ProcNets: Learning to Segment Procedures in Untrimmed and Unconstrained Videos

Article · March 2017

CITATIONS
6

READS
412

3 authors:

Luowei Zhou
University of Michigan
18 PUBLICATIONS **439** CITATIONS
SEE PROFILE

Chenliang Xu
University of Michigan
43 PUBLICATIONS **1,211** CITATIONS
SEE PROFILE

Jason Corso
University of Michigan
233 PUBLICATIONS **6,864** CITATIONS
SEE PROFILE

ProcNets: Learning to Segment Procedures in Untrimmed and Unconstrained Videos

Luowei Zhou
University of Michigan
luozhou@umich.edu

Chenliang Xu
University of Rochester
Chenliang.Xu@rochester.edu

Jason J. Corso
University of Michigan
jjcorso@umich.edu

Abstract

We propose a temporal segmentation and procedure learning model for long untrimmed and unconstrained videos, e.g., videos from YouTube. The proposed model segments a video into segments that constitute a procedure and learns the underlying temporal dependency among the procedure segments. The output procedure segments can be applied for other tasks, such as video description generation or activity recognition. Two aspects distinguish our work from the existing literature. First, we introduce the problem of learning long-range temporal structure for procedure segments within a video, in contrast to the majority of efforts that focus on understanding short-range temporal structure. Second, the proposed model segments an unseen video with only visual evidence and can automatically determine the number of segments to predict. For evaluation, there is no large-scale dataset with annotated procedure steps available. Hence, we collect a new cooking video dataset, named YouCookII, with the procedure steps localized and described. Our ProcNets model achieves state-of-the-art performance in procedure segmentation.

1. Introduction

Action understanding remains an intensely studied problem-space, e.g., action recognition [36, 9, 12, 41], action detection [38, 42, 34] and action labeling [23, 24, 16, 3]. Whether or not action is *the* key to video understanding [6], action clearly plays a role in long, untrimmed videos that convey various procedures, such as preparing coffee [22] and changing a tire [2]. In this paper, we study such long untrimmed videos and propose a new model, *Procedure Segment Networks* or *ProcNets* (Figure 1), for automatically learning to extract the constituent category-independent *procedure segments*.

Procedure learning has been proposed before, as video-subtitle alignment [26, 4] or discovery of common procedure steps of a specific process [2, 33], but, these papers

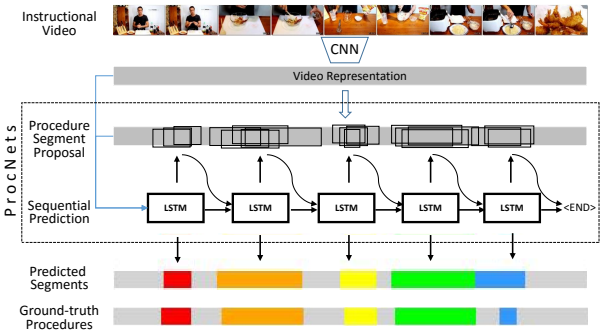

Figure 1. ProcNets learn to segment category-independent temporal procedures in long, untrimmed and unconstrained videos with no assumptions on contiguous segments, number of segments, or specific action classes of the temporal procedure segments.

make restrictive assumptions than we do: they typically assume either language is concurrently available, e.g., from subtitles, the number of procedure steps for a certain procedure is fixed, or both. Such assumptions are limited: extra textual input is unavailable in some scenarios, such as egocentric videos; the subtitles or action sequences automatically generated by machines, e.g., YouTube's ASR system, are inaccurate and require manually intervention; and many procedures of a certain type, such a specific recipe, will vary the number of procedure steps in different instances (process variation). Our proposed ProcNets make neither of these assumptions and hence are more versatile.

In addition to explicit procedure learning, a second area of existing work is in action segmentation or labeling [23, 24, 16, 3]—the problem of segmenting a long video into contiguous segments that correspond to a sequence of actions. The action set typically includes predefined actions such as *Take Bowl* and *Pour Milk* [16, 24]. Most of these methods assume contiguous action segments with limited or no background activities between the segments [31, 22]. Yet, background activities are detrimental to the existing action segmentation approaches [16], and various unconstrained, untrimmed videos spread the procedure segments throughout the video with many gaps. When segment-

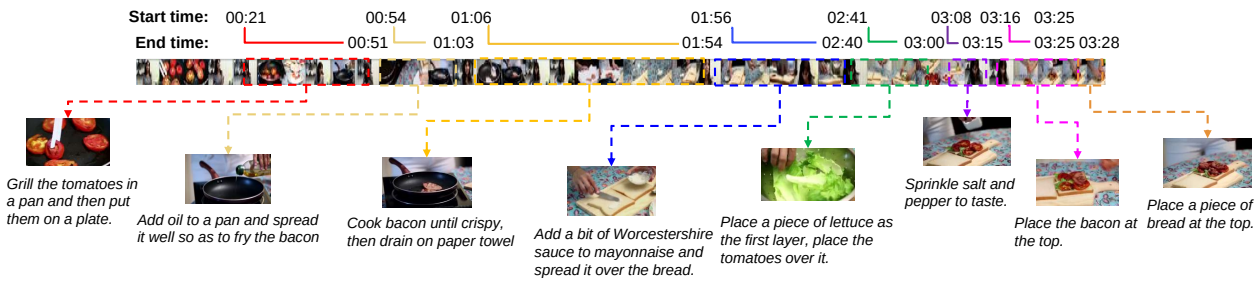

Figure 2. An example from YouCookII dataset on making a BLT sandwich. Each procedure step has time boundaries annotated and is described by a English sentence. Video from YouTube with ID: 4eWzsx1vAi8.

ing the action boundaries, Huang et al. [16] enforce action alignment through frame-wise visual similarities and Kuehne et al. [24] apply Hidden Markov Models (HMM) to learn the likelihood of image features given hidden action states. In both cases, only local temporal structures are learned, i.e., the transitions between adjacent action states, leaving long-range dependencies not captured. For example, in a video of making a BLT sandwich, the procedure step "Grill the tomatoes and put them in a plate" implies a long-range dependency at the end state, which should involve tomatoes either on the plate or in the sandwich.

To address these collective limitations, we propose Procedure Segmentation Networks (ProcNets) for learning the temporal structure of long untrimmed and unconstrained videos while segmenting a video into a sequence of procedure segments. The output procedure segments of ProcNets can be applied for other tasks, such as video description generation [43] or activity recognition[23]. ProcNets have three pieces: 1) context-aware feature encoding from frames, 2) procedure localization for proposing localized candidate temporal segments, which we call *procedure Segment proposals*, 3) sequential procedure modeling for learning the temporal structure among the proposals through an Recurrent Neural Network (RNN) structure. ProcNets address the issues we previously mentioned. First, the proposed model discovers procedure segments in a video using visual evidence only—no language data is assumed available. And, the number of procedure segments is automatically determined by the model. Second, instead of predicting inter-action boundaries, the proposed model detects candidate procedure segments from the background activity; as far as we know, this is the first work in video to use an anchor proposal and offset regression mechanism [30]. Third, we use an RNN structure to learn the long-term temporal structure of the video based on the video content and candidate proposals.

Although our method is general for various situations [35, 1], we focus on instructional videos to allow for an analysis of the extracted procedure segments for a given video, which we expect to be more structured and coupled than in general videos. However, no large-scale instructional video dataset with temporal procedure steps annotated is yet available. To this end, we collect a new cooking video dataset, named YouCookII. YouCookII contains over 2000 videos from 90 recipes with a total length of 140 hours. The procedure steps for each video are annotated with temporal boundaries and described by imperative English sentences (see Fig. 2). To evaluate the sequential ordering of procedure segments, we apply two newly-defined metrics, Average Recall over segments and mean IoU.

