SST: Single-Stream Temporal Action Proposals

Shyamal Buch[1], Victor Escorcia[2], Chuanqi Shen[1], Bernard Ghanem[2], Juan Carlos Niebles[1]
[1]Stanford University, [2]King Abdullah University of Science and Technology (KAUST)
{shyamal, shencq, jniebles}@cs.stanford.edu, {victor.escorcia, bernard.ghanem}@kaust.edu.sa

Abstract

Our paper presents a new approach for temporal detection of human actions in long, untrimmed video sequences. We introduce Single-Stream Temporal Action Proposals (SST), a new effective and efficient deep architecture for the generation of temporal action proposals. Our network can run continuously in a single stream over very long input video sequences, without the need to divide input into short overlapping clips or temporal windows for batch processing. We demonstrate empirically that our model outperforms the state-of-the-art on the task of temporal action proposal generation, while achieving some of the fastest processing speeds in the literature. Finally, we demonstrate that using SST proposals in conjunction with existing action classifiers results in improved state-of-the-art temporal action detection performance.

1. Introduction

The millions of cameras that are deployed every year generate a large amount of recorded, stored and transmitted video. In particular, a large proportion of this video depicts events about people, their activities, and behaviors. In order to effectively interpret this data, computer vision algorithms need the ability to understand and recognize human actions. This has motivated a large body of computer vision literature on the problem of action recognition in videos.

Up until recently, the vast majority of the computer vision work tackles the action recognition problem as one of video classification, where an oracle has pre-segmented videos into short clips that contain a single action. In that case, the task is reduced to classifying the video into one of the relevant actions of interest. In practice, applications (such as smart surveillance, robotics or autonomous driving) require cameras to record video streams continuously and vision algorithms to perform temporal action detection in such long streams. To achieve this, the computer vision system must simultaneously decide both the temporal interval and the category of the action as they occur.

Temporally detecting human actions can be challenging for computer vision algorithms. Algorithms must process very long video sequences and output the starting and ending times of each action in each video. The number and duration of actions can vary significantly, and action intervals can be very short in comparison to the video length.

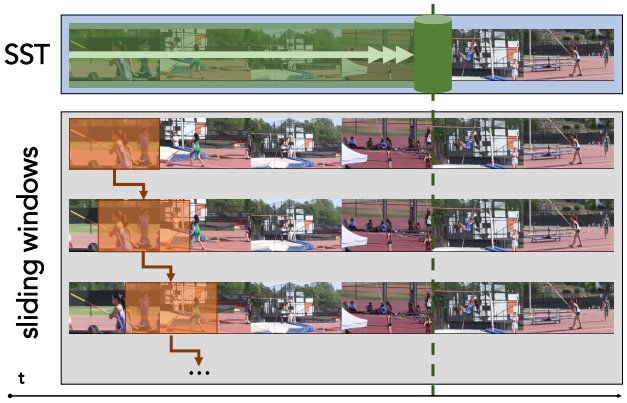

Figure 1. We tackle the problem of temporal action localization of human actions in long, untrimmed video sequences. We introduce a new model (SST) that outputs temporal action proposals at multiple scales in a single processing stream. Our method simultaneously provides stronger action proposals while being significantly more efficient than prior work, which require constructing and processing multiple temporally overlapping sliding windows.

Recent work has leveraged temporal action *proposals* [29, 2, 9] for efficient action detection, where the proposals identify relevant temporal windows that are then independently classified by an action classifier in a second stage. These proposals are generated by a sliding window approach, dividing the video into short overlapping temporal windows. To handle the issue of temporal variations in actions, windows are applied at multiple temporal scales [29]. However, this is computationally expensive due to the exhaustive search in temporal location and scale. Alternatively, one can adopt an architecture that runs a sliding window at a fixed temporal scale but outputs proposals at varied temporal scales [9]. This approach still needs to perform computations on overlapping temporal windows, which results in possibly redundant computations as each frame is

processed more than once. For many practical applications that involve large scale data or real-time interactive systems, it is critical to enable very fast action localization in videos and such redundant computations can be prohibitive.

In this paper, we introduce a framework for temporal action proposals in long video sequences that only needs to process the entire video in a single pass. This means that our architecture is able to analyze videos of arbitrary length without needing to process overlapping temporal windows separately. This results in a much more efficient and effective architecture. The main contributions of our approach are: *(i)* We introduce a new architecture (SST) for temporal action proposals generation that runs over the video in a single pass, without the use of overlapping temporal sliding windows. We design a training scheme to enable the use of a recurrent network with very long input video sequences, without compromising model performance. *(ii)* We demonstrate that our new temporal proposal generation architecture achieves state-of-the-art performance on the proposal generation task. *(iii)* Finally, we verify that SST proposals provide a stronger basis for temporal action localization than prior methods, and integration with existing classification stages leads to improved state-of-the-art performance.

2. Related Work

We review relevant recent work in video action categorization, temporal and spatio-temporal action detection in videos, and sequence modeling with recurrent networks.

Action Categorization. A large body of research has tackled the problem of action categorization from short video clips [14]. In this setting, we assume we are given short video clips with only a single action being performed in the sequence. Many approaches use a global representation [33], but some try to model the temporal structure of motions to achieve classification [10]. Unfortunately, these methods do not perform well on long video sequences where actions have a relatively small temporal duration and the majority of the visual input is considered background.

