to form a correlation score matrix, which captures how the similarity between the example and the untrimmed test video changes through time. Finally, a labeling network predicts the classes of every action proposals based on the correlation scores in a local temporal window.

In the remainder of this section, we will present the details of the three network components: the video encoder, the similarity network and the labeling network. We will discuss the one-shot learning strategy and the optimization process for the overall localization network in Section 4.

3.1. Video encoder network

Our matching-based action localization relies on good alignment between the candidate and the reference videos. To achieve accurate alignment, we intend to maintain the temporal structure of action videos in our action representation. To this end, we develop a segment-based video representation with ranking LSTM to encode each action instance as a fixed-length sequence of video segment features.

More specifically, given a target video x_i, we evenly split the video into S segments. Each segment is first encoded by a two-stream temporal segment network [11, 37, 46], denoted as ϕ, that generates an initial encoding vector $\phi(x_{i,s})$ for the s-th segment where $s = 1, \cdots, S$. The temporal segment structure allows us to generate a finer-grained or partial alignment between two video sequences, e.g., a reference example and a proposal. To capture the temporal context, we add an additional LSTM network layer that

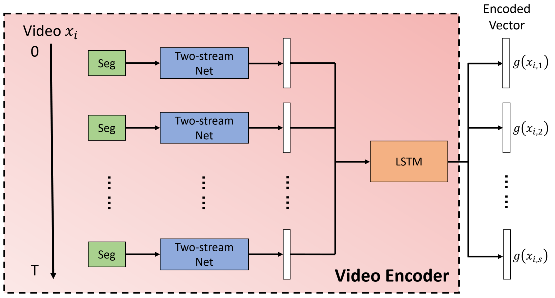

Figure 3. The video encoder. It uses temporal-segment two-stream network to encode the video into a structured representation. The final encoded vector for the video is the concatenation of the LSTM output for each segment.

takes as input the sequence $\{\phi(x_{i,s})\}_{s=1}^{S}$, and produces the final encoding of the action segments as

$$\{g(x_{i,s})\}_{s=1}^{S} = \text{LSTM}(\{\phi(x_{i,s})\}_{s=1}^{S}) \tag{1}$$

The structure of our video encoder is illustrated in Figure 3. The LSTM layer is trained with a ranking loss to enhance the temporal order structure, which we will explain in detail in Sec 4.3. The overall representation of the target video is formed by concatenating the encoding vectors of all the segments.

3.2. Similarity network for matching actions

Given the video representation, we now turn to the task of classifying an action candidate into one of the action classes with example videos or the background. Inspired by the Matching Network architecture [43], we first compute a correlation score between the action candidate and each action example using a similarity network, from which we then predict the class label of the action candidate as described in Sec 3.3.

Formally, we refer to the dataset of action examples and their labels as the support set and denote it as $\{\mathcal{X} = \{(x_i, y_i)\}_{i=1}^{c*k}$ where c is the number of classes and k is the number of examples per class. The similarity network first compute the full context embedding [43] of each individual examples x_i with respect to the entire support set. Specifically[1], let $g(x_i)$ be the encoding vector of example x_i, we define the Full Context Embedding (FCE) $g'(x_i, \mathcal{X})$ as

$$g'(x_i, \mathcal{X}) = \overrightarrow{h}_i + \overleftarrow{h}_i + g(x_i) \tag{2}$$

$$\overrightarrow{h}_i, \overrightarrow{c}_i = \text{LSTM}(g(x_i), \overrightarrow{h}_{i-1}, \overrightarrow{c}_{i-1}) \tag{3}$$

$$\overleftarrow{h}_i, \overleftarrow{c}_i = \text{LSTM}(g(x_i), \overleftarrow{h}_{i+1}, \overleftarrow{c}_{i+1}) \tag{4}$$

where the FCE uses a bi-directional LSTM and treats the $g(x_i)$ as an input sequence. The FCE vector $g'(x_i, \mathcal{X})$ is the summation of outputs of ith step of the LSTM in both directions, and the original $g(x_i)$. The full context embedding encodes the dataset context and enriches the representations of action examples.