The contributions are three-fold. First, as far as we know, we introduce and are the first to tackle the procedure sequential prediction in long, untrimmed and unconstrained videos with no action-specific labels available during training. Second, we propose a generic structure for procedure segmentation and the output segments can be applied to other tasks that involve video segments. Third, we collect a large-scale dataset for understanding procedure segments in instructional videos.

2. Related Work

Our idea of studying the procedure segmentation problem is inspired by multiple research focuses and here we introduce some critical work on forming our viewpoints towards our problem.

Video Segmentation and Procedure. Temporal video segmentation [28, 20] focuses on segmenting a video into some visually or semantically integral segments. Kowdle et al. [21] define the temporal video segmentation problem as detecting shot or scene boundaries. Potapov et al. [28] further develop the definition as segmenting a video into semantically-consistent segments by detecting shot boundaries and also general semantic change points. Here, we summarize the definition of *procedure segment* as the semantic units for fulfilling some subprocess which is rather consistent with existing literature [33, 2].

Temporal Action Detection. The approaches in action detection, especially the recent ones based on action proposal [10], inspire us to segment a video procedure by proprosing procedure segment candidates. Early works on action detection mainly use sliding windows for proposing segments [13, 27]. Disadvantages of this type of method have been

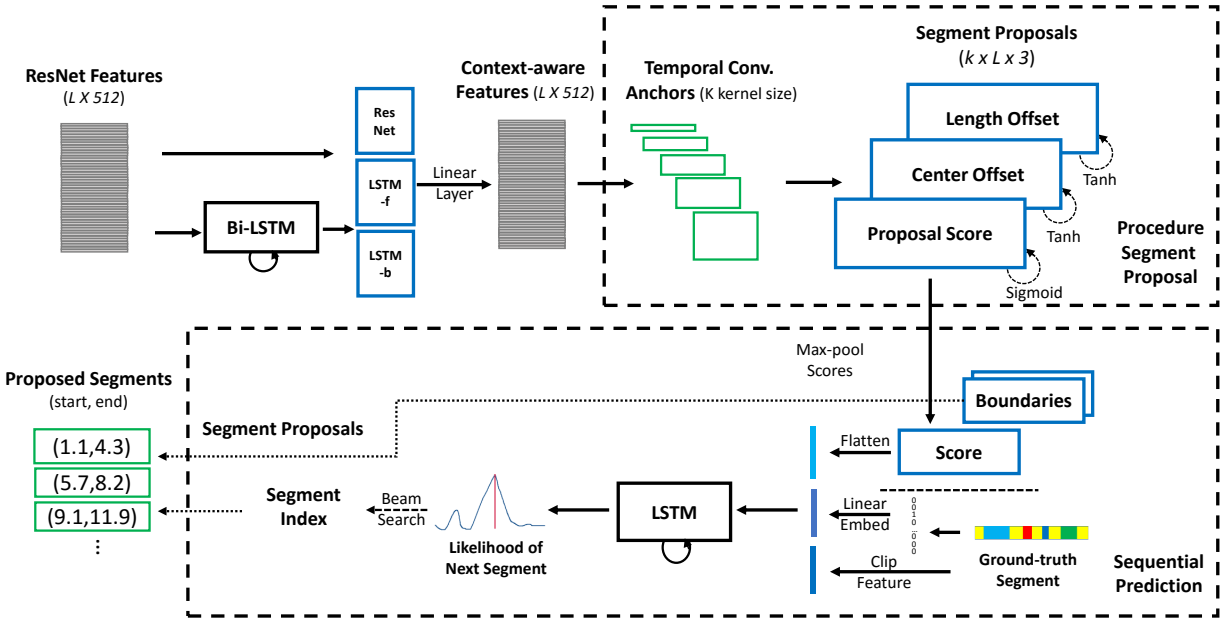

Figure 3. Schematic illustration of the Procedure Segmentation Network (ProcNets). The input is the frame-wise ResNet feature (by row) for a video. The outputs is the proposed procedure segments. The input feature travels though the end-to-end network with three stages. First, the bidirectional LSTM embeds the ResNet features into context-aware features. Then, the procedure segment proposal module proposes procedure proposals. Finally, the sequential prediction module selects the final proposals with LSTM. During training, the ground-truth segments are embedded to composite the LSTM input, which are removed in testing. For inference, beam search is applied, as shown with the dashed lines.

summarized in [10]. More recently, Shou et al. [34] propose a multi-stage convolutional network that penalties action proposals with small overlapping with the ground truth and achieves state-of-the-art performance on Thumos 2014 [17]. The most similar work to ours is Deep Action Proposals (DAPs) proposed by Escorcia et al [10] where the model predicts the likelihood of action proposals to be an action while our model predicts procedure segment proposals. Another difference is, the DAPs model determines the location of action anchors by k-means clustering over the ground-truth segments for training, while our model learns to localize procedures by regression which addresses the case where actions distributed differently in the test video.

Action Labeling and Procedure Learning. Modeling the sequence of actions or procedure segments by leveraging local visual patterns has been studied recently. Given a video and the sequence of actions exist in the video, action labeling approaches temporally localize each action in the video in a weakly supervised manner [3]. Huang et al. [16] propose a model that forces action to have visual consistence based on the frame-wise visual similarities. Kuehne et al [24] apply HMMs to align actions with video frames. In both cases, only local temporal structures are built, i.e., the transitions between adjacent action states, but long-range temporal structure is barely captured. In contrast, our method can learn long-range temporal structure.

3. Procedure Segmentation Networks

We propose a model, named Procedure Segmentation Networks (ProcNets), for segmenting a long untrimmed and unconstrained video into a sequence of procedure segments. At testing, for any given unseen video, ProcNets propose and localize the candidate procedure segments in the video based on their visual appearances and temporal relations, without the need of knowing pre-specified procedure steps [24] or the subtitles of the video [2].

The proposed ProcNets has three stages: context-aware video encoding (Sec. 3.1), procedure segment proposal (Sec. 3.2), and sequential prediction (Sec. 3.3). The overall network structure is shown in Fig. 3 and the overall loss function is defined in Sec. 3.4.

3.1. Context-Aware Video Encoding

Without loss of generality, we represent a long untrimmed video as $\mathbf{x} = \{x_1, x_2, \ldots, x_L\}$, where L denotes the number of sampled frames of the video and $x_i \in \mathbb{R}^e$ is the frame-level CNN feature. Here, e is the feature encoding size. We use ResNet [15, 11], pretrained on both ImageNet [8] and MSCOCO [25], as the video-frame encoder for its state-of-the-art performance in image classification.

This video representation captures the appearance information of separate frames. We then put these ResNet features through one-layer bidirectional LSTM

(Bi-LSTM) [14] with 512 hidden units, to encode local temporal dependencies. The outputs (forward and backward) are concatenated with the ResNet feature for each frame, which is followed by a linear layer to reduce the feature dimension to be the same as ResNet feature. The dimension reduction allows a fair comparison against other encoding options. We call these *context-aware features*, denoted as $b_i = \text{Bi-LSTM}(\mathbf{x})$, where $b_i \in \mathbb{R}^e$. Now, we have a context-aware video encoding: $\mathbf{b} = \{b_1, b_2, \ldots, b_L\}$, and use it as the input for generating procedure segment proposals. Empirically, Bi-LSTM encoder outperforms context-free feature and LSTM-encoded feature by relative 9% on our metric AP@0.5, which will be introduced in Sec. 4.2. Adding motion features such as optical flow may further improve the performance.