Temporal Action Detection and Proposals. A number of existing methods approach the problem of temporal action localization [19, 22, 26, 27, 31, 32, 35, 40, 24, 30, 38]. Traditionally, temporal action detection has been tackled by densely applying action classifiers in a sliding window fashion [8]. Recently, temporal action proposals [2, 9, 29] have been introduced to enable more efficient application of action classifiers on a smaller number of temporal windows. The basic idea is to first generate a reduced number of candidate temporal windows, which can be achieved with dictionary learning [2] or with a recurrent neural architecture [9]. Then, an action classifier discriminates each window independently into one of the actions of interest. Additional processing stages can also be incorporated to refine the prediction scores with respect to the temporal bounds [29].

In particular, our proposal generation framework builds on the progress made by the Deep Action Proposals (DAPs) architecture [9]. One property of DAPs is that it can retrieve action proposals of varied temporal scale by sliding a temporal window of fixed duration T. This avoids running sliding windows of multiple scales, but it still requires running an overlapping sliding window over videos longer than T. This means we need to process each input video frame multiple times, once for each window that overlaps with it. In this paper, our goal is to further reduce computation by introducing a model that processes each input frame only once and thereby processes the full video in a single pass.

Spatio-Temporal Action Detection. A related problem is that of detecting actions not only temporally but also spatially. [4, 12, 16, 34, 37, 39]. Here, the algorithms output spatio-temporal localization of actions. While these provide more detailed localization information, they tend to suffer from very high computational costs, which makes them difficult to use in cases where fast and efficient processing is necessary. Furthermore, it is possible that temporal proposals may actually help reduce the temporal search space for such algorithms. In this paper, we focus on detecting actions temporally, rather than spatio-temporally.

Object Detection. Object detection approaches have drastically improved their performance by adopting two key ideas: (1) the introduction of object proposals to replace sliding window detection, and (2) adoption of deep architecture as the learning machinery. Initial approaches adopted object proposal generation as a preprocessing stage that independently provided windows to an object classifier [11]. However, recent frameworks have implemented the generation of object proposals with deep network architectures. An example of this is the Region Proposal Network (RPN) from [28], which directly outputs proposals on an image without multiple passes. Our approach adopts this philosophy and enables proposal generation from videos in a single pass, processing each frame only once.

Long Sequence Processing with RNNs. Recurrent Neural Networks (RNNs) have recently shown impressive performance in a variety of sequential modeling problems [20]. In general, most demonstrations are limited in terms of the temporal sequence length that can be handled by RNNs at recognition time. This may be caused by the hidden state of the network becoming saturated if the input sequence is too long [21]. For example, in Natural Language Processing, it is common to use the structure of text (chapters, sections, paragraphs, sentences) to break down long corpora into short but meaningful sequences [13]. In the case of video, there is no prior access to equivalent semantic/syntactic structures. To handle long sequences, prior work for proposals [9] adopts a windowed approach such that only short subsequences are processed by RNNs. Here, we enable the use of RNNs on very long input sequences by

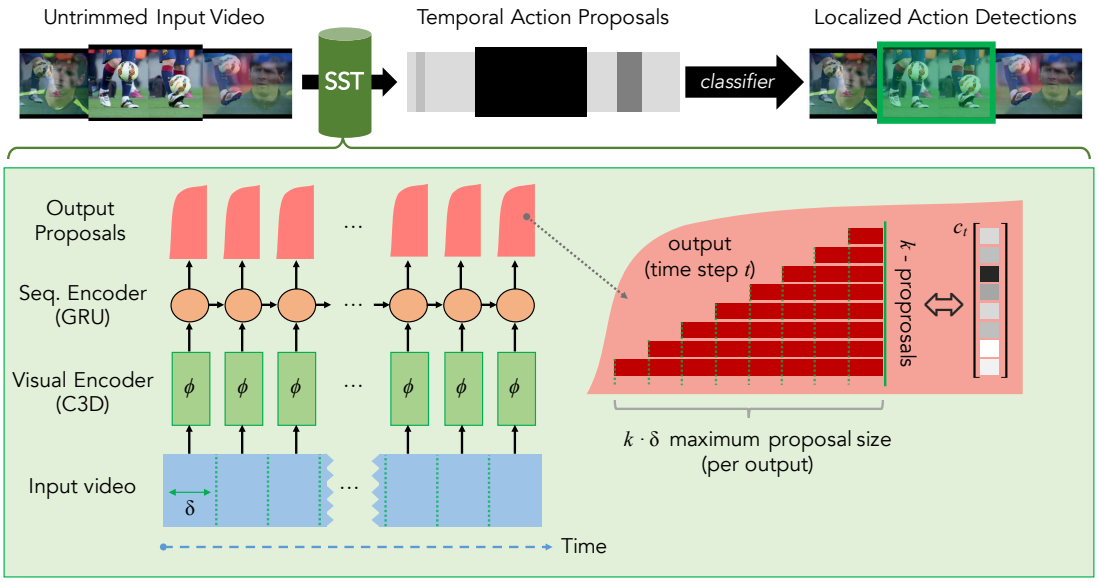

Figure 2. Schematic illustrating our overall approach and model architecture. Here, we extract C3D features from the input video stream, with a time resolution $\delta = 16$ frames for each "time step." These features are the input to the recurrent GRU-based sequence encoder model, which outputs k proposals at each time step t with a confidence vector c_t, where the longest proposal is of length $\delta \cdot k$. Additionally, we can validate the usefulness of the top-ranked SST action proposals to the action detection task by applying a classifier model.

careful architecture and training scheme design.