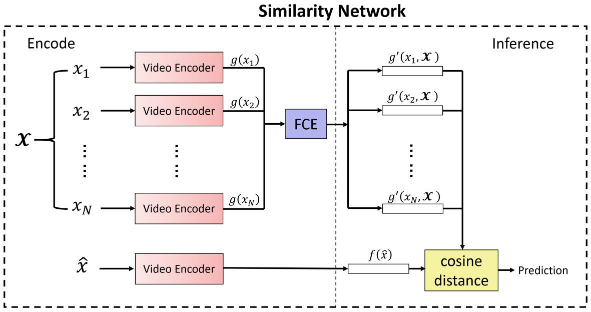

Figure 4. Similarity Network. All example videos along with a test video are the inputs for the similarity network. The similarity network applies FCE on each encoding vectors of example videos. The FCE is parameterized by a bi-directional LSTM.

Now given an action proposal $\hat{x}$ and its encoding vector $g(\hat{x})$, the similarity network computes the cosine distances between the representations of the proposal and all the examples:

$$d(\hat{x}, x_i) = \frac{g(\hat{x})^T g'(x_i, \mathcal{X})}{|g(\hat{x})| \cdot |g'(x_i, \mathcal{X})|} \tag{5}$$

Based on the distances, the original matching network uses an attention mechanism and voting strategy to classify the test data into one of classes in the support set [43]:

$$\hat{y} = \sum_{i=1}^{c*k} a(\hat{x}, x_i) y_i \tag{6}$$

where $a(\hat{x}, x_i)$ is the softmax attention of test sample $\hat{x}$ to the examples x_i:

$$a(\hat{x}, x_i) = e^{d(g(\hat{x}), g'(x_i, \mathcal{X}))} \Big/ \sum_j e^{d(g(\hat{x}), g'(x_j, \mathcal{X}))} \tag{7}$$

We note that, as the support set is only composed of foreground classes, such classification method is not applicable to the localization task, in which we also have to distinguish foreground from background. Therefore, in our one-shot action localization architecture, we use the similarity network to compute the correlation scores, and design a separate labeling network to infer the class labels (including background) of each proposal in the following subsection.

3.3. Labeling network for localization

Our action localization network uses the sliding-window strategy to generate action proposal with a fixed stride step. In this work, we set the stride size equal to the segment length used in the video encoder. For each window, we obtain a encoding vector and compute correlation scores

[1] For simplicity and readability, we drop the subscript s.

with the example videos as discussed above. Formally, we denote the length of each window as l, the number of segments in each window as S, and the length of the test video (number of frames) as N. Then the segment length, also the stride size, is $\frac{l}{S}$ and the number of window is NS/l. Let $\hat{x}_n$ denote nth window proposal, $\hat{x}_n = \{f_{(n-1)l/S} \cdots f_{(n-1)l/S+l}\}$, where $n \in [1, \frac{NS}{l}]$. For each window we have:

$$cor_{i,\hat{x}_n} = g'(x_i, \mathcal{X})^T g(\hat{x}_n) \tag{8}$$

By concatenating $cor_{i,\hat{x}_n}$ for each i and n, we obtain a correlation matrix. Window length l indicate the scale under which we are searching. With different l, we can get correlation matrix for multiple scales, thus can obtain multi-scale predictions.

The labeling network is applied directly on the correlation matrix. Similar to Eq 6 where classification is done through comparing cosine distances, we compare different rows of the same column of the correlation matrix by applying a fully connected neural network on the correlation matrix over a short temporal span, and output a probability distribution over foreground and background via sigmoid activation. The fully connected network slides over the correlation matrix along the time dimension, determining fore/background for every proposal. On the proposals which the labeling network determines to be foreground, Eq 6 is applied to predict an action label, which is shown in the upper part of Figure 2.

Our labeling network is applied over a certain time span, in order to capture the contextual information and to reduce frequent label switch between fore/background. Note that the correlation matrix contains both information about specific action classes, and whether a proposal belongs to the background or foreground. For example, if the correlation with one example are much higher than that with the others, then the proposal most likely belongs to foreground and has the same action label as the example; If the correlations with all examples are relatively low, then it probably belong to the background. The labeling network learns these criteria through training.