3.2. Procedure Segment Proposal

Inspired by the idea of convolutional anchors for spatial object proposal, such as in Faster R-CNN [30], we design a set of K temporal convolutional anchors for procedure segment proposal in videos. Each anchor has the shape: $l_k \times e$ $(k = 1, 2, .., K)$, such that their width is the same as frame feature size and their varying lengths cover different time spans (see Fig. 3). This design of temporal convolutional anchors together with our context-aware video encoding in Sec. 3.1 makes it possible to compute procedure segment proposals.

We apply anchors to video encoding and obtain proposal scores and offsets (center and length), where the scores are regarded as the likelihood for temporal anchors at certain location to contain a procedure segment and the offsets are used to adjust the proposed segment boundaries. By zero-padding the video encoding at the beginning and the end (depends on anchor sizes), we obtain score and offsets matrices of size $K \times L$ (see upper right of Fig. 3) and the total size is $K \times L \times 3$.

We formulate the proposal scoring as a classification problem and proposal offsetting as regression problems. Sigmoid function and 0.1 scaled Tanh function are applied for scoring and offsetting, respectively. The segment proposals are classified as procedure segment or non-procedure segment with cross-entropy loss. During training, the segment proposals having at least 0.8 Intersection-over-Union (IoU) with any ground-truth segments are regarded as positive samples and these having IoU less than 0.2 with all the ground-truth are treated as negative samples. We randomly pick U samples from positive and negative respectively for training. We regress the offsets of proposal length and center in the input feature block instead of their absolute values. For a proposal with center t_a and length l_a, its regressed center t_p and length l_p w.r.t. offsets (θ_t, θ_l) are defined as:

$$t_p = t_a + \theta_t l_a \ , \quad l_p = l_a e^{\theta_l} \ . \tag{1}$$

Therefore, the boundaries for the segment proposal are defined as: $s_p = t_p - \frac{l_p}{2}$ and $e_p = t_p + \frac{l_p}{2}$. To train the regression model, we regress the positive samples to it corresponding ground-truth procedure segment. Smooth l_1-loss [30] is applied for the loss.

3.3. Sequential Prediction

Contrary to spatial objects, video procedure segments, by their natural, have ambiguous temporal boundaries and yet strong temporal dependencies. Therefore, we must treat them differently. We use LSTM to learn a sequential prediction of procedure segments and dynamically determine the number of procedure segments in a video.

Our LSTM leverages three representations: 1) Proposal Vector $\mathbf{S}$: a set of max-pooled proposal scores from Sec. 3.2; 2) Location Embedding B_i: a vector of embedded information by the location of a procedure segment; 3) Segment Content C_i: the segment feature vector. A tuple of the three representation $(\mathbf{S}, B_i, C_i)$ $i = 1, 2, ..., n$ serves as input to the LSTM. The intuition is that the LSTM predicts current procedure segments w.r.t. both previously predicted segments and the set of proposals as a whole.

The softmax output of our LSTM is the likelihood of each proposal being the next segment prediction. Therefore, the likelihood for the entire procedure segment sequence $\epsilon_1, ..., \epsilon_S$ of a video can be formulated as:

$$\begin{aligned} &\log p(\epsilon_1, ..., \epsilon_S | \mathbf{S}, B_1, ..., B_n, C_1, ..., C_n) \\ &= \sum_{t=0}^{S} \log p(\epsilon_t | \mathbf{S}, B_1, ..., B_n, C_1, ..., C_n, \epsilon_0, ..., \epsilon_{t-1}) \ , \end{aligned} \tag{2}$$

where ϵ_0 is the special <start> procedure segment. The objective of the sequential prediction is to maximize the sum of the sequence likelihood for all training videos. We apply cross-entropy loss to the likelihood output P_t at time step t, from which we sample with beam search [40, 9] for inference. In our experiments, simply let beam search equal to 1 yields the best result, i.e., picking the index with the maximal likelihood as the next proposed segment. The algorithm terminates when the special stop bit is picked. Next, we describe the individual LSTM input representations in detail.

Proposal Vector. As shown in the bottom part of Fig. 3, we pass the proposal score through a max-pooling layer to filter out proposals that are less likely contain a procedure segment. We define the kernel size as $K_h \times K_w$ and set the stride size so that there is no overlapping. Experiments on the size of max-pooling layer are in Sec. 4.3. Empirically, setting $K_h = 8$ and K_w at 4 or 5 yields the best results. The boundaries of segment proposals are filtered according to the selected scores in the max-pooling. Then the proposal score matrix is flattened into a vector $\mathbf{S}$ by columns and we name it Proposal Vector.

Location Embedding. During training, the ground-truth procedure segment is represented by a one-hot vector where the one entry matches to the nearest proposal in the Proposal

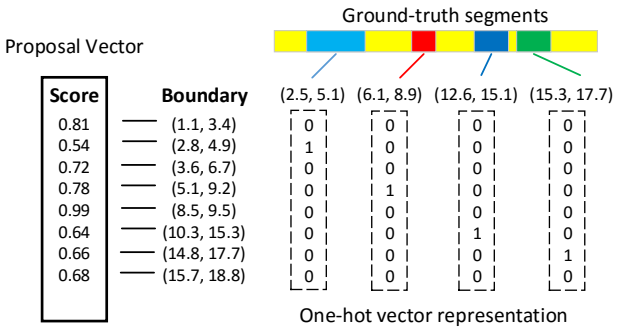

Figure 4. An example on converting ground-truth segments into one-hot vector representations based on the Proposal Vector.

Vector as illustrated in Fig. 4. This discrete representation of locations is easier for structure learning than continuous location values. Through a trainable embedding matrix, this one-hot vector maps to a vector, which we call Location Embedding vector and depicts the location information of a segment. This Location Embedding vector has the same size as the Proposal Vector. During testing, we sample the previous softmax output of LSTM as the one-hot vector representation for the next segment with beam search.

Segment Content Feature. To encode the procedure segment content, we mean-pool the video ResNet feature according to the segment boundaries. The dimension of segment content feature is reduced to match the Proposal Vector by a fully-connected layer and we refer this feature as segment feature. Intuitively, the procedure segment content can infer the temporal structure. Recall the example of making a BLT sandwich, the procedure segment *Grill the tomatoes and put them in a plate* implies long-range dependency at the end state, which should involve tomatoes either on the plate or in the sandwich.

Relations to Other Models. To the best of our knowledge, we are the first to apply sequential modeling to extract general video procedure segments. The proposed sequential prediction model builds the video temporal structure without the need of knowing the hidden states such as in HMM. Note that there are other design choices.

Non-Maximal Suppression (NMS). In terms of proposal selection, a commonly adopted method in object detection [30] or action detection [34] is NMS. This approach fails to capture the temporal structure of instructional videos and procedure sequence dependencies, and usually requires the total segment number as an oracle. Also, we compare it in Sec. 4.3.

Other Time Sequence Models. Other methods for proposing segments have rigid model configurations, such as an HMM or pre-defined "grammar" for the whole video, which is infeasible for general video structure inference.

