Towards Automatic Learning of Procedures from Web Instructional Videos

Luowei Zhou
Robotics Institute
University of Michigan
luozhou@umich.edu

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

Jason J. Corso
Department of EECS
University of Michigan
jjcorso@eecs.umich.edu

Abstract
The potential for agents, whether embodied or software, to learn by observing other agents performing procedures involving objects and actions is rich. Current research on automatic procedure learning heavily relies on action labels or video subtitles, even during the evaluation phase, which makes them infeasible in real-world scenarios. This leads to our question: can the human-consensus structure of a procedure be learned from a large set of long, unconstrained videos (e.g., instructional videos from YouTube) with only visual evidence? To answer this question, we introduce the problem of procedure segmentation—to segment a video procedure into category-independent procedure segments. Given that no large-scale dataset is available for this problem, we collect a large-scale procedure segmentation dataset with procedure segments temporally localized and described; we use cooking videos and name the dataset YouCook2. We propose a segment-level recurrent network for generating procedure segments by modeling the dependencies across segments. The generated segments can be used as pre-processing for other tasks, such as dense video captioning and event parsing. We show in our experiments that the proposed model outperforms competitive baselines in procedure segmentation.

Introduction
Action understanding remains an intensely studied problem-space, e.g., action recognition (Donahue et al. 2015; Wang et al. 2016), action detection (Singh et al. 2016; Yeung et al. 2016; Shou, Wang, and Chang 2016) and action labeling (Kuehne, Richard, and Gall 2017; Huang, Fei-Fei, and Niebles 2016; Bojanowski et al. 2014). These works all emphasize instantaneous or short term actions, which clearly play a role in understanding short or structured videos (Chen et al. 2014). However, for long, unconstrained videos, such as user-uploaded instructional videos of complex tasks—preparing coffee (Kuehne, Arslan, and Serre 2014), changing tires (Alayrac et al. 2016)—learning the steps of accomplishing these tasks and their dependencies is essential, especially for agents’ automatic acquisition of language or manipulation skills from video (Yu and Siskind 2013; Al-Omari et al. 2017; Yu and Siskind 2015).

We define procedure as the sequence of necessary steps comprising such a complex task, and define each individual step as a procedure segment, or simply segment for convenience, inspired by (Sener et al. 2015; Alayrac et al. 2016). For example, there are 8 segments in the making a BLT sandwich video shown in Fig. 1. We represent these segments by their start and end temporal boundaries in a given video. Note that one procedure segment could contain multiple actions, but it should be conceptually compact, i.e., described with a single sentence. The number of procedure segments and their locations reflect human consensus on how the procedure is structured. Can this human-consensus procedure structure be learned by an agent?

To that end, we define the Procedure Segmentation problem as: automatically segment a video containing a procedure into category-independent procedure segments. Although this is a new problem, there are two related, existing problems: event proposal and procedure learning. The event proposal problem (Krishna et al. 2017) is to localize category-independent temporal events from unconstrained videos. Both event proposals and procedure segments can contain multiple actions. However, the event proposal problem emphasizes the recall quality given a large amount of proposals, rather than the identification of a procedure (sequence of segments) from limited but necessary proposals. Events might overlap and are loosely-coupled but procedure segments barely overlap, are closely-coupled and usually have long-term dependencies.

The existing work in procedure learning is less-supervised than that of event proposals (no labels are given for the segments). It emphasizes video-subtitle alignment (Malmaud et al. 2015; Bojanowski et al. 2015) and discovery of common procedure steps of a specific process (Alayrac et al. 2016; Sener et al. 2015). However, the methods proposed in these works make restrictive assumptions: they typically assume either language is concurrently available, e.g., from subtitles, or 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; the subtitles or action sequences automatically generated by machines, e.g., YouTube’s ASR system, are inaccurate and require manual intervention; and many procedures of a certain type, such a specific recipe, will vary the number of steps in different instances (process variation).

Unfortunately, work in neither of these two problems sheds sufficient light on understanding procedure segmen-
Add oil to a pan and spread it well so as to fry the bacon.

Grill the tomatoes in a pan and then put them on a plate.

Cook bacon until crispy, then drain on paper towel.

Sprinkle salt and pepper to taste.

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.

Place the bacon at the top.

Place a piece of bread at the top.

Salsa to mayonnaise and spread it over the bread.