Also note that the labeling network is, in a sense, independent to the action classes, as it is applied on the correlation matrix rather than the feature representation of each videos. This means that criteria it learned should be applicable to input videos from different classes, which makes it suitable for one-shot prediction. One can think of the correlation as an operation that filters out irrelevant video-specific information, and left behind only relevant similarity information. The task of the labeling network is only to determine whether there are any outstanding correlations at a specific location. Therefore, we can directly apply the trained network to other previously unseen classes and still expect a good performance.

Postprocessing In the post processing stage, we combine multi-scale proposal-level predictions to obtain a single frame-level predictions for the test video, and group adjacent frames of the same label to obtain action instances. Specifically, we employ three steps of postprocessing. First, we map the proposal prediction to its central segment and each frame in the segment share the same probability distribution as the segment. Second, we combine multi-scale prediction into one: Each frame has a probability distribution for each scale. We add the probabilities from every scales and choose the highest one and the corresponding class as the confidence score and the final prediction for the frame. Finally, We group adjacent frames of the same label to get predicted action windows. The confidence score for the predicted window is the average score of every frames in that window. This postprocessing step essentially achieves the Non-maximum Suppression (NMS) for the localization.

4. One-shot Model Learning

Our localization system is designed for one-shot action localization, the corresponding meta-learning formulation should be employed in the training of the model. Every component of the system is differentiable and the system can be trained end-to-end. However, to get better initialization and performance, we pre-train the video encoder and the similarity network. In the remainder of this section, we first present the meta-learning formulation employed in this work, than the overall loss function for the localization system. Finally, we present the the model pre-training using slightly different loss functions.

4.1. Meta Learning Formulation

In meta-learning, the model is trained in a meta-phase on a set of training tasks, and is evaluated on another set of testing tasks [43, 33, 12]. Formally, we define $\mathcal{T}_{meta-train}$ to be the collection of the meta-training tasks: $\mathcal{T}_{meta-train} = \{\mathcal{X}, \hat{\mathcal{X}}, \mathcal{L}(\mathcal{X}, \hat{\mathcal{X}}, \theta)\}$. Each task consists of its own training set $\mathcal{X} = \{x_i, y_i\}$, test set $\hat{\mathcal{X}} = \{\hat{x}_j, \hat{y}_j\}$ where y is the ground truth label for classification task, or a real value for regression task, and some loss function $\mathcal{L}$ to measure the performance of the meta-learner depending on the system parameters θ. Similarly, we can define the meta-testing tasks as $\mathcal{T}_{meta-test} = \{\mathcal{X}, \hat{\mathcal{X}}, \mathcal{L}(\mathcal{X}, \hat{\mathcal{X}}, \theta)\}$. The goal of meta learning is to find the model that minimize the meta-test loss across a distribution over the meta-test tasks:

$$\theta^* = \text{argmin}_\theta \mathbb{E}_{\mathcal{T} \sim \mathcal{T}_{meta-test}}[\mathcal{L}(\mathcal{T}, \theta)] \tag{9}$$

During training, the test loss on $\hat{\mathcal{X}}$ of sampled tasks $\mathcal{T} \sim \mathcal{T}_{meta-train}$ is served as the training loss to the meta-learner. For each training set $\mathcal{X}$ in $\mathcal{T}_{meta_train}$, if only one or a few samples are presented, then the problem is known as one(few)-shot learning.

4.2. Optimization for the Localization System

The FC network in Figure 2 outputs a score $F_\theta(n)$ for each segment indicating whether it belongs to foreground or not. Using the segment level mask M_n^* as ground truth for fore/background, we can compute the localization loss function as the binary cross entropy:

$$L_{loc} = -\frac{l}{NS}\sum_{n=1}^{NS/l}[M_n^* \log F_\theta(n) + (1 - M_n^* \log(1 - F_\theta(n))] \tag{10}$$

We also compute classification loss at every segments whose ground truth is foreground:

$$L_{cls} = -\frac{l}{NS}\sum_{n=1}^{NS/l}[M_n^* \log P_\theta(\hat{y}_n|\hat{x}_n, \mathcal{X})] \tag{11}$$

where $\hat{y}$ is given by Eq 6. Note that during testing, we only further classify the segments who are predicted to be foreground into action classes. However during training, we compute classification loss on every ground truth foreground segments.

