# End-to-End, Single-Stream Temporal Action Detection in Untrimmed Videos

Shyamal Buch[1]
shyamal@cs.stanford.edu
Victor Escorcia[2]
victor.escorcia@kaust.edu.sa
Bernard Ghanem[2]
bernard.ghanem@kaust.edu.sa
Li Fei-Fei[1]
feifeili@cs.stanford.edu
Juan Carlos Niebles[1]
jniebles@cs.stanford.edu

[1] Stanford Vision and Learning Lab
Dept. of Computer Science
Stanford University, USA

[2] Image and Video Understanding Lab
Visual Computing Center
KAUST, KSA

**Abstract**

In this work, we present a new intuitive, end-to-end approach for temporal action detection in untrimmed videos. We introduce our new architecture for Single-Stream Temporal Action Detection (SS-TAD), which effectively integrates joint action detection with its semantic sub-tasks in a single unifying end-to-end framework. We develop a method for training our deep recurrent architecture based on enforcing semantic constraints on intermediate modules that are gradually relaxed as learning progresses. We find that such a dynamic learning scheme enables SS-TAD to achieve higher overall detection performance, with fewer training epochs. By design, our single-pass network is very efficient and can operate at 701 frames per second, while simultaneously outperforming the state-of-the-art methods for temporal action detection on THUMOS'14.

## 1 Introduction

The ubiquity of cameras has led to a tremendous volume of untrimmed video data – much of which is focused on human activities. As such, temporal localization of human actions has emerged as a fundamental building block towards general video understanding. Analogous to object detection in images, temporal action localization is the task whereby computer vision algorithms must detect both the temporal bounds of actions and provide the corresponding action categories for each distinct detection. This can prove challenging for long, untrimmed video sequences, where models must be robust to variations in the frequency and temporal length of actions relative to the overall video length, which can exceed hours.

The fundamental nature of this problem has generated significant research interest in the past few years. Recent progress can be broadly categorized into three types of approaches, as shown in Figure 1. The first group [15, 29] performs analysis at the level of individual frames or groups of frames, applies temporal smoothing, and merges into detections. The second

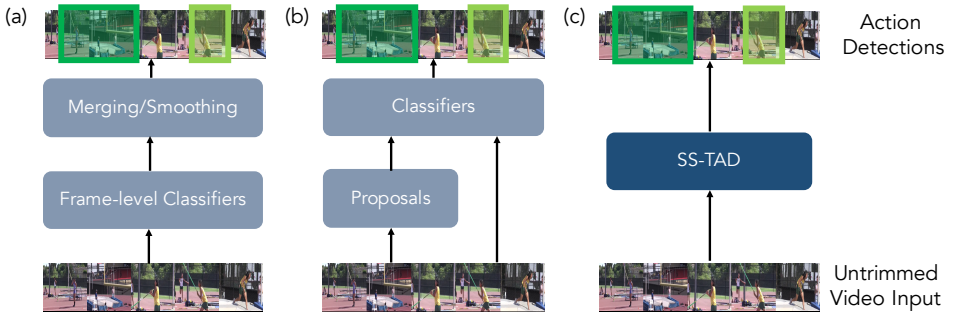


Figure 1: There have been a few dominant approaches for temporal action localization: (a) frame-level analysis followed by a separate merging step to obtain detections, (b) proposals generation followed by a separate classification over the proposals. (c) We present a new end-to-end model (SS-TAD) along a *third approach* that outputs action detections directly from a single-pass over the input video stream.

group [4, 20] does a first series of passes through the video to generate temporal proposals, and subsequently applies a classifier to each proposal to obtain the detection. The trade-off between these two is that the first group tends to offer simpler models that do not require multi-stage training at the cost of detection performance, while the latter approach generally offers superior performance, but has separate training of the proposal and classifier stages and can require multiple passes through the data at test time. This motivates approaches in the third group: modular architectures that can offer the end-to-end trainability and efficiency of the first group of approaches, while providing the superior performance of the latter. There is a strong precedent for such an approach: we draw inspiration for our model design from how single-pass object detectors such as SSD [14] and YOLO [18] improved upon multi-stage detectors such as Faster R-CNN [19] by offering tighter integration of proposals and classification sub-components. Our approach is one of the first within this group and significantly outperforms earlier approaches both in accuracy and efficiency.

