CS 231N Final Report: Single-Stream Action Proposals in Videos

Ines Chami*
Institute for Computational and Mathematical Engineering
Stanford University
chami@stanford.edu

Abstract

Video understanding is a subject of high interest that has many applications such as human-robot interaction or self-driving cars. Temporal action proposals consist to produce temporal segments that are likely to contain an action of interest. This is very important for any video related task since it allows to focus on the temporal segments that contain important information. Single-Stream Temporal Action Proposals (SST) [1] is a recent algorithm that can localize actions in long videos in a single pass. In this project we implement the SST model and evaluate it's performance on the ActivityNet [3] dataset. We also conduct an analysis of the different parameters in the model and how they impact performance and time complexity.

1. Introduction

Video understanding is of particular interest nowadays because of the massive growth of video data available online. For instance, in 2016, more than 300 hours of video were uploaded to YouTube every minute. In order to understand and analyze all these videos, scientist must develop efficient algorithms that can analyze important actions in videos and extract semantic information. Building efficient algorithms is crucial for large scale systems as we do not have the computational power analyze all the video data available online and this is what motivated researchers to create systems for temporal proposals generation.

Temporal proposals systems aim at reducing the computational cost and improve the performance of video related tasks. The goal is to retrieve temporal segments that are likely to contain an action of interest. That is, rather than exploring an entire video to extract semantic information, scientist can now use these temporal proposals to directly focus on the relevant content. For instance in Figure 1, a temporal proposals system would return only segments where an action of interest happens such as "Rafael Nadal and Novak Djokovic are playing tennis" or "Rafael Nadal is raising his arms after winning the game".

Figure 1. Example of action proposals for a tennis game video. The SST model outputs temporal segments that contain actions in the video.

Temporal action proposals were first introduced by Jain et al. [11] who built a model that could produce spatio-temporal object proposals for video. Their work was motivated by the boost in performance brought by object proposals systems for visual tasks. However more recently, scientist have been focusing on the temporal dimension of a video, which is believed to contribute more to the semantic of a video than the spatial dimension.

Previous work on temporal proposals was mainly focused on sliding windows methods [16, 2]. These methods consist to explore several regions of the video at various scales and produce an action score using a separate classifier. However, these sliding window approaches can be very inefficient at test time since the video can not be processed in one single pass and every frame of the video has to be processed several times for each sliding window.

*This project was conducted with Ranjay Krishna

More recently, Escorcia et al. [5] proposed a Deep Action Proposals Architecture (DAPs) that can produce temporal proposals in one pass at varied temporal scales for a fixed video duration T. However when the video duration D is greater than T, this model still needs to process the video $D - T + 1$ times to generate proposals and some frames would have to be processed several time.

Single-Stream Temporal Action Proposals (SST) is a recent model for temporal action proposals in long videos that can run continuously. SST takes as input long video sequences and outputs several starting and ending times for actions in videos, all in one pass. SST achieves state-of-the-art performance on the THUMOS14 dataset [12]. In this project we propose to implement the SST architecture, analyze the impact of each parameter and evaluate the model's performance on the large scale ActivityNet dataset [3].

In section 2, we present an overview of the work related to temporal proposals generation. We then detail the SST model in section 3 and present our experiments in section 4. We compare SST to the DAPs approach on the ActivityNet dataset based on the Recall@K metric and show that SST achieves states-of-the-art performance. We propose an implementation of the SST module using the Pytorch Framework (https://github.com/pytorch/pytorch).