Using the meta-learning framework, we can have the final training loss for the one-shot localization task as:

$$L = \mathbb{E}_{\mathcal{T}\sim\mathcal{T}_{meta-train}}[L_{loc} + L_{cls}] \tag{12}$$

4.3. Pretraining for Video Encoder & Similarity Net

The video encoder and the similarity network contain most of the trainable parameters in the system, including the two-stream temporal segment network, the LSTM lyaer and the FCE module. Therefore we pretrain the video encoder and the similarity network under the action recognition task to get better initialization. During the pretraining stage, only trimmed action instances are used, thus we are only concerned the classification loss:

$$L_{cls,s} = \log P_\theta(\hat{y}_s|\hat{x}_s, \mathcal{X}) \tag{13}$$

where $\hat{y}$ is again given by Eq 6, $\hat{x}$ is the test action instance and $\mathcal{X}$ is the collection of all example videos. With the model pretraining, we can also take advantage of the LSTM in the video encoder by employing ranking loss [28] to incorporate better temporal structures into the encoding vectors $g(x_i)$ given in Eq 1. Formally, the ranking loss is:

$$L_{rank,s} = \max(0,\ p_s^{\hat{y}} - p_s^{*\hat{y}}) \tag{14}$$

$$p_s^{\hat{y}} = (\hat{y}'_s)^T \cdot \hat{y} \tag{15}$$

$$p_s^{*\hat{y}} = \max_{s'\in[1,s-1]} p_{s'}^{\hat{y}} \tag{16}$$

where $\hat{y}'_s$ is the predicted probability distribution and $\hat{y}$ is the ground truth label. In other words, $p_s^{\hat{y}}$ and $p_s^{*\hat{y}}$ are the predicted probability with respect to the ground truth label $\hat{y}$ at segment s, and the past maximum predicted probability respectively.

The intuition of ranking loss is to encourage the classifier to make more confidant prediction as it is presented with more and more of the video content. The loss incurs a penalty when the prediction confidence drops at the sth segment compared to the highest confidence among past segments. This encourages the network to make monotonic increasing predictions, which in turn, encourages the encoding vectors of different segments $\{g(x_{i,s})\}_{s=1}^S$ to be more discriminative to each other, thus improve the temporal structure of the final encoding vector, which is the concatenation of $\{g(x_{i,s})\}_{s=1}^S$.

Under the meta-learning framework, we can write the final training loss for the pretraining of the video encoder and the similarity net as:

$$L = \mathbb{E}_{\mathcal{T}\sim\mathcal{T}_{meta-train}}\{\sum_s [L_{cls,s} + L_{rank,s}]\} \tag{17}$$

5. Experiments

5.1. Datasets & Preparation

We use the Thumos14 [18] and ActivityNet 1.2 [10] datasets to evaluate our one-shot action localization algorithm.

The one-shot problem setup requires that the classes during testing must not be present during training. Thumos14 contains 20 classes from UCF-101, so only the other 81 classes in UCF-101 are used during training the encoder. We denote the two splits of UCF-101 as UCF-101-81 and UCF-101-20. After training the encoder, we use a small part of Thumos14 validation set (contain 6 classes) to train the fully connected network with the features extracted by the video encoder, and use the remaining 14 classes in the test set to test our one-shot localization network. The two splits of Thumos14 validation set and test set are denoted as Thumos-val-6 and Thumos-test-14.

Similarly for activity net, we split the 100 classes into 80-20 splits. Our one-shot action localization network is trained on videos containing only the 80 classes in the training set, denoted by ActivityNet-train-80, and is tested on the other 20 classes in the validation set, denoted by ActivityNet-val-20.