3.4. Loss Function

The loss function for procedure segmentation network consists three parts, the cross-entropy loss for the procedure-ness classification, the smooth $l1$-loss [30] for offset regression and the cross-entropy loss for sequential prediction. The formulations are as follows:

$$L = L_{cla} + \alpha_r L_{reg} + \alpha_s L_{seq} \tag{3}$$

$$L_{cla} = \frac{1}{U_p + U_n}\left(\sum_{i=1}^{U_p} \log(S_i^{(pos)}) + \sum_{i=1}^{U_n} \log(1 - S_i^{(neg)})\right)$$

$$L_{reg} = \frac{1}{U_p}||B_i - B_i^{(gt)}||_{smooth-l1}$$

$$L_{seq} = \frac{1}{S}\sum_{t=1}^{S} \log(P_t^T \mathbb{1}_t^{(gt)})$$

where U_p and U_n are the number of positive and negative samples, respectively, $S_i^{(pos)}$ and $S_i^{(neg)}$ represents their scores, $B_i^{(gt)}$ is the ground-truth boundary corresponding to positive sample i, P_t is the softmax output of LSTM at time t and $\mathbb{1}_t^{(gt)}$ is one-hot vector of ground-truth segment index. Discount factors α_r and α_s are applied to balance the contributions of the regression loss and sequential prediction loss, respectively. The impacts of different discount values on procedure segmentation is discussed in Sec. 4.3.

4. Experiments

In this section, we first introduce our new dataset for procedure learning in instructional videos (see Sec. 4.1). Then, we benchmark the dataset with state-of-the-art methods in video segmentation and action detection as well as our proposed methods. We compare all the approaches with newly-defined metrics focusing on evaluating the sequential ordering of procedure segments. We analyze our model with both functional (e.g., ablation studies) and empirical (e.g., parameters tuning) experiments in Sec. 4.3.

4.1. YouCookII Dataset

The dataset. Existing datasets for analyzing instructional videos suffer from either limited video resources [2, 33] or weak diversity (identical background scene [29] or similar activities [22]). Nonetheless, they provide limited or no ground-truth procedure segment boundaries or descriptions [35]. Some statistics and properties of commonly-used instructional video datasets, namely, YouCook [7], MPII [32], 50Salads [39], Breakfast [22] and Coffee [2] are summarized in Tab. 1. To this end, we collected a cooking video dataset, named YouCookII[1]. Compared to the 88-video YouCook dataset [7], our dataset contains larger amount of videos and more diverse recipes. YouCookII

[1]More details on YouCookII are in the Appendix. The dataset will be public available. The source code of ProcNets is available at Github.

Table 1. Comparisons of instructional video datasets. UnCons. stands for Unconstrained Scene, Act. Ann. means action annotation and Proc. Ann. is short for Procedure Annotation.

Name	Duration	UnCons.	Act. Ann.	Proc. Ann.
YouCook	140 m	Yes	No	No
MPII	490 m	No	Yes	No
50Salads	320 m	No	Yes	No
Coffee	120 m	Yes	Yes	No
Breakfast	67 h	Yes	Yes	No

contains 2007 YouTube videos of 90 recipes (around 23 videos each) from different regions around the world and the recipe steps for each video are extracted and described in English sentences by well-trained annotators. Each recipe contains 3 to 15 steps, where each step is described by a sentence in imperative form, such as *grill the tomatoes in a pan*. An example is on Fig. 2 in Sec. 1. The total video time is 160 hours with an average length of 4.84 mins for each video. The length of procedure segments on the whole dataset is distributed as in Fig. 14 with a mean of 20.4 seconds and a standard deviation of 19.0 seconds. Noting that in this paper, we only use the procedure segment annotations. We make the recipe descriptions available for future research on YouCookII.

Preprocessing. Since our proposed model needs to access the overall information in long untrimmed videos, we uniformly down-sample 500 frames for each video in YouCookII. The average sample rate is 1.73 fps. To further enlarge the training samples, we temporally augment the data, i.e., sample each video 10 times with temporal shifts. Then, we extract the ResNet-34 [15] feature for each frame. A video is represented as a sequence of image spatial features, which already captures the long-ranging temporal changes in instructional videos. Notice that, local motion features are not used in our study, which may further improve performance, and we leave it as our future work.

4.2. Experimental Setup

Baselines. Here we introduce some baseline methods for procedure segmentation. Since we are the first to tackle the procedure sequential prediction problem for instructional videos, no direct baselines exist. We adapt state-of-the-art methods in relevant fields, such as temporal segmentation, video summarization and activity detection, to our problem. The selected methods include: 1) Kernel Temporal Segmentation (KTS) [28], 2) Video Summarization LSTM (vsLSTM) [44], 3) Segment CNN (S-CNN) [34]. For KTS, we compute temporal segmentation for a procedure video on image features. Notice that this method is fully unsupervised and has no exposure of the annotated segments. The vsLSTM is originally used for summarizing a video into keyframes or key clips. In our scenario, we apply the model for proposing procedure segments, which is trained

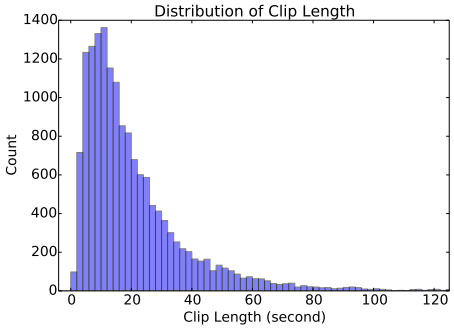

Figure 5. Clip length distribution in the YouCookII dataset.

in an identical way as producing key clips. The original S-CNN has three-stages, namely, action proposing, action classification and action localization. We remove the classification networks given that no procedure labels exist in our setting. Also, we disable the feature fine-tuning of C3D for a fair comparison with our models. Besides, a uniform segment baseline (Uniform) with clip number optimized is also evaluated. Finally, note that we compare with no action labeling or segmentation methods since these approaches directly model the finite action states (e.g., with HMM) which requires the "grammer" of the whole video process.

Metrics. Here we focus on how to evaluate the sequential ordering of procedure segments. Existing metrics on action segmentation or action detection fail to measure ordering information among individual segments. Hence, we propose two new metrics for evaluating procedure segmentation: Average Recall at 0.5 (AR@0.5) and mean IoU (mIoU). For AR@0.5, we compare the proposed segments in chronological order with the ground-truth segments: if the overlapping is greater than 0.5, the proposal is correct, otherwise incorrect. The mIOU is the mean IoU of proposed segments with corresponding ground-truth segments and averaged over all the validation videos. The reason why we use IoU instead of Jaccard meansure [3] is that we penalize all the misalignment of segments in order to evalute the segment regression performance. It worth noting that in our problem, the AR metric is really strict in a sense that the model needs to accurately match all the segments in order. For example, in Fig. 6, almost all the ground-truth segments have corresponding proposed segments, but, owing to the second false positive segment, the recall becomes zero. Despite the accurate localization performance the method has, results of AR metric cannot fully reflect the proposal accuracy. To this end, we also report the results for relaxed procedure matching where the proposed procedure segment can be r steps away from the ground-truth, similar to the relaxation boundary in Bosselut et al. [5]. In our case, we have $r = 0$ or $r = 1$.

Training Details. We randomly split the training, validation and testing data with ratio 67%:23%:10% for videos belonging to each recipe. The sizes of the temporal con-

Table 2. Results on temporal segmentation. The uniform model is optimized with generating 8 segments. KTS and ProcNets-NMS output 7 segments per video. We report the AR@0.5 and mIoU values and highlight top-2 scores.