In this paper, we introduce a new intuitive, end-to-end architecture (SS-TAD) for efficient temporal action localization in untrimmed video sequences. We describe our intermediate recurrent memory modules for temporal action detection, and our process by which we enforce semantic constraints which are relaxed later for general performance. Our framework processes the video in a single pass, and directly outputs the temporal bounds and corresponding action classes for the detections. Furthermore, we demonstrate experimentally that our efficient architecture achieves state-of-the-art performance on temporal action detection, while simultaneously providing a high FPS processing speed.

## 2 Related Work

Here, we review relevant recent work in video action categorization, temporal action detection in videos, and analogous work in object detection.

**Action Recognition and Detection.** There has been substantial literature on the problem of action recognition on short video clips [9]. Here, the task is to take pre-segmented video clips of a single action and provide the action class present in the video. Approaches include frame-level analysis with typical object classification networks, global representations [9],

or local models of temporal structure of motions [5, 6]. Recent work [24], such as 3D-convolutional networks (C3D), aim to improve on such methods by capturing more local information in the temporal region. However, such models are limited to classification tasks in short clips and for actions with relatively small duration.

An extension of action recognition work is spatiotemporal localization of actions in videos, where the task is to provide three-dimensional tubes or bounding boxes across time on the actors of interest in addition to the classification [3, 8, 10, 25, 28, 31]. Due to high computational complexity, these algorithms are also typically run on shorter-form videos. Thus, while such algorithms provide additional information beyond the temporal bounds, the high computational cost means these algorithms are not well-suited for analysis of long, untrimmed video sequences with sparse actions, where efficient processing is paramount.

**Temporal Action Localization.** Approaches for temporal action localization can be grouped in three categories. In the first group (Fig.1(a)), we find methods for action detection that perform frame or segment-level classification, as in [15, 29]. These approaches require post-processing smoothing and merging steps to obtain the temporal bounds, and are furthermore not as performant at temporal detection as the multi-stage methods of the second group.

Next in Fig.1(b), we find the recent development of temporal action proposals [1, 2, 4, 20, 21] as a first stage towards temporal action localization, whereby these methods provide high-recall temporal window candidates. These candidates are then passed as input to later classification stages, which can provide a classification score or further refinement of the temporal bounds, as in [1, 20, 21]. Unfortunately, such multi-stage methods treat proposals and classification as two independent, sequential processing stages, which inhibits collaboration between them and leads to repeated computation between the two stages.

Our framework belongs to the third group (Fig.1(c)). We introduce an end-to-end trainable method that tightly integrates proposal generation and classification, which results in a more efficient and effective architecture for unified temporal action detection. Recently, [30] offers an end-to-end trainable framework based on reinforcement learning, but this approach requires learning a separate policy for each class, and does not provide tight localization bounds when compared with other work. Other recent work [13, 22, 23, 27] have investigated use of neural recurrent models for temporal modeling, but rely on depth, pose, or other information for action detection, which may not be readily available for many datasets.

**Object Detection.** In many ways, this progression of action localization models parallels the development of object detection frameworks. Initial approaches to object detection adopted object proposal generation as a distinct preprocessing stage that independently provided windows to an object classifier [7], with later frameworks implementing the generation of object proposals with deep network architectures. An example of this is the Region Proposal Network (RPN) from [19], integrated with Faster R-CNN. However, we draw inspiration from subsequent generation of object detection architectures such as YOLO [17] and SSD [14], which provided a unified end-to-end approach for simultaneously outputting object proposals alongside their predicted classifications. Our approach to temporal action detection is inspired by these developments and introduces a framework that tightly integrates temporal proposals with action classifiers to achieve effective temporal localization of human actions.