Model pre-training We split the UCF-101-81 dataset into 10000 small datasets each containing 5 example videos of 5 different classes and 1 test video belonging to one of the 5 classes. For each small dataset, the 5 action classes, 5 example videos and 1 test video are all randomly chosen. The 5 action classes are randomly assigned labels from 0-4. Same operation is applied for ActivityNet. We train the one-shot video encoder and the similarity net under the meta-learning setup on all such small datasets.

	mAP		mAP
Heilbron *et al.* [5]	13.5	Ours@1	13.6
Yeung *et al.* [49]	17.1	Ours@5	14.0
Yuan *et al.* [50]	17.8	Ours@15	14.7
S-CNN [35]	19.0	CDC@1	6.4
S-CNN + SST [4]	23.0	CDC@5	6.5
CDC [34]	23.3	CDC@15	6.8

Table 1. Comparision of our one-shot localization method with state-of-the-art methods on Thumos14. The left half are the results under full supervision, the right half are results for one-shot setup. @n means n examples per class are used during training. The IoU threshold is set to 0.5 for all comparison.

Training for Localization We split the Thumos-val-6 dataset and UCF-101-20 into small datasets in a similar manner. We randomly choose one video from Thumos-val-6, and pair it with 5 examples UCF-101-20 to form a small dataset. The examples are chosen from the mutual classes of UCF-101-20 and Thumos14-6. All such datasets are used for training the one-shot action localization network. The 5 examples are randomly assigned labels from 1-5, with 0 represent background. Same operation is applied for ActivityNet. At this stage, the parameters of the video encoder are fixed, only the fully connected network are updated.

Testing for Localization During the meta-testing stage, the setup is identical to that of meta-training stage, only now we use Thumos-test-14. We pair a randomly chosen video in Thumos-test-14 with 5 examples from the mutual classes with UCF-101-20. As can be easily seen, there are a large number of different combinations of the 5 examples (random classes, and random sample within each class), and the localization performance is dependent on the example videos chosen. To get a reliable test results, we randomly sample 1000 different datasets from each of the 14 test classes and calculate mAP across all these datasets. Same evaluation setup is use for ActivityNet.

Training details During training of the encoder as well as the FC network, the initial learning rate is set to be 0.001 and decrease by a factor of 5 once the training loss stop decreasing. To prevent including too many backgrounds during training, we only use the cropped clips of the untrimmed videos where the ratios of background and foreground frames are between 0.5:1 and 1.5:1 . Also, to ensure all such clips have the same length, we repeat multiple instances of shorter clips to make its length the same as longer ones.

5.2. Experiments Results

Comparison with fully supervised action detection To demonstrate the effectiveness of our approach, we make a comparison with one of the state-of-the-art action detection method under the one-shot setup, showing that a method which works well with a lot of training data will perform poorly with very little data. In addition, we include the results of other state-of-the-art methods under full supervision to put our method in context.

	mAP@0.5	Average mAP
TCN [8]	37.4	23.5
R-C3D [48]	-	26.8
Wang *et al.* [26]	42.2	14.8
Lin *et al.* [27]	48.9	32.2
Xiong *et al.* [47]	41.1	24.8
CDC [34]	43.8	22.7
Ours@1	22.3	9.8
Ours@5	23.1	10.0
CDC@1	8.2	2.4
CDC@5	8.6	2.5

Table 2. Compare our one-shot localization method with state-of-the-art methods on ActivityNet. The Upper half are the results under full supervision, the lower half are results for one-shot setup. @n means n examples per class are used during training.

We see from Table 1 and 2 that, although there is still a performance gap between one-shot and fully supervised action detection, our method significantly outperforms the state-of-the-art method when tested in one-shot setup[2]. We further test our method using more training data, 15 samples per class, on the Thumos14 dataset. Our results show noticeable performance jumps while only small improvements are observed for CDC, which indicates a good scalability of our approach w.r.t. the number of samples.

Under the meta-learning setup, the labels are randomly changed for each action class for every iteration during training, which forces the feature representation of our video encoder to decouple from the labels. This way the video encoder can treat new action classes just as any other classes, thus achieving fast adaptation from only one sample. The class-independent property of the parameters learned from meta-learning are particularly desirable since the matching network employs a non-parametric approach for classification. However, for existing methods, when a sample changes its ground truth label from one iteration to another, it can confuse the network, preventing the model from learning discriminative features effectively.