	$r = 1$				$r = 0$			
	validation		test		validation		test	
Method (%)	AR@0.5	mIoU	AR@0.5	mIoU	AR@0.5	mIoU	AR@0.5	mIoU
Uniform	16.1	**28.0**	16.7	**28.0**	6.2	10.6	7.1	10.8
KTS [28]	16.1	25.3	13.8	24.3	7.6	11.0	6.1	10.0
vsLSTM [44]	16.7	26.9	12.2	23.9	8.1	11.4	5.5	9.6
S-CNN [34]	11.2	21.2	11.7	22.1	4.8	8.9	5.6	9.2
ProcNets-NMS (ours)	**18.3**	27.3	**19.1**	27.6	**8.3**	**11.6**	**9.4**	**11.8**
ProcNets-LSTM (ours)	**24.2**	**35.3**	**22.8**	**33.1**	**11.8**	**15.0**	**10.9**	**14.0**

Figure 6. A negative example (ID: Y9fvfbZGad0) on procedure segmentation. The color order is consistent with the chronological order. Recall=0 owing to the second false positive segment.

volutional kernel are from 3 to 123 with an interval of 8, which are approximated to make 95% of the segments in the training set drop into this range. Hence, 16 anchors are proposed at each segment center. For procedure classification, we pick $U = 100$ samples from all the positive and negative respectively and feed negative samples if positive ones are less than U .Our implementation is based on Torch 7 and refers to Neuraltalk2 [18]. For hyper-parameters, we set the weight decay rate for the procedure segmentation network to 10^{-3} to avoid overfitting. The learning rate for language model is set to 4×10^{-5}. We use Adam optimizer [19] for updating weights with $\alpha = 0.8$ and $\beta = 0.999$. The training process for the temporal segmentation model takes about six days on a single NVIDIA TITAN X GPU. Note we are able to train ProcNets end-to-end from raw video to the segment output. However, in practice, we disable the CNN fine-tuning as it heavily slows down the training process.

4.3. Procedure Segmentation

We report the procedure segmentation results on both validation and testing sets in Tab. 2. The first three methods share the same property of proposing contiguous segments. S-CNN detects segments among the background activity. The proposed ProcNets-LSTM model outperforms all other methods by a huge margin in all the metrics. The performances of S-CNN suffers in the segmentation task possibly result from the lack of sequential modeling mechanism. Note that optimized Uniform method is a relative strong baseline since in instruction videos, generally procedures span in the whole video. In our experiments, it performs better than some supervised methods, such as vsLSTM, S-CNN and ProcNets-NMS in some scenarios. The Uniform method achieves even higher scores during testing, implying the test videos have procedures more evenly distributed. For the rest experiments, except for model selection is on validation set, all the results are on testing data.

Table 3. Ablation study. We remove one of Proposal Vector (denoted as *-Proposal Vec*), Location Embedding (denoted as *-Location Emb*) or Segment Content feature (denoted as *-Segment Feat*) from the LSTM input.

	AR@0.5	mIoU
Full model	10.9	14.0
-Proposal Vec	8.0	11.4
-Linear Emb	9.3	13.1
-Segment Feat	9.7	13.9

Ablation study on sequential prediction. The input of the sequence modeling LSTM consists of three parts: Proposal Vector, Location Embedding and Segment Content vector. We remove each of them as the ablation study. Results are shown in Tab. 3. Unsurprisingly, the proposal scores (Proposal Vector) play the most significant role in inferring the temporal structure. When this information is unavailable, the overall performance drops by 26% relatively. The Location Embedding implicitly encodes the location information for the segments and it helps the model memorize how segments are distributed in a video. The video content represented by segment feature has less contribution to the sequence prediction, which implies the video information contained in the feature representation is noisy.

Model selection. We study the impact of different parameter combinations on the model performance. Two groups of experiments are conducted regarding to: 1) max pooling kernel size, 2) loss discount factors. All the model selection results are on the validation set to reach the final model parameters for test uses. For max-pooling kernel size, we adopt seven combinations (see Tab. 4). It turns out that the cases, when $K_h = 8$ and K_w is at 4 or 5, yield decent results. This implies that a moderate number of proposal scores, say 200, benefits the segmentation. For different combinations of discount factors in the combined loss function, we show the results in Tab. 5. In the case when all three loss terms have equal weights, it leads other combinations by a huge relative gap (20%-27% for AR@0.5).

Proposal localization accuracy. We report the proposal accuracy on segment detection in terms of recall, precision and F1 score (see Tab. 6). S-CNN shows better seg-

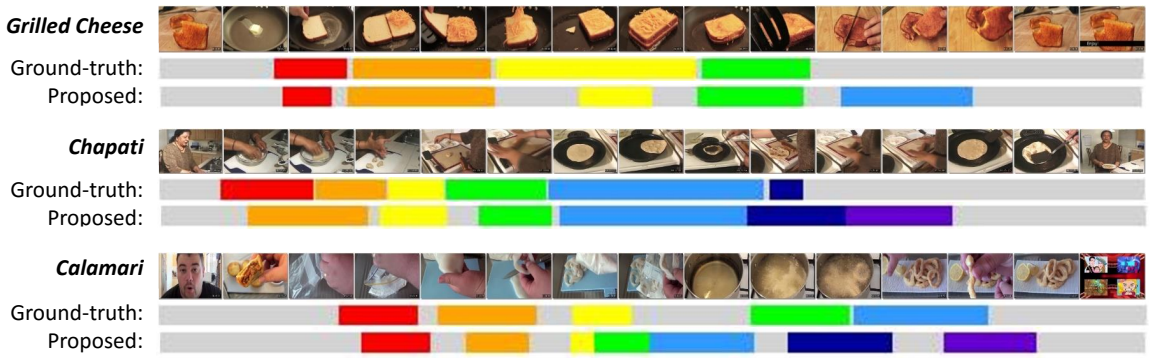

Figure 7. Examples on procedure segmentation. YouTube IDs: BlTCkNkfmRY, jD4o_Lmy6bU and jrwHN188H2I.

Table 4. Experiments on the max pooling kernel size. K_h is kernel height and K_w is kernel width. $\alpha_r = 0.2, \alpha_s = 0.2$

	AR@0.5	mIoU
$K_h = 16, K_w = 2$	7.2	10.5
$K_h = 4, K_w = 4$	7.8	10.4
$K_h = 5, K_w = 5$	8.9	11.3
$K_h = 8, K_w = 2$	8.9	12.3
$K_h = 8, K_w = 4$	10.3	13.9
$K_h = 8, K_w = 5$	10.0	13.3
$K_h = 8, K_w = 10$	9.9	13.2

Table 5. Experiments on discount factors. α_r is for clip regression loss and α_s is for sequence prediction loss. The max pooling kernel size is 8×5.

	AR@0.5	mIoU
$\alpha_r = 0, \alpha_s = 0.2$	9.3	13.1
$\alpha_r = 0.2, \alpha_s = 0.2$	10.0	13.3
$\alpha_r = 1, \alpha_s = 0.2$	9.8	12.7
$\alpha_r = 1, \alpha_s = 1$	11.8	15.0

Table 6. Results on segment detection accuracy. ProcNets output 10 segments with the highest procedureness scores under NMS. Recall, precision and F1 score are computed w.r.t. the ground-truth segments. Top two scores are highlighted.

Method (%)	Recall	Precision	F1
Uniform	27.4	23.3	25.2
KTS [28]	20.9	23.3	22.0
vsLSTM [44]	22.1	**24.1**	23.0
S-CNN [34]	**28.2**	23.2	**25.4**
ProcNets (ours)	**37.1**	**30.4**	**33.4**

ment detection capability than optimized Uniform, KTS and vsLSTM models in most of the metrics. Our proposed model has at least 9% higher recall and 7% higher precision than the baslines.