# 3 Technical Approach

Our overall goal is to generate temporal action detections from an input untrimmed video sequence. Figure 2 provides an overview of our model and approach. Given an input video

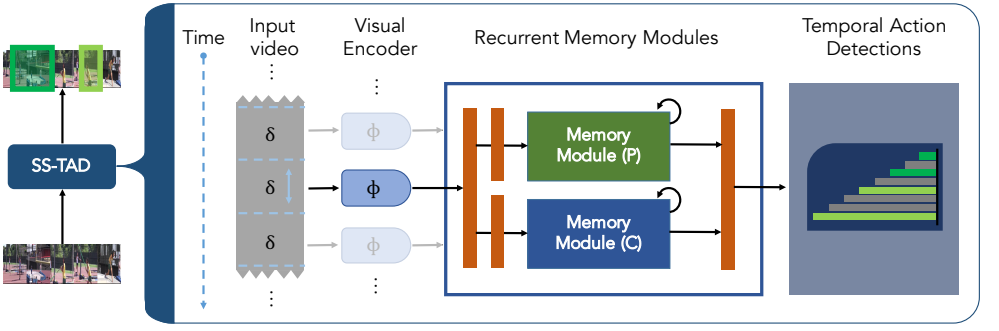


Figure 2: SS-TAD model architecture. Given an input video stream, we represent each non-overlapping "time step" $t$ with a visual encoding over $\delta$ frames. This visual encoding is the input to two recurrent memory modules that are semantically-constrained to learn proposals and classification-based *features*. These features are combined before providing the final temporal action detection output. Please refer to Section 3.1 for additional details.

sequence $X = \{x_l\}_{l=1}^{L}$ with $L$ frames, our model should provide as output (1) the temporal boundaries and (2) the corresponding action class of any activities contained within. Importantly, the model must be able to disregard irrelevant background information while still retaining relevant action information – a particularly challenging task on untrimmed video datasets. In this section, we introduce the technical details of Single Stream Temporal Action Detection (SS-TAD), our new efficient model for end-to-end temporal action detection.

## 3.1 Model

The temporal action detection task is the natural product of two main sub-tasks: (1) *temporal action proposals*, which provides temporal bounds where non-background actions are occurring, and (2) *local action classification*, which provides frame or time-step resolution classification. Rather than providing explicit solutions to these sub-tasks as done in prior work, we propose an efficient model design that focuses on providing the detections directly with information gained by *implicitly* solving these sub-tasks during inference.

Our model consists of three main components: our input visual encoding, our two recurrent memory modules, and our final output. We draw inspiration for our intuitive design from single-shot object detector frameworks, such as YOLO and SSD [14, 18]. Our *visual encoder* is reponsible for encoding lower-level spatiotemporal information from the video, while the two *memory modules* are responsible for selectively aggregating relevant context for the joint temporal action detection task. For our final *output detections*, we leverage these feature encodings from both memory modules to output the final temporal bounds and associated class scores. In contrast with prior work, our approach provides end-to-end temporal action detections with a single pass over the input video stream.

**Visual Encoder.** We capture lower-level spatiotemporal visual information from the input video frames by leveraging a 3D-Convolutional (C3D) network [24]. We choose C3D as it has been shown to effectively capture visual and motion information over small time clips with $\delta$ frames [20, 24], which have been demonstrated as effective building blocks in prior work for temporal action detection [1, 4, 20]. Since we process each frame only once, we essentially discretize the input video stream into $T = L/\delta$ non-overlapping time steps, similar to [1], where each time step has a visual encoding $\phi(\{x_i, \ldots, x_{i+\delta-1}\})$. We substitute

the *fc8* layer with a linear layer initialized from the PCA matrices released publicly by [4].

**Semantically-Constrained Recurrent Memory Modules.** A key component of our model is the introduction of recurrent memory modules, which are semantically constrained during training as a way to induce better training and test-time performance. The purpose of this component is to accumulate evidence across time as the video sequence progresses relevant to both distinguishing background from action and between classes. Each memory module consists of a multi-layer gated recurrent unit (GRU)-based network. Each module takes as input the visual encoder output $\phi$ for that time step $t$, as well as the hidden state representation from the previous time step. The output of each memory module is the hidden state embedding of the final GRU layer in that corresponding module.

As illustrated in Figure 3, a key aspect of the design of our memory modules is that they are *semantically constrained* during the training process. This means we train the modules so that their hidden embedding is useful to solve intermediate semantically meaningful tasks. While we strictly enforce the constraints early in the training process, we find that the best models are obtained when the semantic constraints are relaxed later in the training procedure.