2. Related Work

In this section, we present the work that is most related to the SST model. Video action proposals generation shares with **object proposals** the high-level goal of generating proposals that contain important information. In video proposal, the goal is to find temporal segments that are likely to contain actions. Similarly in object proposal, the goal is to propose image regions that are likely to contain objects. Recent deep learning methods were introduced for object detection. Girshick et al. [8, 7] developed a system that can generate a fixed number of candidate and classify each region proposals using class-specific linear SVMs. This work significantly improved the state-of-the-art performance for object detection, especially on large scale challenges [15, 6] as it allowed to significantly reduce the computational complexity of this task by focusing on a smaller set of regions. Furtheremore, object proposals have also brought significant boost in performance for image related task such as image segmentation [9, 10] or image captioning [13]. In [13], the authors first use an object proposals module to generate 2000 regions in an image and extract the top-19 detected location. They extract a feature vector with a CNN for each region proposals and use this set of vectors as a new image representation to generate image captions.

Similarly with action proposals modules, one could easily imagine a similar scheme where an action proposals module is first applied to a video to extract a new representation and then use this representation for **video captioning** or any other video related task.

This motivated research in the domain of **temporal action proposal** and Shou et al. [16] introduced a proposal network using sliding windows. Their system generates temporal sliding windows with varied lengths and then classify segments into background or action classes using a C3D network. However, the main disadvantage of sliding window methods is that each video frame has to be processed several times for each temporal scale. This is very inefficient in terms of computational complexity since some computations are redundant for overlapping segments and therefore, these methods would not scale well for large datasets.

More recently, Escorcia et al. [5] introduced a Deep Action Proposals (DAPs) framework for action understanding. Their system can localize proposals of varied lengths without conducting an exhaustive search over different temporal scales. Their model consist to encode video frames using a visual encoder and forward these using a long-short term memory (LSTM) network. They then compute a linear combination of the last hidden state to output multiple proposals of varied lengths. That is, DAPs can generate action proposals of multiple temporal scales in one pass for a fixed video length T. This work significantly outperformed previous sliding window methods in terms of performance (i.e retrieve proposals with a high recall) and computational efficiency. However as mentioned in the introduction, when the video duration is greater that T, DAPs has to process video frames several times and SST was introduced as an amelioration of DAPs to overcome this issue.

This leads us to the recent work of Krishna et al. [14] who combine a proposal module (DAPs) and a captioning module for dense captioning events in video. For each detected proposal, they extract the hidden representation from the LSTM in the proposal module an use this as input for their captioning module. The captioning module then generates captions for each proposal while utilizing the context from other events in the video. This results in the generation of multiple sentences for a video, that all relate to each other.

Finally, action proposals models have a lot in common with **action detection** systems that aim at localizing and classifying actions in videos. That is, given an input video, an action detection module would produce temporal segments and the corresponding action category. Previous work used sliding window modules for action proposals and then trained separate classifiers for action detection [17, 12]. Similarly, DAPs or SST can also be used as a proposal module for action detection. Both approaches outperform prior action detection models and detect high-quality actions efficiently.