Figure 5 visualize the correlation matrix and the classification criteria learned by the FC network. We can see that the FC network will output background when all correlations are relatively low, or when there is no outstanding correlation. Correct predictions can be made through comparing correlations most of the time, which indicates that the feature representation of our one-shot video encoder is discriminative. However, false predictions do appear around the action boundaries. Also, different time steps of the

[2] In the comparison of one-shot performance, we train the CDC network from scratch, as opposed to using the pre-trained model on Sports-1M in the paper, because the one-shot setup forbids us of using any pre-training.

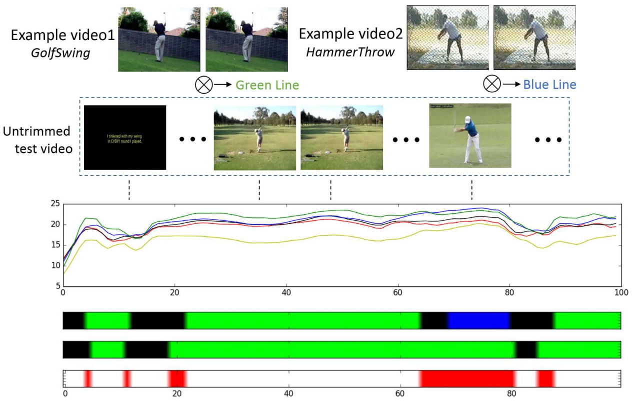

Figure 5. A portion of the correlation matrix with its corresponding video frames and predictions. Test video contains multiple instances of *GolfSwing*, one of which is similar to the *HammerThrow* example video. **Green** is the ground truth action, **black** is background and **blue** is another action class. On the last row, **red** is false predictions.

same action may express resemblance with different examples (blue part in Fig 5), which will lead to false predictions.

Effect of the Similarity Net We conduct an ablative study and evaluate the one-shot performance where the similarity network is replaced by binary SVM classifiers. The result is shown in Table 3, which demonstrates the vital role of the similarity network in learning effective representations for the one-shot localization.

	mAP
Our encoder + Binary classifier	6.0
Our encoder + Similarity net	13.6

Table 3. Comparison between the similarity network and a binary linear SVM classifier with the same video encoder.

Effect of Temporal Alignment In this work, a finer level of alignment is obtained through the segment-based video representation of the encoder, as well as the ranking loss LSTM. Below we compare one-shot localization and recognition performance under different alignment schemes, demonstrating the critical role of our alignment for the localization task and its effectiveness in learning discriminative feature representation, respectively.

Alignment Scheme	mAP	accuracy
No alignment	7.8	-
TS, w/o ranking loss	12.9	57.5
TS, with ranking loss	13.6	57.4

Table 4. One-shot detection mAP and one-shot recognition accuracy under different alignment scheme. In the table, *No alignment* means not employing segment-based representation, nor ranking LSTM.

We can also see from the Table 4, a good alignment is critical in our algorithm. With segment-based representation, the finer temporal structure allows for more accurate localization. Although most of the alignment property is gained through the temporal segment structure of the encoder, ranking loss also provides a noticeable performance boost, as it encourages the features from different segment to evolve over time, discriminating features from different segments, thus improved the structured representation. In the third column, the accuracy with and without ranking loss are practically identical, indicating that with feature alignment we can also learn an expressive representation.

Validity of the Video Encoder As explained in section 3.1, the video encoder employs the temporal-segment two-stream structure. Specifically, the architecture of the CNN is what we call mini-VGG, which follows the architecture of the VGG-16 network, but with a only quarter of the convolutional kernels, resulting in a significantly smaller network size. The main reason of employing such architecture is to reduce the memory requirements without causing too much performance degradation. Table 5 shows the encoder performance of the fully supervised action recognition task on the UCF-101 dataset as well as the number of parameters. Although our encoder is not the best performing one, it is much more efficient and with a competitive performance. More importantly, the segment-based encoder is critical in our localization system, as shown before.

	Accuracy(%)	# parameters
two-stream network[37]	82.9	~27.5M
dynamic image [2]	76.9	~36.1M
two-stream fusion [11]	85.2	~97.3M
my encoder (mini-VGG16)	81.3	~3.6M

Table 5. Video encoder performance comparison for fully supervised action recognition. Our encoder performs reasonably well with only a fraction of the network size compared to other widely used encoders.