Our final design incorporates two such memory modules, which operate in parallel. We combine both output embeddings together before proceeding to generate the final output detections. Below, we describe the different constraints applied to each module, with corresponding training loss functions defined in Section 3.2.

**Semantic Constraints: Memory Module (P).** Under semantic constraints, the goal of this module is to capture and accumulate relevant information with regard to temporal proposals, which allows the network to discern if the video under consideration contains background or action over several temporal scales.

During learning, we semantically constrain this module by training it to generate a vector $m^{(t)}_{prop}$ with confidence scores corresponding to $K$ proposals. This vector is the sigmoid output from a fully connected layer that operates on top of the hidden state of the recurrent network, and is trained with the loss $\mathcal{L}_{prop}$ defined in Section 3.2. We adopt the convention from [1], where each proposal is right-aligned; that is, at a given time step $t$ each proposal has an anchored ending boundary at frame $b^{(t)}_e = t\delta$, with a varying starting boundary $b^{(t)}_s \in \left\{b^{(t)}_e - k\delta + 1\right\}_{k=1}^{K}$. Since this supervision signal would be generated over each time step $t$ at training, we are able to consider multiple temporal scales and positions with a single pass over the input data.

We emphasize that these intermediate semantic outputs $m^{(t)}_{prop}$ are not directly used during inference for the overall detection task. Instead, this module forwards the hidden state embedding $h^{(t)}_{prop}$ of its final hidden layer for the final detection task.

**Semantic Constraints: Memory Module (C).** While the previous memory module is focused on capturing the "actionness" of multiple temporal intervals ending at a time step $t$, the classification-focused memory module is focused on retaining features representing the precise class encoding of the immediate past. To encourage this, at training time we semantically-constrain this module whereby the visual content at each time step is associated with a $(C+1)$-dimensional vector $m^{(t)}_{cls}$ with softmax confidence scores for each class and an extra dimension for the background class. This is passed as input to the loss $\mathcal{L}_{cls}$ (Sec. 3.2). Note again that $m^{(t)}_{cls}$ is not generated during inference, and the state embedding $h^{(t)}_{cls}$ is what is actually forwarded to the final detection task.

**Output Detections.** We create a joint embedding $h_{det} = h_{prop}||h_{cls}$ by concatenating the feature representation outputs from the two memory modules, taken from the final recurrent

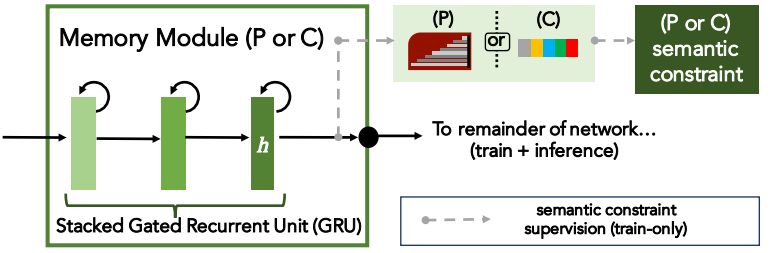


Figure 3: Each recurrent memory module consists of multiple gated-recurrent units (GRUs), and provides the final hidden state encoding $h$ as output. We apply semantic constraints on our recurrent memory modules during training to improve overall detection performance.

layer in each. For a given time step $t$, we calculate the final detection output $D^{(t)} = f_{out}(h_{det})$ as a function of this embedding. Here, $D^{(t)} \in [0,1]^{K\times(C+1)}$ is a tensor of score values such that $D^{(t)} \Leftrightarrow \{(b_s^{(t)}, b_e^{(t)}, v_k^{(t)})\}_{k=1}^{K}$ where $b_s^{(t)}, b_e^{(t)}$ correspond to right-aligned temporal intervals as defined in the proposals-based semantic constraints above, and $v_k^{(t)} \in [0,1]^{C+1}$ is the corresponding classification score vector for this interval.