Analysis on temporal structure learning. We conduct some experiments to demonstrate ProcNets' temporal structure learning capability. For a given testing video, denote the first half as V_a and the second half as V_b. We inverse the order of V_aV_b to V_bV_a to construct the permutated video. We evaluate our model on both original test set and the permutated test set. The performance of pre-trained

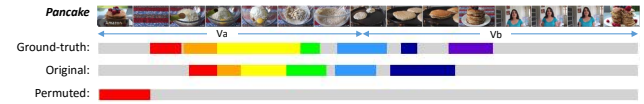

Figure 8. An example output of ProcNets on the original and the permuted video. YouTube ID: ejq2ZsHgwFk.

ProcNets decreases by over a half in the permutated set and 10%-20% of the videos only have segments predicted at the begining of V_b (see Fig. 8). To explain this, We have two hypotheses. First, the model captures the ending content in V_b and terminates the segment generation within V_b. Second, the temporal structure of V_a has no dependencies on V_b and hence is ignored by the model.

Qualitative results. We demonstrate qualitative results from YouCookII test set (see Fig. 7). For some procedure segments, the model can accurately localize segments and predict their lengths. Moreover, the number of segments proposed varies and the model learns to propose no segments at the begining and the end of the video. In the example of making Grilled Cheese, ProcNets propose the fifth segment to cover the process of cutting bread slices into two pieces. This trivial segment is not annotated but is still semantically meaningful.

5. Conclusion

We present an approach for video procedure learning and segmentation, named ProcNets. ProcNets take frame-wise video feature as the input and predicts procedure segments exist in the video. The output procedure segments can be applied for other tasks, such as video description generation or activity recognition. We evaluate the model against baselines on the newly collected large-scale cooking video dataset and the proposed method outperforms the baselines by a huge margin. We also demonstrate in the experiments that ProcNets are capable of inferring the video structure by video content and modeling the temporal dependencies among procedure segments.

Acknowledgement. This work has been supported in part by Google, ARO W911NF-15-1-0354 and NSF NRI 1522904. This article solely reflects the opinions and conclusions of its authors and not Google, ARO nor NSF. We sincerely thank my colleagues Vikas Dhiman, Madan Ravi Ganesh and Ryan Szetor for their helpful discussions. We also thank Mingyang Zhou, Yichen Yao, Haonan Chen and Dali Zhang for their efforts in constructing the dataset.

References

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 2

[2] J.-B. Alayrac, P. Bojanowski, N. Agrawal, I. Laptev, J. Sivic, and S. Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 2, 3, 5, 11

[3] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision*, pages 628–643. Springer, 2014. 1, 3, 6

[4] P. Bojanowski, R. Lajugie, E. Grave, F. Bach, I. Laptev, J. Ponce, and C. Schmid. Weakly-supervised alignment of video with text. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4462–4470, 2015. 1

[5] A. Bosselut, J. Chen, D. Warren, H. Hajishirzi, and Y. Choi. Learning prototypical event structure from photo albums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016. 6

[6] W. Chen, C. Xiong, R. Xu, and J. J. Corso. Actionness ranking with lattice conditional ordinal random fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 1

[7] P. Das, C. Xu, R. F. Doell, and J. J. Corso. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2634–2641, 2013. 5, 11

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009. 3

[9] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015. 1, 4

[10] 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. 2, 3

[11] Facebook. Torch implementation of resnet. https://github.com/facebook/fb.resnet.torch, 2016. 3, 14

[12] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1933–1941, 2016. 1

[13] A. Gaidon, Z. Harchaoui, and C. Schmid. Temporal localization of actions with actoms. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2782–2795, 2013. 2

[14] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005. 3

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 3, 6

[16] D.-A. Huang, L. Fei-Fei, and J. C. Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision*, pages 137–153. Springer, 2016. 1, 2, 3

[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. 3

[18] A. Karpathy. Neuraltalk2. https://github.com/karpathy/neuraltalk2, 2015. 7

[19] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[20] I. Koprinska and S. Carrato. Temporal video segmentation: A survey. *Signal processing: Image communication*, 16(5):477–500, 2001. 2

[21] A. Kowdle and T. Chen. Learning to segment a video to clips based on scene and camera motion. In *European Conference on Computer Vision*, pages 272–286. Springer, 2012. 2

[22] H. Kuehne, A. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 780–787, 2014. 1, 5

[23] H. Kuehne, J. Gall, and T. Serre. An end-to-end generative framework for video segmentation and recognition. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8. IEEE, 2016. 1, 2

[24] H. Kuehne, A. Richard, and J. Gall. Weakly supervised learning of actions from transcripts. *arXiv preprint arXiv:1610.02237*, 2016. 1, 2, 3

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. 3

[26] J. Malmaud, J. Huang, V. Rathod, N. Johnston, A. Rabinovich, and K. Murphy. What's cookin'? interpreting cooking videos using text, speech and vision. *arXiv preprint arXiv:1503.01558*, 2015. 1

[27] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1817–1824, 2013. 2

[28] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid. Category-specific video summarization. In *European conference on computer vision*, pages 540–555. Springer, 2014. 2, 6, 7, 8

[29] M. Regneri, M. Rohrbach, D. Wetzel, S. Thater, B. Schiele, and M. Pinkal. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics*, 1:25–36, 2013. 5

[30] 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*, pages 91–99, 2015. 2, 4, 5

[31] A. Rohrbach, M. Rohrbach, W. Qiu, A. Friedrich, M. Pinkal, and B. Schiele. Coherent multi-sentence video description with variable level of detail. In *German Conference on Pattern Recognition*, pages 184–195. Springer, 2014. 1

[32] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1194–1201. IEEE, 2012. 5

[33] O. Sener, A. R. Zamir, S. Savarese, and A. Saxena. Unsupervised semantic parsing of video collections. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4480–4488, 2015. 1, 2, 5

[34] 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. 1, 3, 5, 6, 7, 8

[35] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, 2016. 2, 5

[36] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 1

[37] K. Simonyan and A. Zisserman. Vgg16 pre-trained models. http://www.robots.ox.ac.uk/~vgg/research/very_deep/, 2014. 14

[38] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1961–1970, 2016. 1

[39] S. Stein and S. J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 729–738. ACM, 2013. 5

[40] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015. 4, 15

[41] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016. 1

[42] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2678–2687, 2016. 1

[43] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4584–4593, 2016. 2

[44] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. Video summarization with long short-term memory. *arXiv preprint arXiv:1605.08110*, 2016. 6, 7, 8

[45] L. Zhou, C. Xu, P. Koch, and J. J. Corso. Image caption generation with text-conditional semantic attention. *arXiv preprint arXiv:1606.04621*, 2016. 15

6. Appendix

6.1. Results on Other Datasets

We demonstrate procedure segmentations results on some other long, untrimmed and unconstrained videos. The ProcNets model remains the same, i.e., trained on the YouCookII training set. We apply it to segment procedures on unseen videos from two commonly used cooking video datasets, YouCook [7] and Coffee [2].

For YouCook, we randomly pick 8 videos from the 37 test videos of YouCook and compare ProcNets against the top two baselines in our experiments on YouCookII, i.e., KTS and Optimized Uniform. The results are shown in Fig. 10. ProcNets generalize well to unseen videos belonged to unknown recipe categories. The first and the last segments generated by ProcNets are closer to the actual starting time and ending time of the cooking process than these generated by the baselines (see the first three examples). For Video50, ProcNets accurately predicts the orange and the green segments. For video03, the ground-truth segments are rather uniformly distributed which favor the uniform baseline. ProcNets generate comparable beginning segments. For Video24 and Video33 where all the model fail, the total segments in the video are 4 which is lower than usual cases. For videos in Coffee dataset, the procedure segments are sparse and the video content are barely observed in YouCookII training set. Despite this, ProcNets can still generate reasonable segments (see Fig. 9). For instance, the first segment in Video01 captures the first procedure step, and the predicted segments in Video03 have large overlappings with the ground truth. Also, the number of segments generated for video03 is exactly the same as the annotation while for video09 is only one segment away.