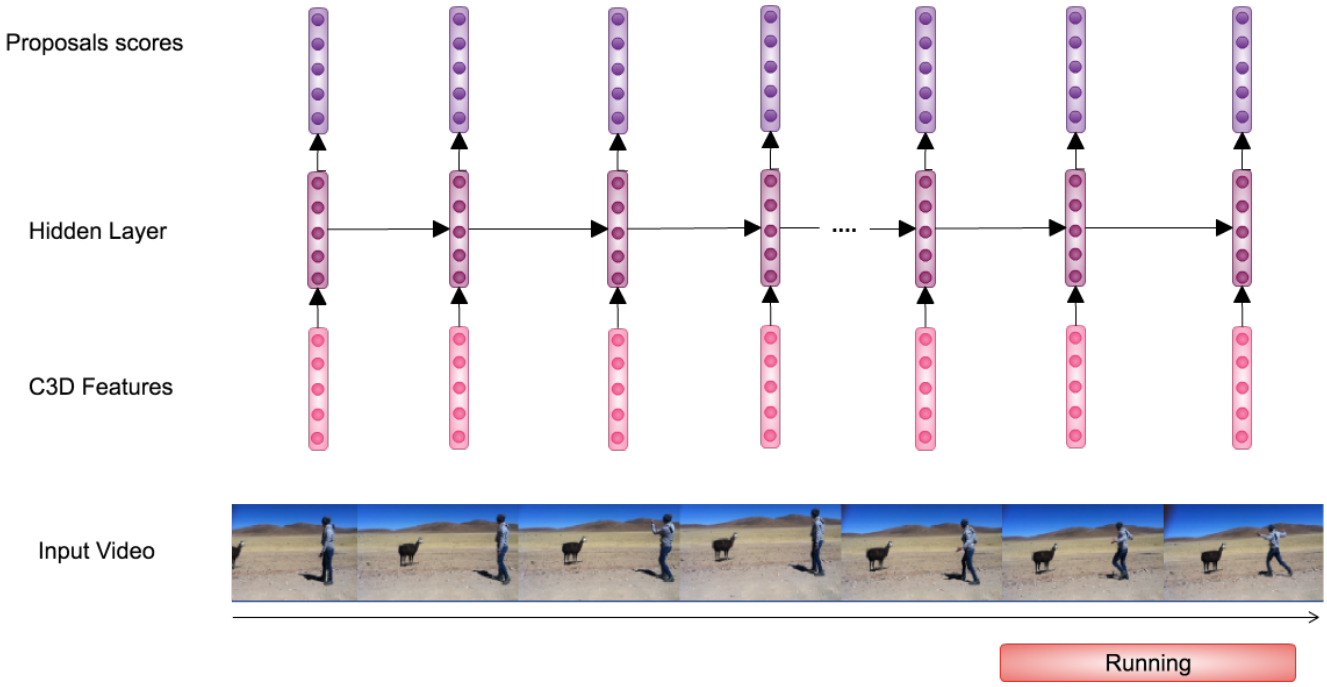

Figure 2. Schematic illustration of the SST architecture. The video frames a first encoded using a visual encoder. The encoded frames are then used as input for a Gated Recurrent Unit network that produces proposal scores for each timestep t.

3. Method

3.1. The SST model

In this section, we present the SST model. SST was recently proposed for temporal action proposals. Given an input video, the model produces temporal intervals that are likely to contain actions. The novelty of this model lies in the fact that the proposals are generated continuously, in a single forward pass and that it can operate on long input videos. This leads to significant speed up compared to previous models that used sliding windows. In the SST model, the videos are first encoded using a visual encoder 3.2 and then passed through a recurrent network 3.3 that outputs proposal scores 3.4 for each timestep t.

3.2. Visual Encoder

Given an input video sequence, the first step in the SST model is to encode the videos into a feature representation that contains relevant information about the input video. In practice, the authors use the top layer of a 3D convolution (C3D) network trained for action classification [18]. The C3D network encodes the video frames with a temporal resolution of δ frames. For instance, if a video sequence contains N frames, the visual encoding will result in $T = \frac{N}{\delta}$ feature vectors $\{x_i\}_{i=1}^{T}$ where x_i has the same dimensionality as the top layer of the C3D network.

3.3. Network architecture

The SST architecture relies on a recurrent network architecture and more specifically on Gated Recurrent Unit (GRU) [4], that tends to saturate less than recurrent neural networks for long sequences and has fewer parameters than LSTM networks. The GRU encodes the visual features $\{x_i\}_{i=1}^{T}$ into hidden representations $\{h_i\}_{i=1}^{T}$ using the following recurrent formula:

$$r_t = \sigma_i(x_t W_r + h_{t-1} U_h + b_r) \quad (1)$$

$$z_t = \sigma_u(x_t W_z + h_{t-1} U_h + b_z) \quad (2)$$

$$\widetilde{h}_t = \tanh(x_t W_h + r_t \odot h_{t-1} U_h + b_h) \quad (3)$$

$$h_t = (1 - z_t) \odot \widetilde{h}_t + z_t \odot h_{t-1} \quad (4)$$

$$(5)$$

Where r_t is the reset gate, z_t is the update gate and $\widetilde{h}_t$ is the new memory. GRUs have a more persistent memory and can therefore capture long term dependencies, which enable them to work well for long videos. The recurrent architecture is very appropriate in this setting since it allows to encode the video sequentially while keeping track of the previous timesteps. A key property of the SST model is that it can produce proposals in one forward pass and we detail the method to achieve this in the next section.