## 3.2 Training

Our end goal for training is to estimate the model parameters in our architecture, such that we maximize our performance on the final temporal action detection task. Our end-to-end architecture and corresponding loss functions are fully differentiable, enabling training with backpropagation by design. Note that our single-pass design for the model means the recurrent modules must be robust when we unroll the full model over long input sequences at test time. We encourage this robustness by adopting a similar data augmentation mechanism during training as proposed in [1]. Briefly, this approach involves dense sampling of overlapping, long training window segments to encourage hidden state robustness. We provide additional discussion of this in our Supplementary Material.

**Semantic Constraints: Loss Functions.** To encourage better state representations of action context in the memory modules described in Section 3.1, we define loss functions $\mathcal{L}_{prop}, \mathcal{L}_{cls}$ for supervision on each module, respectively.

During training, we constrain the proposals-focused memory module according to a multi-label loss function on the semantic output $z^{(t)} = m_{prop}^{(t)}$ defined in Section 3.1. For a training video $X$ with corresponding groundtruth $y$, the loss at time $t$ is given by a weighted binary cross entropy objective:

$$\mathcal{L}_{prop} = -\sum_{k=1}^{K} w_0^k y_k^{(t)} \log z_k^{(t)} + w_1^k \left(1 - y_k^{(t)}\right) \log\left(1 - z_k^{(t)}\right), \tag{1}$$

where the weights $w_0^k, w_1^k$ are calculated proportional to the frequency of positive and negative proposals in the training set at each scale $k$.

We penalize the classification-focused memory module at each time step with a weighted log-likelihood loss on the output $m_{cls}^{(t)}$, with corresponding groundtruth class index $y^{(t)}$:

$$\mathcal{L}_{class} = -w^{(t)} \log\left(m_{cls}^{(t)}[y^{(t)}]\right), \tag{2}$$

where $w^{(t)}$ is set to some constant $\rho$ if $y^{(t)}$ indicates the segment is background, and 1 otherwise. This weight is tuned to the relative frequency of the background class.

**Temporal Action Detections.** We define our main joint detection loss $\mathcal{L}_{det}$ which is applied to the final detections $D^{(t)}$ output at each time step $t$ by the model. We have two components to this loss. First, we define a loss function over the classification output $v_k^{(t)}$ corresponding to each temporal interval:

$$\mathcal{L}_{detcls} = -\sum_{k=1}^{K} w_k^{(t)} \log\left(v_k^{(t)}[y_k^{(t)}]\right), \tag{3}$$

Second, to encourage the network to provide tighter localization bounds, we define an additional loss based on the localization stage in [20] to make the optimal confidence score associated with the correct class a direct function of the overlap $o$ between the groundtruth temporal annotation and the output temporal detection:

$$\mathcal{L}_{detloc} = \frac{1}{2} \cdot \sum_{k=1}^{K} \left( \frac{\left(v_k^{(t)}[y_k^{(t)}]\right)^2}{\left(o_k^{(t)}\right)^{\alpha}} - 1 \right) \cdot \mathbb{1}[y_k^{(t)} > 0], \tag{4}$$

where the indicator $\mathbb{1}[y_k^{(t)} > 0]$ is 1 if $y_k^{(t)}$ does not indicate the background class. We discuss additional context for this localization loss in the Supplementary Material.
Thus, our overall *detection* loss is:

$$\mathcal{L}_{det} = \mathcal{L}_{detcls} + \lambda_{detloc} \cdot \mathcal{L}_{detloc}, \tag{5}$$

**Overall.** We backpropagate the cumulative loss from all loss functions above at every time step $t$, so the total loss for all training examples $(X, y)$ in the batch $\mathcal{X}$ is:

$$\mathcal{L} = \sum_{(X,y)\in\mathcal{X}} \sum_{t} (\lambda_{prop} \cdot \mathcal{L}_{prop} + \lambda_{cls} \cdot \mathcal{L}_{cls} + \lambda_{det} \cdot \mathcal{L}_{det}). \tag{6}$$

where $\lambda_*$ are weighting parameters to normalize the contribution of each loss component to the overall training loss. We demonstrate in Section 4 that dynamically adjusting the terms $\lambda_{prop}$ and $\lambda_{cls}$ is effective during training, such that we have an increased importance placed on these sub-tasks for the overall detection earlier in training, but focus more on the overall detection output in the later stages. In this manner, we effectively train the network with a dynamic curriculum learning-style approach by enforcing semantic constraints on easier sub-tasks before relaxing them to focus training on the main overall detection task.