Start time: 00:21                      End time: 00:54          01:06                                            01:56               02:41              03:08  03:16   03:25

Figure 1: An example from the YouCook2 dataset on making a BLT sandwich. Each procedure step has time boundaries annotated and is described by an English sentence. Video from YouTube with ID: 4eWzsx1vAi8.

Our contributions are three-fold. First, we introduce and are the first to tackle the category-independent procedure segmentation problem in untrimmed and unconstrained videos. Second, we collect and distribute a large-scale dataset for procedure segmentation in instructional videos. Third, we propose a segment-level recurrent model for proposing semantically meaningful video segments, in contrast to state-of-the-art methods that model temporal dependencies at the frame-level (Zhang et al. 2016). The output procedure segments of ProcNets can be applied for other tasks, including full-agent-based procedure learning (Yu and Siskind 2013) or smaller-scale video description generation (Yu et al. 2016; Krishna et al. 2017).

Related Work

The approaches in action detection, especially the recent ones based on action proposal (Escorcia et al. 2016), inspire our idea of segmenting video by proposing segment candidates. Early works on action detection mainly use sliding windows for proposing segments (Gaidon, Harchaoui, and Schmid 2013; Oneat, Verbeek, and Schmid 2013). More recently, Shou et al. (Shou, Wang, and Chang 2016) propose a multi-stage convolutional network called Segment CNN (SCNN) and achieves state-of-the-art performance (Jiang et al. 2014). The most similar work to ours is Deep Action Proposals, also DAPs (Escorcia et al. 2016; Krishna et al. 2017), where the model predicts the likelihood of an action proposal to be an action while in our case segment proposal. DAPs determines fixed proposal locations by clustering over the ground-truth segments, while our model learns to localize procedures with anchor offsets, which is a generalization of the location pattern from training to testing instead of directly transferring.

Another topic similar to ours is action segmentation or labeling (Kuehne, Gall, and Serre 2016; Kuehne, Richard, and Gall 2017; Huang, Fei-Fei, and Niebles 2016; Bojanowski et al. 2014). It addresses the problem of segmenting a long video into contiguous segments that correspond to a sequence of actions. Most recently, Huang et al. (Huang, Fei-Fei, and Niebles 2016) propose to enforce action alignment through frame-wise visual similarities. Kuehne et al. (Kuehne, Richard, and Gall 2017) apply Hidden Markov Models (HMM) to learn the likelihood of image features given hidden action states. Both meth-
ods focus on the transitions between adjacent action states, leaving long-range dependencies not captured. Also, these methods generally assume contiguous action segments with limited or no background activities between segments. Yet, background activities are detrimental to the action localization accuracy (Huang, Fei-Fei, and Niebles 2016). We avoid these problems with a segment proposal module followed by a segment-level dependency learning module.

**YouCook2 dataset**

Existing datasets for analyzing instructional videos suffer from either limited videos (Alayrac et al. 2016; Sener et al. 2015) or weak diversity in background (Regneri et al. 2013) and activities (Kuehne, Arslan, and Serre 2014)). They provide limited or no annotations on procedure segment boundaries or descriptions (Sigurdsson et al. 2016). To this end, we collect a novel cooking video dataset, named YouCook2.

YouCook2 contains 2000 videos that are nearly equally-distributed over 89 recipes. The recipes are from four major cuisine locales, e.g., Africa, Americas, Asia and Europe, and have a large variety of cooking styles, methods, ingredients and cookwares. The videos are collected from YouTube, where various challenges, e.g., fast camera motion, camera zooms, video defocus, and scene-type changes are present. Table 1 shows the comparison between YouCook2 and other commonly-used instructional video datasets, e.g., YouCook (Das et al. 2013), MPII (Rohrbach et al. 2012), 50Salads (Stein and McKenna 2013), Coffee (Alayrac et al. 2016), Breakfast (Kuehne, Arslan, and Serre 2014) and Charades (Sigurdsson et al. 2016).

Most of the datasets mentioned above have temporally localized action annotations. Compared to action segments, our procedure segments can contain richer semantic information and better capture the human-involved processes in instructional videos. Due to the variety of instructional processes and how each process can be performed, a fixed set of actions fails to describe the details in the video process (e.g., attributes and fine-grained objects). For example, the attribute “crispy” in the recipe step “cook bacon until crispy then drain on paper towel” (see Fig. 1) cannot be described by any action nor activity labels.