3. Technical Approach

The main goal of our paper is to generate temporal action proposals in long untrimmed videos. Given an input video sequence, our model should produce a reduced number of temporal intervals that are likely to contain an action. It is important for temporal action proposal methods to reject many temporal intervals that contain no action, while retrieving the true action intervals with very high recall. It is also important for the retrieved action intervals to have very high temporal overlap with the correct intervals where actions are performed. Generating high overlap proposals is key to facilitate the work of the following action classification stages. Finally, it is crucial for the temporal proposals to be fast, so that the computational gains over the simple temporal sliding window approach are significant. In this section, we introduce the technical details of Single Stream Temporal Action Proposals (SST), a new model for temporal action proposals that encapsulates these three properties in an efficient and effective deep learning architecture.

3.1. Model

We propose a recurrent model architecture for the generation of temporal action proposals. Our model is illustrated in Figure 2. In contrast with prior work, the key properties of our approach are: (i) our architecture considers the input video exactly once at *multiple* time-scales with *no overlapping sliding windows*, which results in *fast runtime* during inference; (ii) it considers and evaluates a large number of action proposals over densely sampled time-scales and locations, which results in the model producing proposals with *high temporal overlap* with the groundtruth action intervals.

Input. At inference time, our model takes as input a long, untrimmed video sequence $X = \{x_l\}_{l=1}^{L}$ with L frames. Unlike prior work that divides the video into highly overlapping temporal windows for independent batch processing, we construct no overlapping sliding windows over the input video and process each frame once, sequentially.

Visual Encoding. The goal of the Visual Encoder module is to compute a feature representation that encapsulates the visual content of the input video. We achieve this in our framework by feeding the video input through a 3D-Convolutional (C3D) network [33]. We choose C3D, as it is able to effectively capture visual and motion information over some small temporal resolution δ [33, 29]. We leverage the architecture and pre-trained weights from [33] for our initialization, with a time resolution of $\delta = 16$ frames. In this manner, we efficiently discretize the input stream into $T = L/\delta$ non-overlapping time steps. Each time step t is represented by a C3D feature encoding $v_t = \phi\left(\{x_i\}_{i=(t-1)\cdot\delta+1}^{t\cdot\delta}\right)$ taken from the top layer of the C3D network and captures visual information over δ frames. In practice, we perform PCA to reduce the dimensionality further to improve computational performance in a similar fashion as [9].

Sequence Encoding. The goal of the Sequence Encoder module is to accumulate evidence across time as the video

sequence progresses. The idea is that in order to be able to produce good proposals, the model should be able to aggregate information until it is confident that an action is taking place in the video, while simultaneously disregarding irrelevant background. At each time step $t = 1, \ldots, T$, the module receives the corresponding encoded C3D feature vector v_t as input to the recurrent sequence model.

A key property for our model is the ability to process the input video in a single pass. In order to achieve this, the recurrent model should be able to operate over the input testing video by unrolling in time over the entire duration of the video. Our training procedure in Section 3.2 is designed to facilitate the operation over long sequences at test time.

Though similar work often leverages Long Short-Term Memory (LSTM) cells for sequence encoding, we find that a Gated Recurrent Unit (GRU)-based architecture offers slightly better performance, is more robust over a wider range of hyperparameters, and has fewer parameters which means slightly faster training and test-time performance. This is consistent with empirical findings from prior work on deep recurrent models in other domains [18, 6, 7].

Thus, at each time step t, we take the hidden state h_t of the final GRU layer in the recurrent stage as our sequence encoding, where h_t is defined as per the formulation [5, 7]:

$$\begin{aligned} r_t &= \sigma_r(W_r v_t + U_r h_{t-1} + b_r) \\ z_t &= \sigma_z(W_z v_t + U_z h_{t-1} + b_z) \\ \widetilde{h}_t &= \tanh(W v_t + r_t \odot (U h_{t-1}) + b) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t \end{aligned} \tag{1}$$

where $\odot$ is the Hadamard product. Additional discussion of this formulation is in our Supplementary Material.

Output. The goal of the output module is to produce confidence scores of multiple proposals at each time step t. This module takes as input the hidden representation h_t calculated by the Sequence Encoding layer at time t. As illustrated in Figure 2, we design our architecture to score a large number of potential proposals at each time step by considering proposals of multiple time scales that end at time t. Concretely, at each time step t we output confidence scores $\{c_t^j\}_{j=1}^k$ that correspond to set of k proposals $P_t = \{(b_{t-j}, b_t)\}_{j=1}^k$, where the tuple (b_{t-1}, b_t) indicates a proposal with start and end bounds at frame b_{t-1} and b_t, respectively. The output confidence scores are given by a fully connected layer with sigmoid nonlinearity:

$$c_t^j = \sigma_o(W_o^j \cdot h_t). \tag{2}$$

All proposals considered at time t have a fixed ending boundary, and the model considers proposals of sizes $1, 2, \ldots, k$ time steps, which correspond to sizes $\delta, 2\delta, \ldots, k\delta$ frames. Note that this is done in a single forward pass at each time step, without needing to re-run the model for each temporal scale. In this manner, our model considers multiple time scales with a single pass through the input video sequence. Since we consider time scales densely at each frame, the model effectively computes proposal confidences for all proposals with a time resolution of δ frames over the video. In a way, this is effectively pushing the sliding window to the output layer in a computationally efficient manner that does not require extra computation on overlapping temporal windows. We apply standard post-processing techniques consistent with prior literature [9, 29] to select the top proposals, such as thresholding by confidence score and non-maximum suppression.