# 4 Experiments

We empirically evaluate the effectiveness of our unified end-to-end architecture for the task of temporal action detection. As our experiments show, our method achieves state-of-the-art performance while achieving some of the fastest processing speeds in the literature.

## 4.1 Experimental Setup

**Dataset.** To evaluate our model against prior work, we use the temporal action localization subset of the THUMOS'14 dataset [11]. This subset with 22+ hours of video consists of

a validation set of 200 and test set of 213 untrimmed videos annotated with the temporal intervals that depict human actions. As is standard practice, we leverage the "validation set" as our training data, performing an 80-20 split for hyperparameter optimization. To enable direct comparisons with prior work, we adopt the experimental settings from recent prior work on temporal action detection [1, 21].

**Implementation details.** We implement our model using PyTorch, with training executed on Titan X Maxwell GPUs. We initialize the *conv1* through *fc7* layers of the visual encoder with pretrained weights released by the authors of [24] from the Sports1M dataset. For training efficiency, we fix layers prior to *fc8*. We vary the number of stacked recurrent layers and hidden state size in our memory modules, as well as the number of temporal intervals $K$ considered at each time step. We optimize our model parameters with the *adam* update rule [12] and a learning rate of 0.001. For our best performing model, we relax the semantic constraints by a factor ($d_{sc} = 0.5$) every 5K batch iterations for MM-P ($\lambda_{prop}$) and every 8.5K for MM-C ($\lambda_{cls}$), with a batch size of 128 training instances. We provide sample output detections, implementation code, and trained models in the Supplementary Material.[1]

## 4.2 Results

| mAP @ tIoU threshold | 0.3 | 0.5 | 0.7 |
|---|---|---|---|
| Wang *et al.* (2014) [26] | 14.6 | 8.5 | 1.5 |
| Oneata *et al.* (2014) [16] | 28.8 | 15.0 | 3.2 |
| Yuan *et al.* (2015) [32] | 33.6 | 18.8 | - |
| Yeung *et al.* (2016) [30] | 36.0 | 17.1 | - |
| Shou *et al.* (2016) [20] | 36.3 | 19.0 | 5.3 |
| Buch *et al.* (2017) [1] | 37.8 | 23.0 | - |
| Shou *et al.* (2017) [21] | 40.1 | 23.3 | 7.9 |
| **SS-TAD (Ours)** | **45.7** | **29.2** | **9.6** |

| Category | [20] | Ours | Category | [20] | Ours |
|---|---|---|---|---|---|
| Baseball Pitch | 0.149 | **0.193** | Hammer Throw | 0.191 | **0.416** |
| Basketball Dunk | 0.201 | **0.385** | High Jump | 0.2 | **0.22** |
| Billiards | **0.076** | 0.046 | Javelin Throw | 0.182 | **0.52** |
| Clean and Jerk | 0.248 | **0.541** | Long Jump | 0.348 | **0.717** |
| Cliff Diving | 0.275 | **0.639** | Pole Vault | 0.321 | **0.489** |
| Cricket Bowling | **0.157** | 0.151 | Shotput | 0.121 | **0.16** |
| Cricket Shot | **0.138** | 0.103 | Soccer Penalty | 0.192 | **0.264** |
| Diving | 0.176 | **0.269** | Tennis Swing | **0.193** | 0.123 |
| Frisbee Catch | 0.153 | **0.22** | Throw Discus | **0.244** | 0.074 |
| Golf Swing | 0.182 | **0.205** | Volleyball Spiking | 0.046 | **0.108** |

Table 1: Temporal Action Detection Results on THUMOS'14. SS-TAD provides state-of-the-art performance (mAP) on action detection at different temporal overlap thresholds, while still maintaining an efficient single-pass, end-to-end design. ("-" denotes not reported).

