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 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-
Add oil to a pan and spread it well so as to fry the bacon. Cook bacon until crispy, then drain on paper towel. Add a bit of Worcestershire sauce to mayonnaise and spread it over the bread. Place a piece of lettuce as the first layer, place the tomatoes over it. Sprinkle salt and pepper to taste. Place the bacon at the top, place a piece of bread at the top.

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: 4eWzx1vAi8.

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 proposing 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
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 $x = \{x_1, x_2, \ldots, x_L\}$, 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

...
Therefore, the boundaries for the segment proposal are de-

This design of temporal convolutional anchors together with our context-aware video en-

duction 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 $(\theta_i, \theta_l)$ are defined as:

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

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 $S$: 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 $(S, B_i, C_i)$ $i = 1, 2, ..., n$ 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:

$$p(\epsilon_1, ..., \epsilon_S|S, B_1, ..., B_n, C_1, ..., C_n)$$

$$= \sum_{t=0}^{S} \log p(\epsilon_t|S, B_1, ..., B_n, C_1, ..., C_n, \epsilon_0, ..., \epsilon_{t-1}),$$

where $\epsilon_0$ is the special $<\text{start}>$ 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 $S$ 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
To encode the procedure segment content, we mean-pool the video ResNet feature to extract a segment content vector. 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:

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

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

\[
L_{reg} = \frac{1}{U_p} ||B_t - B_t^{(gt)}||_{smooth-l1}
\]

\[
L_{seq} = \frac{1}{S} \sum_{i=1}^{g} \log(P_i)^T 1_t^{(gt)}
\]

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, \(B_t^{(gt)}\) is the ground-truth boundary corresponding to positive sample \(i\), \(P_i\) is the softmax output of LSTM at time \(t\) and \(1_t^{(gt)}\) is one-hot vector of ground-truth segment index. Discount factors \(\alpha_r\) and \(\alpha_s\) 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., parameter 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.
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 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 “grammar” 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 measure [3] is that we penalize all the misalignment of segments in order to evaluate 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.

<table>
<thead>
<tr>
<th>Method (%)</th>
<th>validation</th>
<th>test</th>
<th>validation</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AR@0.5</td>
<td>mIoU</td>
<td>AR@0.5</td>
<td>mIoU</td>
</tr>
<tr>
<td>Uniform</td>
<td>16.1</td>
<td>28.0</td>
<td>16.7</td>
<td>28.0</td>
</tr>
<tr>
<td>KTS [28]</td>
<td>16.1</td>
<td>25.3</td>
<td>13.8</td>
<td>24.3</td>
</tr>
<tr>
<td>vsLSTM [44]</td>
<td>16.7</td>
<td>26.9</td>
<td>12.2</td>
<td>23.9</td>
</tr>
<tr>
<td>S-CNN [34]</td>
<td>11.2</td>
<td>21.2</td>
<td>11.7</td>
<td>22.1</td>
</tr>
<tr>
<td>ProcNets-NMS (ours)</td>
<td>18.3</td>
<td>27.3</td>
<td>19.1</td>
<td>27.6</td>
</tr>
<tr>
<td>ProcNets-LSTM (ours)</td>
<td>24.2</td>
<td>35.3</td>
<td>22.8</td>
<td>33.1</td>
</tr>
</tbody>
</table>

Figure 6. A negative example (ID: Y9vfZGad0) on procedure segmentation. The color order is consistent with the chronological order. Recall=0 owing to the second false positive segment.

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 -SegmentFeat) from the LSTM input.

<table>
<thead>
<tr>
<th></th>
<th>AR@0.5</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full model</td>
<td>10.9</td>
<td>14.0</td>
</tr>
<tr>
<td>-Proposal Vec</td>
<td>8.0</td>
<td>11.4</td>
</tr>
<tr>
<td>-Linear Emb</td>
<td>9.3</td>
<td>13.1</td>
</tr>
<tr>
<td>-SegmentFeat</td>
<td>9.7</td>
<td>13.9</td>
</tr>
</tbody>
</table>

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 the validation set, all the results are on testing data.

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-
Grilled Cheese

Ground-truth:  
Proposed:  

Chapati

Ground-truth:  
Proposed:  

Calamari

Ground-truth:  
Proposed:  

Figure 7. Examples on procedure segmentation. YouTube IDs: BITckNkfmRY, jD4oLmy6bU and jrwHN188H2I.