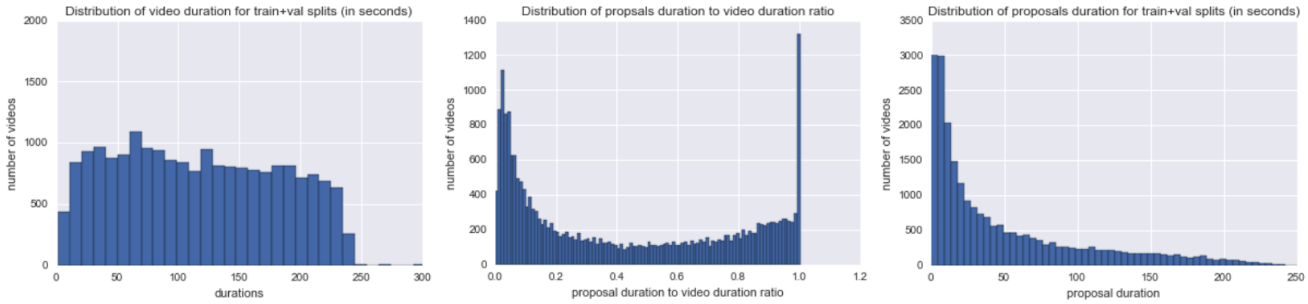

Figure 3. Data statistics for videos and proposals in the ActivityNet dataset

3.4. Proposals scores

As mentioned before, the novelty of SST lies in the fact that it can run continuously and produce proposals for varied time scales. This property is due to the GRU architecture as the network is trained to produce multiple proposal scores for each timestep. More concretely, for each timestep t, the GRU outputs K proposal scores $\{\hat{s_t}^j\}_{j=1}^K$:

$$\hat{s_t} = \sigma(h_t W_{hs}) \tag{6}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function and $\hat{s_t}^j$ is a score representing the likelihood that a proposal starts at time $t-j$ and ends at time t. Therefore at test time, SST will produce proposals in a single pass, since all time scales are taken into account in the output score $\hat{s_t}$.

3.5. Training the SST network

In order to train the SST model, the authors propose to densely generate training sequences by extracting temporal segments of length W with some stride. This allows to generate more training examples and consider each timestep in different context during training. In practice, W is chosen in order to satisfy the criterion $W >> K$ so that the network can operate on long sequence at test time. In our SST implementation, we propose to randomly sample segments of length W in the video for each batch to reduce memory storage and speed-up the training procedure. Once the segments of length W are generated, we compute a weighted multilabel cross-entropy loss at each timestep t:

$$L_1(s_t, \hat{s}_t) = -\sum_{j=1}^{K} w_0^j s_t^j \log(\hat{s_t}^j) + (1-w_0^j)(1-s_t^j)\log(1-\hat{s_t}^j)$$

Where s_t are the ground truth labels, $\hat{s}_t$ are the predicted labels and $\{w_0^j\}_{j=1}^K$ are the proportions of negative examples for each proposal length j. In practice, L2 regularization of the weights is also added to the objective to prevent overfitting. This weighting of the loss improves learning since the generated labels can be very sparse. We detail the label generation procedure in section 4.3.