6.2. YouCookII Dataset

Some important features that distinguish YouCookII from existing datasets of instructional videos:

- Large-scale cooking dataset with more than 2000 annotated videos.
- Unconstrained videos, can be preformed by individual persons at their houses with unfixed cameras.
- Long untrimmed videos, up to 10 minutes.
- Extremely rich recipe types and various cooking styles from all over the world.
- Procedure steps temporally localized and described by English sentences.

In this section, we describe some details of YouCookII dataset in four aspects: 1) Data acquisition, 2) Annotations, 3) Dataset statistics, and 4) Precomputed Feature.

Data acquisition. We allow workers to collect video data through a web interface. The interface can record various information for a given video, which is listed as follows.

- Video URL.
- Number of cooks.
- Sound options, include English Speech, Native Language Speech, English Speech and Background Music, and Native Language Speech and Background Music.
- Time length estimation.
- Time interval of the cooking process. Start time and end time.
- Cooking style. Choose one from 11 types. Broiling/Baking, Steaming, Grilling, Roasting, Stewing, Frying, Raw/Cold-prep, Slow Cooking, Boiling, Kneading and Deep-frying.
- Recipe type. 110 recipes in total. 90 of them are used to construct YouCookII.

Recipe types are specified as the keywords to retrieve videos from YouTube. For each recipe type, we collect at most 25 videos. Each video is within 10 mins and should be recorded by camera devices but not slideshows. All the videos are in third-person viewpoint and are made under various kitchen environments. The videos collected should have high resolutions and be recently uploaded.

YouCookII consists of 90 recipes from four major cuisine locales, i.e., America, European/Middle East, East Asia and South Asia. The full recipe list is shown in Fig. 11. After removing videos that are no longer available online, we collect a total of 2007 unique YouTube videos with 23 videos per recipe in average. We randomly split the videos belonging to each recipe into 67%:23%:10% as the training, validation and testing sets.

Annotations. We acquire structured recipe descriptions for each video. Recipe steps are temporally annotated with the starting time and ending time. Each step is described by a human annotator with a English sentence. Each recipe contains 3 to 15 steps, where each step is described by a sentence in imperative form, such as *grill the tomatoes in a pan*. The annotators have access to the audio/subtitle but are required to organize and summarize the recipe descriptions in their own ways. Also, the annotators should not be biased

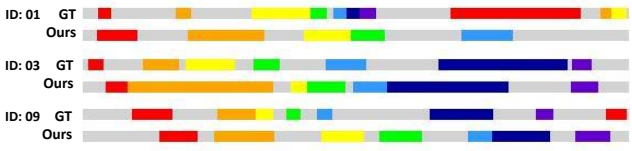

Figure 9. Procedure segmentation results on Coffee dataset with pre-trained ProcNets model. GT stands for ground-truth.

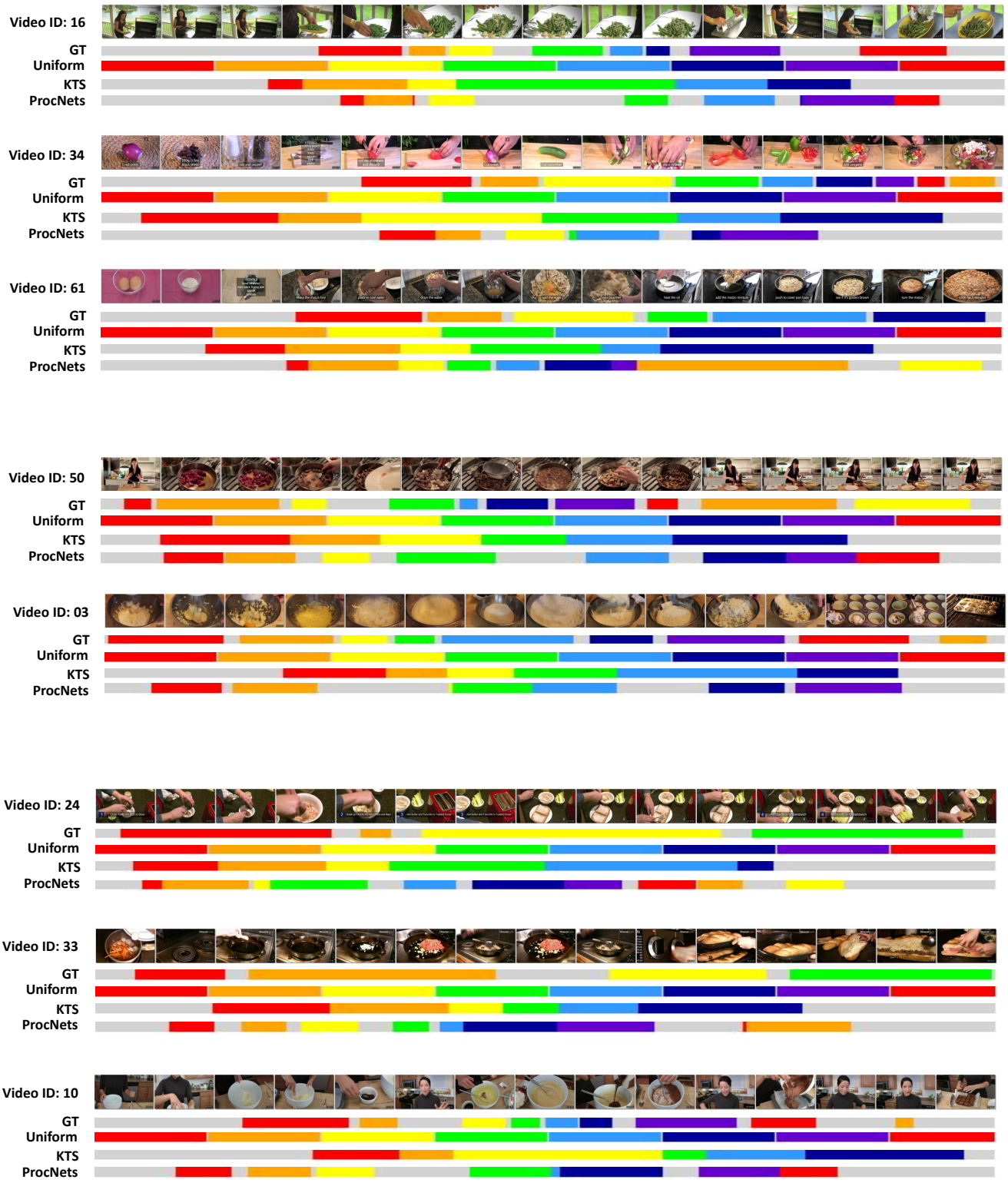

Figure 10. Procedure segmentation results on YouCook dataset.