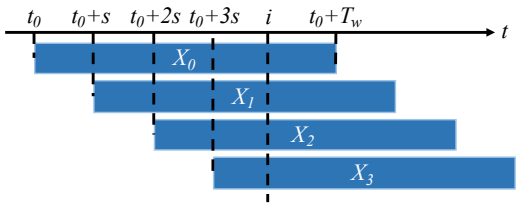

Figure 3. Training examples are generated densely by extracting temporal segments of length T_w in a sliding window fashion with stride s. Our dense sampling of long training instances helps enable the recurrent sequence encoder to fully unroll over very long video sequences at testing time.

3.2. Training

The goal of the training process is to estimate the model parameters in our architecture. By design, our proposal architecture and loss function are fully differentiable, enabling training with backpropagation. We also design the training procedure so that our recurrent networks can be fully unrolled over very long input sequences at test time. This is a key property that, in conjunction with the design of our output layer, enables the model to operate without using overlapping sliding windows over the input at test time.

This constraint means that the recurrent network must be able to properly handle inference over very long input sequences. This can prove challenging since the model must disregard irrelevant background in input untrimmed video while retaining relevant context. We observe that the hidden state of related recurrent models [9] tends to saturate when run over many steps, resulting in overconfident outputs.

Our strategy to improve robustness is motivated by better simulation of testing operating conditions during training. Briefly, we wish to provide the network with densely sampled, overlapping training video segments that are significantly longer than the temporal proposals we aim to detect.

We generate these training segments as follows: For each training video with L frames and length $T = L/\delta$ time steps, we extract segments by running sliding windows of length $T_w = L_w/\delta$ with stride s, as illustrated in Figure 3. We set $L_w \gg k\delta$, so that the training instances simulate

the operation of long sequences, encouraging the network to avoid saturation of the hidden state. This is in contrast to explicitly constructing training examples of length no longer than the maximum proposal length $k\delta$, which is the strategy used in prior work [29, 9]. Furthermore, our stride s is kept small, allowing for dense generation of training data.

The dense sampling of training segments also allows for each time step in the original video sequence to be considered multiple times with *different* contexts. For instance, consider time step $t = i$ in Figure 3. The visual content and groundtruth observed at time t is part of several training segments $X_0, \ldots, X_3$. This means that during the training process, we will be able to backpropagate the training loss at $t = i$ within the context of four examples $X_0, \ldots, X_3$. When considering X_0, the hidden state of the sequence encoder at $t = i$ would be $h^0_{t_0:i}$; likewise the hidden state at $t = i$ for X_1 would be $h^1_{t_0+s:i}$. In both cases, the training process would backpropagate the loss at $t = i$ with different contexts given by the hidden state in each example, encouraging the prediction and encoding to be robust to the specific initializations of the hidden state.

Each training example is associated with groundtruth labels that indicate which temporal intervals correspond to actions in the video. The idea is that our network will classify each temporal interval in consideration as a positive or negative action proposal. For example, consider X_0 in Figure 3, which we associate with groundtruth labels $Y_0 = \{y_t\}_{t=t_0}^{t_0+T_w-1}$. At time step t, the groundtruth y_t is a k dimensional vector with binary entries. The j-th entry y_t^j is set to 1 if the corresponding proposal interval (of scale j time steps) has a temporal Intersection-over-Union (tIoU) with the groundtruth larger than 0.5 and set to 0 otherwise.

During training, we penalize the network for errors according to a multi-label loss function. In practice, for a training video X the loss at time t is given by a weighted binary cross entropy objective:

$$\mathcal{L}(c, t, X, y) = -\sum_{j=1}^{k} w_0^j y_t^j \log c_t^j + w_1^j (1 - y_t^j) \log(1 - c_t^j), \tag{3}$$

where the weights w_0^j, w_1^j are calculated according to the frequency of positive and negative proposals in the training set at each scale j and c is the output of the network. We add dropout and ℓ_2 regularization on the learned parameters.

Our model backpropagates at every time step t, so the total loss for all training examples $\mathcal{X}$ is:

$$\mathcal{L}_{train} = \sum_{(X,y)\in\mathcal{X}} \sum_{t=1}^{T_w} \mathcal{L}(c, t, X, y). \tag{4}$$

4. Experiments

We empirically evaluate the effectiveness of our temporal proposal method for the task of proposal generation as well as its application to temporal action detection. As our experiments will show, our proposal method achieves competitive performance at faster computing speeds. We describe our experimental settings and results here.

Dataset. To train and evaluate our model, we use the temporal action localization subset from the THUMOS'14 dataset [17], which contains 20+ hours of video with 200 validation and 213 test untrimmed video sequences. We use the validation videos as our training set, as is standard practice on this dataset. To enable direct comparisons with prior work, we adopt the experimental settings from [9].