4. Experiments

4.1. Dataset

We use the ActivityNet 200 [3] benchmark from the Large Scale Activity Recognition Challenge (http://activity-net.org/challenges/2016/). This dataset was established to encourage research in the field of human activity understanding. It contains 19,994 videos (10,024 for training, for 4,926 for validation and 5,044 for testing). Each video is labelled with timestamp proposals for activities and each proposals is labelled with one of 200 activity categories. The C3D features detailed in section 3.2 are also provided with the ActivityNet Benchmark. In practice, the C3D features dimensionality is reduced using principal component analysis, resulting into 500-dimensional vectors. In figure 3.3, we plotted the distribution of the proposal lengths and the video durations, as well as the distribution of the proposal lengths to video lengths ratios. We can observe that an important amount of videos have proposal lengths to video lengths ratios close to 1 and those videos would not be good training examples for action localization in long videos. We can also see from the plot that the video durations are approximately uniformly distributed between 0 and 120 seconds and that the proposals durations have a decay similar to an exponential decay. Based on these statistics, we can compute W and K in order to capture a good proportion of proposals in the model. Furtheremore, we conducted experiments in section 4.4 to study the impact of W and K on SST.

4.2. Evaluation Metrics

Following the work of Buch et al. [1], we compute Recall@K to evaluate performance. Concretely, for a test video, we extract the top-K proposals that were gener-

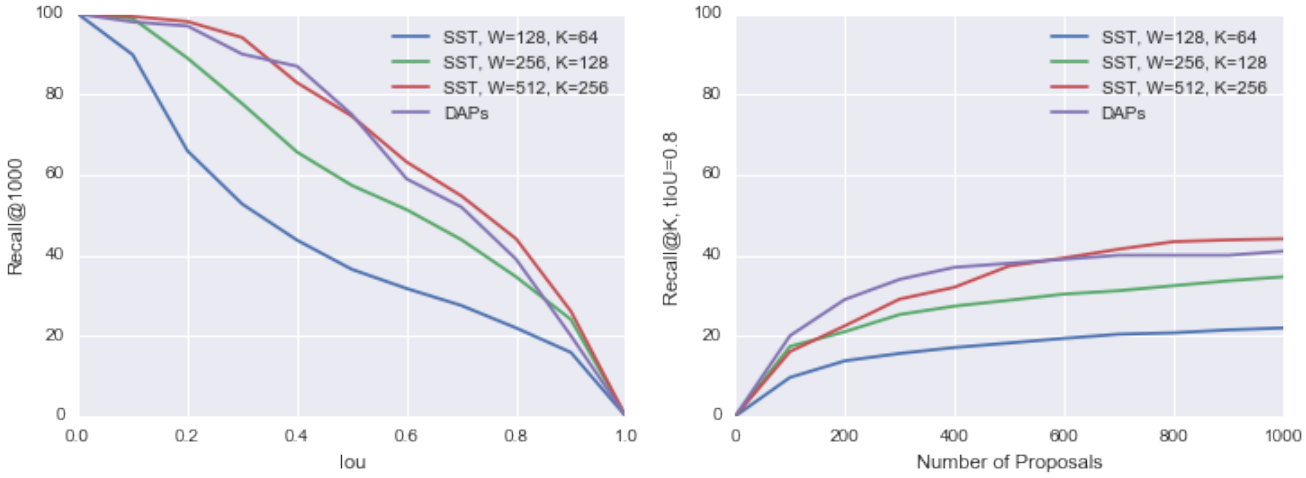

Figure 4. Recall@1000 as a function of IoU (left) and Recall@K for IoU=0.8 as a function of K, the number of proposals (right)

ated by the SST model and compute their intersection of union (IoU) with the ground truth proposals. If IoU is greater than some threshold, we consider the proposal as a true positive. We then compute recall which is simply the fraction of ground truth proposals that were retrieved by the system. In practice, we compute Recall@K for K $\in \{100, 200, \ldots, 1000\}$ for IoU ranging from 0.1 to 0.9.

We also conducted experiments to measure time performance of the SST model. In order to do so, we compute the average number of frames per second (FPS) that can be processed by the system (high is better).