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

<table>
<thead>
<tr>
<th>$K_h$</th>
<th>$K_w$</th>
<th>AR@0.5</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>2</td>
<td>7.2</td>
<td>10.5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>7.8</td>
<td>10.4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>8.9</td>
<td>11.3</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>8.9</td>
<td>12.3</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>10.3</td>
<td>13.9</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>10.0</td>
<td>13.3</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>9.9</td>
<td>13.2</td>
</tr>
</tbody>
</table>

Table 5. Experiments on discount factors. $\alpha_r$ is for clip regression loss and $\alpha_s$ is for sequence prediction loss. The max pooling kernel size is $8 \times 5$.

<table>
<thead>
<tr>
<th>$\alpha_r$</th>
<th>$\alpha_s$</th>
<th>AR@0.5</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
<td>9.3</td>
<td>13.1</td>
</tr>
<tr>
<td>0.2</td>
<td>0.2</td>
<td>10.0</td>
<td>13.3</td>
</tr>
<tr>
<td>1</td>
<td>0.2</td>
<td>9.8</td>
<td>12.7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>11.8</td>
<td>15.0</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Method (%)</th>
<th>Recall</th>
<th>Precision</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>27.4</td>
<td>23.3</td>
<td>25.2</td>
</tr>
<tr>
<td>KTS [28]</td>
<td>20.9</td>
<td>23.3</td>
<td>22.0</td>
</tr>
<tr>
<td>vsLSTM [44]</td>
<td>22.1</td>
<td>24.1</td>
<td>23.0</td>
</tr>
<tr>
<td>S-CNN [34]</td>
<td>28.2</td>
<td>23.2</td>
<td>25.4</td>
</tr>
<tr>
<td>ProcNets (ours)</td>
<td>37.1</td>
<td>30.4</td>
<td>33.4</td>
</tr>
</tbody>
</table>

ProcNets decreases by over a half in the permutated set and 10%-20% of the videos only have segments predicted at the beginning 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 beginning 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


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 overlapings 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.
- 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 on 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 an 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.](image-url)
Figure 10. Procedure segmentation results on YouCook dataset.
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.

<table>
<thead>
<tr>
<th>America</th>
<th>European/Middle East</th>
<th>East Asia</th>
<th>South Asia</th>
</tr>
</thead>
<tbody>
<tr>
<td>calamari (25/25)</td>
<td>fish and chips (25/25)</td>
<td>california roll (25/25)</td>
<td>thai red curry chicken (0/25)</td>
</tr>
<tr>
<td>caesar salad (25/25)</td>
<td>boxty (25/25)</td>
<td>tuna nigiri (5/25)</td>
<td>tom yum goong (0/25)</td>
</tr>
<tr>
<td>grilled cheese (25/25)</td>
<td>croque monsieur (25/25)</td>
<td>escargot (25/25)</td>
<td>cebu siu (0/25)</td>
</tr>
<tr>
<td>mashed potato (25/25)</td>
<td>foie gras (25/25)</td>
<td>bratwurst (25/25)</td>
<td>roast goose (0/25)</td>
</tr>
<tr>
<td>tomato soup (25/25)</td>
<td>fettuccine (25/25)</td>
<td>fettuccine (25/25)</td>
<td>hairtongue chicken rice (0/25)</td>
</tr>
<tr>
<td>baked potato (0/25)</td>
<td>beef pot roast (25/25)</td>
<td>top tortilla (25/25)</td>
<td></td>
</tr>
<tr>
<td>beef tacos (0/25)</td>
<td>beef pot roast (25/25)</td>
<td>top tortilla (25/25)</td>
<td></td>
</tr>
<tr>
<td>bean burrito (25/25)</td>
<td>beef pot roast (25/25)</td>
<td>top tortilla (25/25)</td>
<td></td>
</tr>
<tr>
<td>fajita (0/25)</td>
<td>beef pot roast (25/25)</td>
<td>top tortilla (25/25)</td>
<td></td>
</tr>
<tr>
<td>enchilada (0/25)</td>
<td>beef pot roast (25/25)</td>
<td>top tortilla (25/25)</td>
<td></td>
</tr>
<tr>
<td>tamale (0/25)</td>
<td>beef pot roast (25/25)</td>
<td>top tortilla (25/25)</td>
<td></td>
</tr>
</tbody>
</table>

Accomplished: 80.8%  Accomplished: 95.466666666667%  Accomplished: 87.2%  Accomplished: 67.2%
Total Collected: 2287

Figure 12. Top 100 words in the YouCookII recipe descriptions. Image generated from TagCrowd.
Table 7. An example of the annotated recipes. ID: 4eWzsx1vAi8.

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

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 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.