We perform an 80%-20% split over the training examples in the dataset to cross-validate the hyperparameters for our model, ensuring that the distribution of activity classes is approximately the same. For our generalizability analysis, we leverage subsets of unseen classes in the ActivityNet dataset [3]. Further details in Section 4.1.

Comparisons. We compare our SST model with Deep Action Proposals (DAPs) [9], the proposal stage of S-CNN (SCNN-prop) [29], BoFrag [25], and Sparse-Prop [2].

Implementation details. We generate training data with $L_w = 2048$. We vary the number of recurrent layers and hidden state size, as well as number of proposals k. We implement the model and training/validation pipeline using Lasagne/Theano and Caffe, with training executed on GeForce TITAN X (Maxwell) GPUs. We optimize our model parameters with backpropagation using the *adam* update rule [23], with an initial learning rate of $5 \cdot 10^{-2}$ annealed after every $l = 5$ epochs. We include further analysis of hyperparameters, trained models, code, and sample output proposals as part of our Supplementary Material[1].

4.1. Temporal Proposal Generation

The task of temporal proposal generation consists of taking an input video and producing a set of temporal intervals that are likely to contain human actions. A successful proposal method should be able to retrieve action intervals with very *high recall* and *high temporal overlap* (tIoU) with the true temporal segments that correspond to actions, while only producing a small number of proposals. Additionally, it is key for the model to have *fast runtime*. We evaluate these three aspects of our method below.

First, we consider the ability of our model to retrieve proposals with high recall. We measure this with the proposal average recall. This is computed by extracting a fixed number of proposals and calculating the average recall over a range of tIoUs. We plot average recall against number of retrieved proposals in Figure 4(center) for tIoUs in the range 0.7 - 0.95, and Figure 4(left) for tIoUs in the range 0.5 - 1.0 (for consistency with [9]). We observe that our model outperforms all existing state-of-the-art methods for low and

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

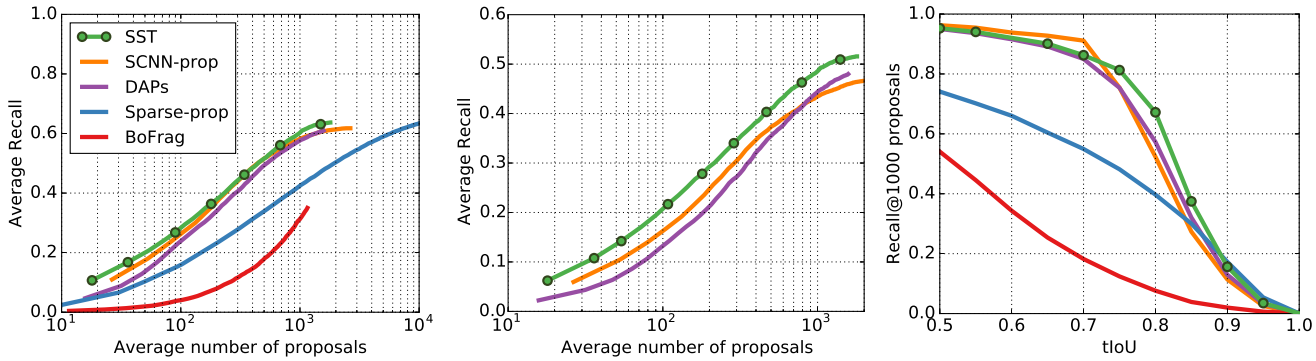

Figure 4. Comparison of our proposal network with state-of-the-art proposal localization methods. SST offers strong performance against prior literature, even though it constructs no overlapping sliding windows at test time and passes through the input in a single stream. **(left)** SST has higher average recall and requires fewer proposals. **(center)** This difference is particularly visible when average recall is computed over a higher tIOU range (0.7-0.95). For clarity, we show results here for the top three proposal methods. **(right)** Recall @ 1K proposals vs. tIoU plot shows that SST has the largest improvements around tIoU ≈ 0.8.

	Recall		
Method	tIoU= 0.6	tIoU= 0.8	FPS
DAPs	0.916	0.573	134
S-CNN-prop	**0.938**	0.524	60
SST (Ours)	**0.920**	**0.672**	**308**

Table 1. Comparison of proposal generation performance with prior state-of-the-art in terms of recall at 1000 proposals. We observe our method offers comparable performance for lower tIoU thresholds, outperforms for higher thresholds, and offers a significant boost in proposal speed. Note that we used the older GPU set-up of [9] to ensure fair comparison. We discuss benchmarks on newer GPU architectures in Supplementary Material.

high number of proposals. Also, we note that when operating at the high overlap regime in Figure 4(center), our model more significantly outperforms prior work.

Second, we consider the ability of the model to retrieve proposals with high tIoU overlap. Figure 4(right) plots proposal recall for our method in comparison to prior work. We note that our model performs comparably with competing approaches at the lower tIoU range, but more importantly, our method performs better at the higher tIoU regime. This is key, since it means our method can retrieve proposals that more tightly capture the true temporal action intervals.