4.3. Labels Generation

The action proposals in ActivityNet are provided as time intervals in seconds. Given an input video that has T timesteps (T is the number of C3D features computed in section 3.2), we need to convert these proposals into a TxK matrix where the (t,k) element indicates whether the proposal ($t-k$,t) contains an action or not. To do so, we compute the IoU for each ($t-k$,t) interval with all the ground truth proposals. If the maximum IoU is greater than some threshold IoU_{min} we label the proposal ($t-k$,t) with 1, otherwise we label it with 0. Note that $\frac{K\delta}{\text{IoU}_{min}}$ is the maximum proposal length that can be captured with this model. We therefore need to judiciously select the parameter K in order to capture a good proportion of proposals. In section 4.4, we conducted experiments to analyze how the parameters W and K influence the SST model.

4.4. Performance and Time Complexity Threshold

The parameters K and W characterize the maximum proposal length and the maximum video length that can be captured by the SST model. We therefore need to judiciously select theses parameters K in order to capture a good proportion of proposals. In this section, we compare the SST performance as well as running times for varied values of K and W.

In general, we want W to be smaller than the number of steps in the video to generate multiple training examples in different contexts. We also want W to be greater than K to be able to detect actions in long videos at test time. However, if increasing K and W would lead to an increase in model performance, it would certainly increase the computational cost of the model. We therefore conducted experiments for varied values of K and W and compared the recall scores and the number of frame processed per second (FPS). We used a hidden dimension of 512 for the GRU, a batch size of 128 for training, a learning rate of 0.1 reduced by half every ten epochs, a dropout coefficient of 0.01 and $L2$ regularization for the weights set to 0.005. The results are presented in table 1. We can observe that Recall@1000 increases with W and K, however FPS also increases with W and K. This suggests that to take fully advantage of the SST model, one would have to find a threshold between performance and time complexity. Note that [5] reported an average FPS of 134 frames per second evaluated on the THUMOS dataset [12]. This result suggests that SST would be faster. However, for fair comparison, we would have to measure running times of DAPs on our own machine for the ActivityNet dataset.

4.5. Comparison to the DAPs baseline

In this section, we compare the SST model to the DAPs model for the proposals generation task on the ActivityNet dataset. We reported the scores of Khrishna et al. [14] for the temporal generation task in figure 4. We computed Recall@1000 for varied ious for different values for W and K. We also computed Recall@K for several values of K

(W,K)	Recall@1000 IoU=0.8	FPS
(128,64)	20.37 %	**441**
(256,128)	32.43 %	327
(512,256)	**43.02**%	206

Table 1. Recall@1000 for IoU=0.8 for the proposals generation task for varied values of K and W and average FPS

and IoU 0.8. We can observe that the performance of the best SST model are slightly better or comparable to those of the DAPs model. Furthermore, SST operates in one pass and is therefore preferable to DAPs in terms of computational efficiency. Additionally, as pointed out by Buch et al. [1], SST for $W = 512$ and $K = 228$ achieves better recall than DAPs for high IoU regime. This result is important since high IoU regimes allow to retrieve precise proposals, that is proposals with a high overlap with the ground truth proposals.

5. Conclusions

As a conclusion, this project was focused on the implementation of SST and a performance analysis for different values of W and K. We showed that there must be a threshold between performance and time complexity and we showed that SST achieves state-of-the-art performance on the ActivityNet dataset. Finally, the next step in this project is to slightly modify the SST architecture to perform more complex tasks. For instance we can add another layer that produces scores for action classes or a layer that generates captions for proposals.

References

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

[2] 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.

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

[4] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[5] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem. Daps: Deep action proposals for action understanding. In *European Conference on Computer Vision*, pages 768–784. Springer, 2016.

[6] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[7] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[9] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[10] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[11] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. G. Snoek. Action localization with tubelets from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 740–747, 2014.

[12] Y. Jiang, J. Liu, A. R. Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. Thumos challenge: Action recognition with a large number of classes, 2014.

[13] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[14] R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. C. Niebles. Dense-captioning events in videos. *arXiv preprint arXiv:1705.00754*, 2017.

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[16] 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.

[17] K. Tang, B. Yao, L. Fei-Fei, and D. Koller. Combining the right features for complex event recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2696–2703, 2013.

[18] 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.