**Performance.** We summarize the results of applying our model to the task of temporal action detection on THUMOS'14 in Table 1. We observe that our model provides state-of-the-art performance for this task for both low and high overlap thresholds, which indicates that the output detections are more precisely localizing the groundtruth action intervals in the input untrimmed video sequences compared with prior state-of-the-art approaches.

**Ablation.** We verify the efficacy of our semantically-constrained memory modules by performing an ablation study on our architecture. We examine three variants of the architecture: (1) semantic constraints with relaxation during training (our full model), (2) semantic constraints with no relaxation, and (3) no semantic constraints. Other hyperparameters are fixed for fair comparison. As we see in Figure 4, adding semantic constraints reduces the number of training epochs needed to reach a particular performance level. However, if we enforce these throughout training, the overall performance of the model suffers after convergence, compared against the model trained with no semantic constraints. Overall, our full model strikes a balance between the two: by enforcing semantic constraints on the memory modules early, the model learns to tackle the overall detection task quicker, and by relaxing the semantic constraints gradually the model converges to a higher performance value.

[1] Please see https://github.com/shyamal-b/ss-tad/.

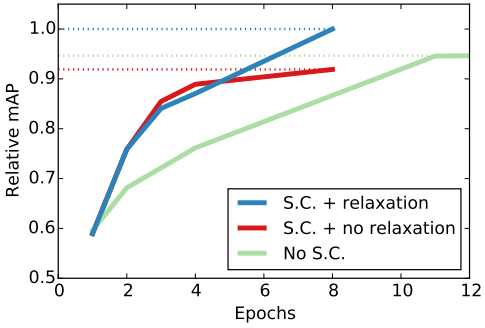


Figure 4: Results from ablation analysis of semantically-constrained (S.C.) memory modules in SS-TAD. We plot relative mAP compared to our best performing model against number of training epochs. Our best performing model leverages both semantically-constrained memory modules while also relaxing constraints during training. See Section 4.2.

**Processing Speed.** Since our model has a single-stream, end-to-end design, it is also very efficient. We benchmark our model on a GeForce GTX Titan X (Maxwell) GPU [2], and find that it operates at 701 FPS. In other words, it can process the input frame data from a 60 minute, 30 FPS video in ∼2.5 minutes. For comparison, both approaches from 2017 [1, 21] in Table 1 have significantly lower FPS for *overall* temporal action detection. Notably, both of these methods are bottlenecked by the classification and proposal stages, respectively, which were from [20]. Thus, SS-TAD offers a significant boost in both performance and processing speed. We provide additional discussion in our Supplementary Material.

**Qualitative Results.** In Figure 5(a), we provide example *positive* detections from our model on action classes such as BasketballDunk, Diving, FrisbeeCatch, and BaseballPitch (Fig. 5(a)i-iv, respectively). We observe that our model is able to provide strong temporal localization of actions in videos overall, while still maintaining a single-stream, end-to-end design. Importantly, SS-TAD is able to provide localizations over long, untrimmed video sequences. Additionally, in Fig. 5(a)iii-iv, we see that the model is able to correctly localize the actions across different camera angles and temporal distortions (e.g. slow-motion editing). Furthermore, as we observe in Figure 5(c), SS-TAD is able to provide localization and correct classification of different actions occuring in the same video. However, there remain challenging cases for the model. Videos with high frequencies of camera cuts, shots over oblique angles during the action frame, or strong temporal editing lead to weaker localization or misclassification. Two such examples are shown in Figure 5(b). We include additional discussion of these cases in our Supplementary Material, along with video visualizations of our qualitative results which more clearly illustrate these cases.

## 5 Conclusion

In conclusion, we have presented SS-TAD, a new intuitive end-to-end model for single-stream temporal action detection in untrimmed video sequences. We propose a method for training our architecture which leverages the semantic sub-tasks of temporal action detection as dynamically-adjusted semantic constraints to improve training and test-time performance.

[2] Later versions of PyTorch/newer GPU architectures (e.g. Titan X Pascal) may yield higher FPS.