Label the primary recipes in the time interval of the video

We are targeting to collect **25 videos for each recipe*

America	European/Middle East	East Asia	South Asia
BLT(25/25) onion rings(25/25) burger(25/25) scrambled eggs(25/25) fried chicken(25/25) macaroni and cheese(25/25) calamari(25/25) pancake(25/25) buffalo wings(25/25) caesar salad(25/25) waldorf salad(25/25) pasta salad(25/25) grilled cheese(25/25) mashed potato(25/25) corn dogs(25/25) pepperoni pizza(25/25) eggs benedict(25/25) ~~fried eggs~~(1/25) meatloaf(25/25) hash browns(25/25) clam chowder(25/25) tomato soup(25/25) ~~poutine~~(0/25) hot dogs(25/25) ~~baked potato~~(0/25) beef tacos(25/25) bean burrito(25/25) ~~fajita~~(0/25) ~~enchilada~~(5/25) ~~tamales~~(0/25)	chicken parmesan(25/25) minestrone(25/25) spaghetti and meatballs(25/25) penne alla vodka(25/25) pizza marghetta(25/25) spaghetti carbonara(25/25) fish and chips(25/25) bangers and mash(25/25) shepherd's pie(25/25) boxty(25/25) colcannon(25/25) cottage pie(25/25) croque monsieur(25/25) foie gras(25/25) escargot(25/25) bratwurst(25/25) currywurst(11/25) pierogi(25/25) sauerkraut(25/25) ~~porkolt hungarian stew~~(5/25) goulash(25/25) mussels(25/25) beef bourguignon(25/25) wiener schnitzel(25/25) pasta e fagioli(25/25) fattoush(25/25) tabbouleh(25/25) hummus(25/25) falafel(25/25) shish kabob(25/25)	kung pao chicken(25/25) chinese spring rolls(25/25) mapo tofu(25/25) yaki udon noodle(25/25) kimchi(25/25) shrimp tempura(25/25) california roll(25/25) spicy tuna roll(25/25) potstickers(25/25) tuna sashimi(25/25) salmon sashimi(25/25) ~~tuna nigiri~~(5/25) salmon nigiri(25/25) authentic japanese ramen(25/25) ~~spider roll~~(5/25) miso soup(25/25) bulgogi(25/25) galbi(25/25) bibimbap(25/25) ~~sichuan boiled fish~~(2/25) general's chicken(25/25) ~~pork lo mein~~(8/25) pork fried rice(25/25) udon noodle soup(25/25) sour soup(25/25)	indian chicken curry(25/25) ~~pho~~(0/25) pad thai(25/25) singapore rice noodle(25/25) indian lamb curry(25/25) vietnam spring roll(25/25) ~~thai green curry chicken~~(8/25) ~~thai red curry chicken~~(0/25) vegetable biryani(25/25) chapati(25/25) ~~tom yum goong~~(0/25) thai fried rice(25/25) vietnam sandwich(25/25) ~~char siu~~(0/25) ~~roast goose~~(0/25) dal makhani(25/25) ~~roti jala~~(10/25) chana masala(25/25) naan(25/25) ~~shumai~~(2/25) samosa(25/25) wanton noodle(25/25) singapore curry laksa(25/25) ~~hainanese chicken rice~~(0/25) masala dosa(25/25)
Accomplished: 80.8%	Accomplished: 95.466666666667%	Accomplished: 87.2%	Accomplished: 67.2%

Total Collected: **2287**

Figure 11. Cooking video acquisition status by recipe type. The goal for each recipe is to acquire 25 unique videos. For some recipes, existing video tutorials are rare. We end up picking 90 recipes closest to the goal to construct the YouCookII dataset.

Figure 12. Top 100 words in the YouCookII recipe descriptions. Image generated from TagCrowd.

Table 7. An example of the annotated recipes. ID: 4eWzsx1vAi8.

Step ID	Starting Time	Ending Time	Description
1	00:00:21	00:00:51	Grill the tomatoes in a pan and then put them in a plate
2	00:00:54	00:01:03	Add oil to a pan and spread it well so as to fry the bacon
3	00:01:06	00:01:54	Cook bacon until crispy, then drain on paper towel
4	00:01:56	00:02:40	Add a bit of Worcestershire sauce to mayonnaise and spread it over the bread
5	00:02:41	00:03:00	Place a piece of lettuce as the first layer, place the tomatoes over it
6	00:03:08	00:03:15	Sprinkle salt and pepper to taste
7	00:03:16	00:03:25	Place the bacon at the top
8	00:03:25	00:03:28	Place a piece of bread at the top

by the user-uploaded recipe descriptions, mostly of which are in casual forms. The full requirement of the YouCookII annotation task is shown below.

- Each clip (recipe step) is less than two minutes long.
- Each recipe includes 5 to 15 steps; each step should be described with one sentence. Ordering among the steps counts.
- Generally, each sentence description should have 5 to 20 words, use proper grammar and punctuation, and be in imperative form.
- The starting time and the ending time for each step should be recorded in the form HH:MM:SS.
- For some videos, the user-uploaded recipe is present in the text box. For the annotation task, these recipes should be discarded and be not read or used. Only use the video/audio/subtitle to generate the recipe description.

An example of the annotation is shown in Tab. 7. Due to the complexity of fine-grained recipe annotation, we hire well-trained native English speakers as the annotators instead of crowdsourcing. Also, the segment annotations are rather subjective. To eliminate the annotation bias, we involve as many annotators as we can.

So far, fine-grained recognition from a raw video stream is rather challenging, especially for some ingredients and utensils in cooking videos. We might further annotate the dataset by spatiotemporally segmenting cooking tools, ingredients, related objects and actions in the future update of YouCookII.

Dataset Statistics. The total video time is 160 hours with an average length of 4.84 mins for each video. The distribution of number of clips per video is shown in Fig. 13. 5-9 clips per video are rather common in the dataset. The length of procedure segments on the whole dataset is distributed as in Fig. 14 with a mean of 20.4 seconds and a standard deviation of 19.0 seconds. For the recipe descriptions, the total vocabulary is around 3400 and the top 100 frequent words

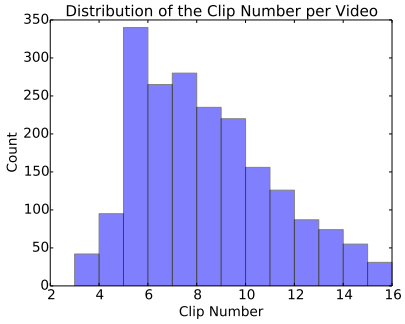

Figure 13. Distribution of number of clips per video.

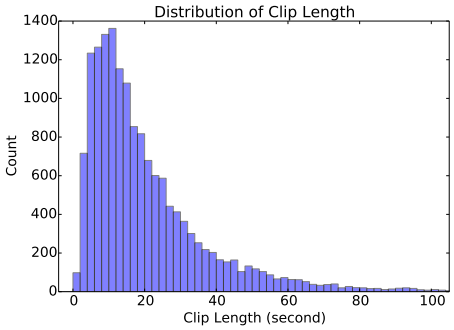

Figure 14. Clip length distribution in the YouCookII dataset.

are shown in Fig. 12. Other statistics include: 1) the majority of the recipes are operated by a single person, 2) 56% of the video sound is in English speech and the rest is in native language.

Precomputed Feature. We uniformly down-sample 500 frames for each video in YouCookII. The average sample rate is 1.73 fps. To further enlarge the training samples, we temporally augment the data, i.e., sample each video 10 times with temporal shifts. We crop all the video frames to size 224*224. For YouCookII, we provide multiple options for video frame representation, namely, VGG-16 feature [37], ResNet-34 feature [11] and ResNet-200 feature [11]. Note that we provide two versions of ResNet features that are extracted from two pre-trained models. The

first one is trained with ImageNet dataset on image Classification task. The second one is a fine-tuned version of the first model on image captioning tasks [40, 45]. The video feature computed based on the second model contains richer semantic information and yields better procedure segmentation performance in our experiment. Motion features, such as optical flow and dense trajectory, might be provided in the future.