6. Conclusion

In this work, we present a one-shot action localization system that employs a similarity based classification strategy. We develop the correlation matrix representation and train a labeling network on the meta-level to characterize how the correlations evolve over time. This enable us to distinguish between example classes and to detect background as well, which is crucial to the localization task but lacking in the original matching network. Our network is trained under the meta-learning framework so that it can quickly adapt to new classes with just one(few) training samples. We also develop a structured representation of action videos for matching that encodes the temporal ordering information and can benefit to more accurate localizations.

Acknowledgments The Titan X used for this research was donated by the NVIDIA Corporation.

References

[1] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[2] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[3] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision*, pages 628–643. Springer, 2014.

[4] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. C. Niebles. Sst: Single-stream temporal action proposals.

[5] F. Caba Heilbron, J. Carlos Niebles, and B. Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1914–1923, 2016.

[6] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. *arXiv preprint arXiv:1705.07750*, 2017.

[7] P. D. Human and animal cognition: Continuity and discontinuity. 104(35), 2007.

[8] X. Dai, B. Singh, G. Zhang, L. S. Davis, and Y. Q. Chen. Temporal context network for activity localization in videos. *arXiv preprint arXiv:1708.02349*, 2017.

[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*, June 2015.

[10] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. 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.

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

[12] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning*, 2017.

[13] S. H. Frey and V. E. Gerry. Modulation of neural activity during observational learning of actions and their sequential orders. 26, 2006.

[14] J. Gao, Z. Yang, C. Sun, K. Chen, and R. Nevatia. Turn tap: Temporal unit regression network for temporal action proposals. *arXiv preprint arXiv:1703.06189*, 2017.

[15] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[16] S. Herath, M. Harandi, and F. Porikli. Going deeper into action recognition: A survey. *arXiv preprint arXiv:1605.04988*, 2016.

[17] D.-A. Huang, L. Fei-Fei, and J. C. Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision*, pages 137–153. Springer, 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] S. Karaman, L. Seidenari, and A. Del Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. In *ECCV THUMOS Workshop*, volume 1, page 5, 2014.

[20] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[21] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[22] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[23] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks for action segmentation and detection. 2017.

[24] K. Li and J. Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

[25] I. Lillo, J. Carlos Niebles, and A. Soto. A hierarchical pose-based approach to complex action understanding using dictionaries of actionlets and motion poselets. In *CVPR*, pages 1981–1990, 2016.

[26] T. Lin, X. Zhao, and Z. Shou. Uts at activitynet 2016. *ActivityNet Large Scale Activity Recognition Challenge 2016*, 2016.

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

[28] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1942–1950, 2016.

[29] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*, 2017.

[30] B. Ni, X. Yang, and S. Gao. Progressively parsing interactional objects for fine grained action detection. In *CVPR*, pages 1020–1028, 2016.

[31] R. Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

[32] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*.

[33] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. One-shot learning with memory-augmented neural networks. *International Conference on Machine Learning*, 2016.

[34] 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.

[35] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1049–1058, 2016.

[36] P. Shyam, S. Gupta, and A. Dukkipati. Attentive recurrent comparators. *International Conference on Machine Learning*, 2017.

[37] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[38] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1961–1970, 2016.

[39] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

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

[41] K. Soomro and M. Shah. Unsupervised action discovery and localization in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 696–705, 2017.

[42] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[43] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

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

[45] L. Wang, Y. Xiong, D. Lin, and L. Van Gool. Untrimmednets for weakly supervised action recognition and detection. *arXiv preprint arXiv:1703.03329*, 2017.

[46] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016.

[47] Y. Xiong, Y. Zhao, L. Wang, D. Lin, and X. Tang. A pursuit of temporal accuracy in general activity detection. *arXiv preprint arXiv:1703.02716*, 2017.

[48] H. Xu, A. Das, and K. Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. *arXiv preprint arXiv:1703.07814*, 2017.

[49] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2678–2687, 2016.

[50] Z. Yuan, J. C. Stroud, T. Lu, and J. Deng. Temporal action localization by structured maximal sums. 2017.

[51] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, D. Lin, and X. Tang. Temporal action detection with structured segment networks. *arXiv preprint arXiv:1704.06228*, 2017.