Finally, we study the runtime speed of our method in comparison to alternative approaches in the literature. To achieve this, we measure runtime speed in frames per second (FPS). In comparison to prior work that relies on multi-scale temporal sliding windows [29] or single scale temporal sliding windows [9], our single-pass model achieves significantly faster processing speeds, as shown in Table 1.

Robustness to Video Length. An important goal for our architecture is the ability to handle very long testing video sequences. As outlined above, the idea is that our recurrent model should be able to unroll over the entire duration of the testing video, regardless of its duration, so that the proposals are generated in a single pass through the video. We achieve this by two aspects of our model: dense predictions at each frame, and a training scheme that encourages robustness into the model with respect to video length.

We analyze the performance of our model to highlight its robustness from three perspectives. For this analysis, we select the operating point of 1000 retrieved proposals. First, we study the recall stability with respect to the temporal location of the proposal middle frame, which we plot in Figure 5(left). Note that our model processes the entire video by unrolling a single recurrent network, so the longer the video, the more time steps the recurrent network processes. We observe that SST recall performance is stable and nearly independent of the temporal location of the proposal.

Second, we study the recall stability with respect to video length. Here, we compute recall per video and compute average recall for videos with similar length. We also observe stable behavior with respect to video length as shown in Figure 5(center).

Finally, we analyze recall performance with respect to proposal length. We would like to analyze if longer action sequences are harder to detect than shorter actions. We plot recall against proposal length in Figure 5(right). Again, we observe that recall performance is also stable with respect to length of the groundtruth annotations we wish to localize.

We also note that some videos in the THUMOS testing set are particularly long, with durations of over 20 minutes. This corresponds to video sequences of more than 30-50 thousand frames. Our qualitative evaluation confirms our empirical observation that our network can unroll over very long testing videos without compromising its performance.

Qualitative Results. We generate sample proposals

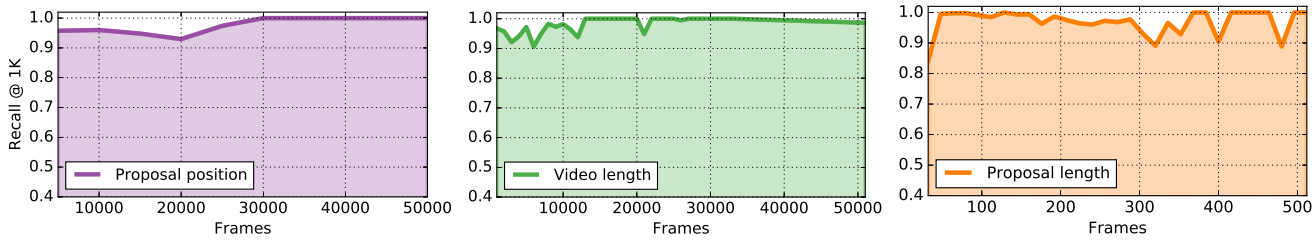

Figure 5. We evaluate the recall stability and robustness of our model against **(left)** groundtruth proposal temporal position, **(center)** video length, and **(right)** groundtruth proposal length. We observe that SST is indeed able to handle long, untrimmed video sequences without compromising performance, while gaining a dramatic boost in efficiency.

	ActivityNet		*ActvityNet* $\leq$ *1024*	
Method	@500	@600	@500	@600
DAPs [9]	0.236	0.257	0.396	0.433
SST	**0.242**	**0.288**	**0.423**	**0.494**

Table 2. Summary of generalization analysis in ActivityNet in terms of average recall for @k average proposals. We observe that proposals generated by SST are comparable or outperform prior proposal methods for unseen activities. For example, our proposals offer a relative improvement of +3.1% for the general unseen dataset **(group 1, col. 2)**, and +6.1% over DAPs segments for unseen activities that span up to 1024 frames **(group 2, col. 2)**.

	Action detection (mAP)				
Method	@50	@100	@200	@500	@1000
Sparse-prop[2]	5.7	6.3	7.6	**8.2**	8.0
SCNN-prop[29]	5.6	7.7	10.5	**13.57**	13.45
DAPs[9]	8.4	12.1	**13.9**	12.5	12.0
SST	*10.9*	*13.2*	**13.94**	13.1	13.1

Table 3. Summary of action detection results with VLAD-based SVM classifier from [9] against number of proposals. Best value for each method shown in bold. SST outperforms prior work with fewer proposals needed, indicating that SST outputs high-quality proposals within a limited budget.

Method	**Action detection (mAP)**
S-CNN (full system) [29]	0.19
SST + (S-CNN classifier)	**0.23**

Table 4. Applying S-CNN *classifier* [29] on top of SST proposals, we exceed the prior state-of-the-art action detection results on THUMOS'14, *without* any fine-tuning of the classifier. Thus, our SST approach provides a more efficient and effective way to extract precise video segments where actions are detected more accurately even with existing classifiers.

from our model in Figure 6. We see that the model localizes the groundtruth annotations well - our top confidence proposals satisfy tIoU criteria well, and our highest overlap proposals have high confidences. We observe that among false positive detections, a common case occurs where our model outputs a high-confidence "umbrella proposal" over short action sequences tightly packed together with brief periods of non-action between them (although the model does output other high-confidence proposals correctly localizing those proposals, just with slightly lower confidence).