Figure 5: Qualitative results of SS-TAD on THUMOS'14, including both (a,c) positive and (b) negative detections. We show a key frame from the video with the detections and groundtruth along a time axis below each image. Detections are marked as positive if the temporal overlap (tIoU) with groundtruth is greater than 0.5 and the correct action label is provided. See Section 4.2 + Supplementary Material for discussion and video visualizations.

We empirically evaluate our model on the standard THUMOS'14 benchmark, and find that our model is very efficient and provides state-of-the-art performance for temporal action detection. Future work may leverage this model as a base component for more complex video understanding tasks, such as activity understanding and dense video captioning.

## Acknowledgements

This research was sponsored, in part, by the Stanford AI Lab-Toyota Center for Artificial Intelligence Research, Toyota Research Institute (TRI), and by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research. This article reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. We thank our anonymous reviewers and De-An Huang for their comments and discussion.

# References


[1] Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. SST: Single-Stream Temporal Action Proposals. In *CVPR*, 2017.

[2] Fabian Caba, Juan Carlos Niebles, and Bernard Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *CVPR*, 2016.

[3] Wei Chen, Caiming Xiong, Ran Xu, and Jason J. Corso. Actionness ranking with lattice conditional ordinal random fields. In *CVPR*, 2014.

[4] Victor Escorcia, Fabian Caba, Juan Carlos Niebles, and Bernard Ghanem. DAPs: Deep action proposals for action understanding. In *ECCV*, 2016.

[5] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Actom sequence models for efficient action detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3201–3208. IEEE, 2011.

[6] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Activity representation with motion hierarchies. *International journal of computer vision*, 107(3):219–238, 2014.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, June 2014.

[8] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 759–768, 2015.

[9] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. Going deeper into action recognition: A survey. *arXiv preprint arXiv:1605.04988*, 2016.

[10] Mihir Jain, Jan C. van Gemert, Hervé Jégou, Patrick Bouthemy, and Cees G. M. Snoek. Action localization with tubelets from motion. In *CVPR*, 2014.

[11] 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.

[12] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Yanghao Li, Cuiling Lan, Junliang Xing, Wenjun Zeng, Chunfeng Yuan, and Jiaying Liu. Online human action detection using joint classification-regression recurrent neural networks. In *European Conference on Computer Vision*, pages 203–220. Springer, 2016.

[14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, and Scott Reed. SSD: Single Shot MultiBox Detector. *arXiv preprint arXiv:1512.02325*, 2015.

[15] Alberto Montes, Amaia Salvador, Santiago Pascual, and Xavier Giro-i Nieto. Temporal activity detection in untrimmed videos with recurrent neural networks. In *1st NIPS Workshop on Large Scale Computer Vision Systems*, December 2016.

[16] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. The LEAR submission at THUMOS 2014. 2014.

[17] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015.

[18] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[20] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage CNNs. In *CVPR*, 2016.

[21] 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. *arXiv preprint arXiv:1703.01515*, 2017.

[22] Bharat Singh, Tim K. Marks, Michael Jones, Oncel Tuzel, and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[23] Gurkirt Singh and Fabio Cuzzolin. Untrimmed video classification for activity detection: submission to activitynet challenge. *CVPRW*, 2016.

[24] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015.

[25] Jan C van Gemert, Mihir Jain, Ella Gati, and Cees GM Snoek. APT: Action localization proposals from dense trajectories. In *BMVC*, 2015.

[26] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition and detection by combining motion and appearance features. *THUMOS14 Action Recognition Challenge*, 1:2, 2014.

[27] Ruxing Wang and Dacheng Tao. UTS at ActivityNet 2016. *CVPRW*, 2016.

[28] A. Yao, J. Gall, and L. Van Gool. A Hough transform-based voting framework for action recognition. In *CVPR*, June 2010.

[29] 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. *arXiv preprint arXiv:1507.05738*, 2015.

[30] Serena Yeung, Olga Russakovsky, Greg Mori, and Li 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.

[31] Gang Yu and Junsong Yuan. Fast action proposals for human action detection and search. In *CVPR*, 2015.

[32] Jun Yuan, Yong Pei, Bingbing Ni, Pierre Moulin, and Ashraf Kassim. ADSC submission at THUMOS challenge 2015. In *CVPR THUMOS Workshop*, volume 1, 2015.