Generalizability of Proposals. Another key characteristic of action proposal approaches is their capability to generate segments for unseen action categories [9, 15, 1]. Following the observations found in the analysis of object proposals [15], Escorcia *et al*. proposed to evaluate the average recall of temporal action proposal methods on a diverse set of action classes unseen during training [9]. Thus, we can assess the *generalizability* of these proposal approaches.

Table 2 summarizes the performance of our approach in this scenario. For fair comparisons, we used the same experimental protocol used in [9]. We report the results in two subsets of the validation set of ActivityNet v1.2 [3]. These subsets correspond to: (i) "*ActivityNet*" results over the validation set and (ii) "*ActivityNet* $\leq$ *1024*" results for videos with actions from categories not present in [17] on *annotations* that span up to 1024 frames. We observe our method exhibits a comparable or better degree of generalization than [9]. Additional results and example video are available in the Supplementary Material.

4.2. Action Detection with SST proposals

Finally, we apply our proposal generation architecture to the task of temporal action localization. The goal is not only to localize the temporal intervals where the actions happen, but also to label the interval with the correct action category.

For direct comparison to prior state-of-the-art proposal methods, we implement the approach of Xu *et al*. [36, 9]. Briefly, for each temporal segment, we encode the corresponding C3D features from the *fc7* layer using *VLAD*. We then train a one-vs-all linear SVM classifier for each class C. We evaluate the mean AP (mAP) with 0.5 tIoU overlap threshold on the THUMOS'14 test set.

We summarize these standardized action detection results in Table 3. We observe that SST consistently outperforms other proposal methods for lower number of proposals, and our mAP@200 proposals (13.94%) matches or out-

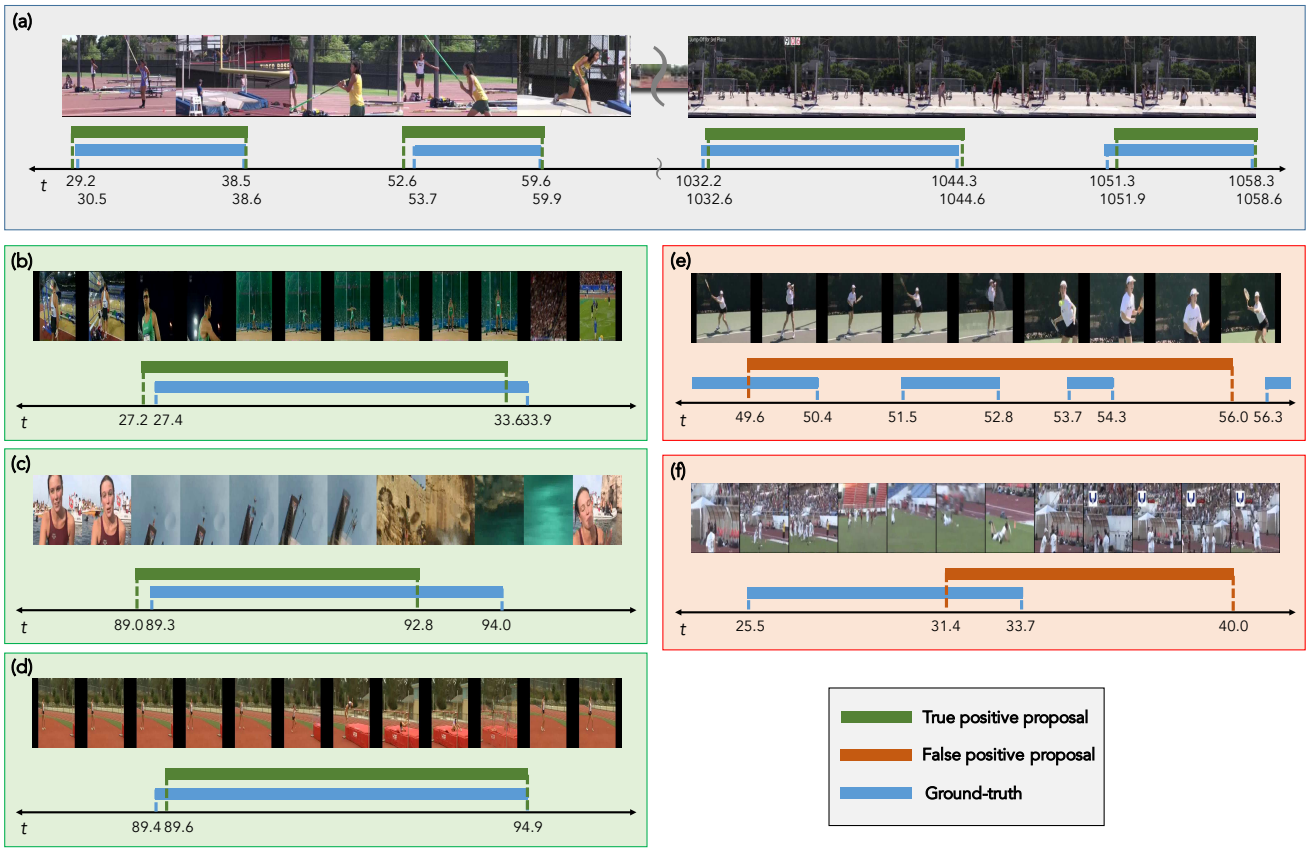

Figure 6. Qualitative results of our SST model on THUMOS'14. Time measured in seconds. **(a)** Proposals generated by SST on a long, untrimmed video sequence. Each groundtruth annotation is coupled with the best proposal on top. We observe that performance of the network is maintained for the full duration. **(b-d)** Performance of the *top-ranked* temporal action proposal retrieved for a given input video sequence, coupled with the nearest groundtruth annotation. SST provides tight localization bounds with high tIoU. **(e-f)** False-positive results for the top-ranked retrievals. In particular, (e) illustrates an issue where the model will sometimes rank "umbrella proposals" which encompass several short, consecutive action sequences with a slightly higher confidence than the individually-localized proposals.

performs all other methods, regardless of proposal number.

Finally, we demonstrate that by applying the previous state-of-the-art classification stage from [29] to SST proposals, we significantly improve temporal action localization. As shown in Table 4, the full S-CNN action detection architecture had the prior state-of-the-art detection performance of 0.19 mAP [29], while application of the same classifier stage to SST proposals results in 0.23 mAP. Thus, SST provides a strong base for temporal action localization. Moreover, we demonstrate the importance of improving proposal methods for overall temporal action detection.

5. Conclusions

We have introduced a new architecture, SST, for temporal action proposals that can operate on long video sequences at high processing speeds. Our approach processes input video data as a single stream, and constructs no overlapping sliding windows at evaluation time. We demonstrate that our model achieves state-of-the-art performance on the action proposals task, and, when considered as part of a full detection architecture, provides effective action localization performance with fewer proposals.

6. Acknowledgments

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, De-An Huang, Oliver Groth, Fabian Caba, Joseph Lim, Jingwei Ji, and Fei-Fei Li for helpful comments and discussion.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 73–80, 2010. 7

[2] F. Caba, J. C. Niebles, and B. Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *CVPR*, 2016. 1, 2, 5, 7

[3] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. ActivityNet: a large-scale video benchmark for human activity understanding. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 961–970, 2015. 5, 7

[4] W. Chen, C. Xiong, R. Xu, and J. J. Corso. Actionness ranking with lattice conditional ordinal random fields. In *CVPR*, 2014. 2

[5] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111, 2014. 4

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 4

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 4

[8] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *ICCV*, 2009. 2

[9] V. Escorcia, F. Caba, J. C. Niebles, and B. Ghanem. DAPs: Deep action proposals for action understanding. In *ECCV*, 2016. 1, 2, 3, 4, 5, 6, 7

[10] A. Gaidon, Z. Harchaoui, and C. Schmid. Activity representation with motion hierarchies. *International journal of computer vision*, 107(3):219–238, 2014. 2

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, June 2014. 2

[12] G. Gkioxari and J. Malik. Finding action tubes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 759–768, 2015. 2

[13] A. Graves. Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 5–13. Springer, 2012. 2

[14] S. Herath, M. Harandi, and F. Porikli. Going deeper into action recognition: A survey. *arXiv preprint arXiv:1605.04988*, 2016. 2

[15] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *PAMI*, 2015. 7

[16] M. Jain, J. C. van Gemert, H. Jégou, P. Bouthemy, and C. G. M. Snoek. Action localization with tubelets from motion. In *CVPR*, 2014. 2

[17] 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. 5, 7

[18] R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. *Journal of Machine Learning Research*, 2015. 4

[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 6, 2014. 2

[20] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3128–3137, 2014. 2

[21] A. Karpathy, J. Johnson, and F. Li. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078, 2015. 2

[22] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 2

[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[24] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1942–1950, June 2016. 2

[25] P. Mettes, J. van Gemert, S. Cappallo, T. Mensink, and C. Snoek. Bag-of-fragments: Selecting and encoding video fragments for event detection and recounting. In *ACM International Conference on Multimedia Retrieval (ICMR)*, 2015. 5

[26] D. Oneata, J. Verbeek, and C. Schmid. The lear submission at thumos 2014. 2014. 2

[27] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 612–619, 2014. 2

[28] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 2

[29] Z. Shou, D. Wang, and S. Chang. Temporal action localization in untrimmed videos via multi-stage CNNs. In *CVPR*, 2016. 1, 2, 3, 4, 5, 6, 7, 8

[30] 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 *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1961–1970, June 2016. 2

[31] Y.-C. Su and K. Grauman. Leaving some stones unturned: Dynamic feature prioritization for activity detection in streaming video. *arXiv preprint arXiv:1604.00427*, 2016. 2

[32] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 371–380. ACM, 2015. 2

[33] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 2, 3

[34] J. C. van Gemert, M. Jain, E. Gati, and C. G. Snoek. Apt: Action localization proposals from dense trajectories. In *BMVC*, 2015. 2

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

[36] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2015. 7

[37] A. Yao, J. Gall, and L. V. Gool. A hough transform-based voting framework for action recognition. In *CVPR*, June 2010. 2

[38] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and F. Li. Every moment counts: Dense detailed labeling of actions in complex videos. *CoRR*, abs/1507.05738, 2015. 2

[39] G. Yu and J. Yuan. Fast action proposals for human action detection and search. In *CVPR*, 2015. 2

[40] J. Yuan, Y. Pei, B. Ni, P. Moulin, and A. Kassim. Adsc submission at thumos challenge 2015. In *CVPR THUMOS Workshop*, volume 1, 2015. 2