**Annotations**

Each video contains 3–16 procedure segments. The segments are temporally localized (timestamps) and described by English sentences in imperative form (e.g., *grill the tomatoes in a pan*). An example is shown in Fig. 1. The annotators have access to audio and subtitles but are required to organize and summarize the descriptions in their own way. As indicated in prior work (Baldassano et al. 2017), people generally agree with boundaries of salient events in video and hence we collect one annotation per video. To reflect the
Figure 4: Schematic illustration of the ProcNets. The input are the frame-wise ResNet features (by row) for a video. The output are the proposed procedure segments. First, the bi-directional LSTM embeds the ResNet features into context-aware features. Then, the procedure segment proposal module generates segment candidates. Finally, the sequential prediction module selects the final proposals for output. During training, the ground-truth segments are embedded to composite the sequential prediction input, which are replaced with beam-searched segment in testing (as shown in the dashed arrows).

Table 1: Comparisons of instructional video datasets. Un-Cons. stands for Unconstrained Scene and Proc. Ann. is short for Procedure Annotation.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>YouCook</td>
<td>140 m</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MPII</td>
<td>490 m</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>50Salads</td>
<td>320 m</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Coffee</td>
<td>120 m</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Breakfast</td>
<td>67 h</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Charades</td>
<td>82 h</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>YouCook2</td>
<td>176 h</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Statistics and Splits
The average number of segments per video is 7.7 and the mean and standard deviation of the number of procedure segments per recipe are shown in Fig. 2. The distribution of video duration is shown in Fig. 3(a). The total video length is 175.6 hours with an average duration of 5.27 min per video. All the videos remain untrimmed and can be up to 10 min. The distribution of segment durations is shown in Fig. 3(b) with mean and standard deviation of 19.6s and 18.2s, respectively. The longest segment lasts 264s and the shortest one lasts 1s. For the recipe descriptions, the total vocabulary is around 2600 words.

We randomly split the dataset to 67%:23%:10% for training, validation and testing according to each recipe. Note that we also include unseen recipes from other datasets for analyzing the generalization ability of the models discussed.

Procedure Segmentation Networks
We propose Procedure Segmentation Networks (ProcNets) for segmenting an untrimmed and unconstrained video into a sequence of procedure segments. We accomplish this by three core modules: 1) context-aware video encoding; 2) segment proposal module that localizes a handful of proposal candidates; 3) sequential prediction that predicts final segments based on segment-level dependencies among candidates. At training, ProcNets are given ground-truth procedure segment boundaries for each video; no recipe categories or segment descriptions are given. At testing, for any given unseen video, ProcNets propose and localize procedure segments in the video based on their visual appearance and temporal relations. The overall network structure
is shown in Fig. 4 and next, we explain each component.

**Context-Aware Video Encoding**

Define a video as $x = \{x_1, x_2, \ldots, x_L\}$, where $L$ denotes the number of sampled frames and $x_1$ is the frame-wise CNN feature vector with fixed encoding size. In this paper $L = 500$ and encoding size is 512. We use ResNet (He et al. 2016) as the appearance feature extractor for its state-of-the-art performance in image classification. We then forward the ResNet features through a bi-directional long short-term memory (Bi-LSTM) (Graves and Schmidhuber 2005) as context encoding. The outputs (forward and backward) are concatenated with the ResNet feature at each frame and the feature dimension is reduced to the same as ResNet feature for a fair comparison. We call these frame-wise context-aware features, denoted as $b_i = \text{Bi-LSTM}(x_i)$. Empirically, Bi-LSTM encoder outperforms context-free ResNet feature and LSTM-encoded feature by a relative 9% on our evaluation metric.

**Procedure Segment Proposal**

Inspired by the anchor-offset mechanism for spatial object proposal, such as in Faster R-CNN (Ren et al. 2015), we design a set of $K$ explicit anchors for segment proposal. Each anchor has the length: $l_k$ ($k = 1, 2, \ldots, K$) and their centers cover all the frames.

Each anchor-based proposal is represented by a proposal score and two offsets (center and length), from the output of a temporal convolution applied on the context-aware feature. The score indicates the likelihood for an anchor to be a procedure segment and the offsets are used to adjust the proposed segment boundaries. By zero-padding the video encoding at the boundaries (depending on anchor sizes), we obtain score and offset matrices of size $K \times L$ (see upper right of Fig. 4) respectively, and hence the output of proposal module is $K \times L \times 3$. Sigmoid function and Tanh functions are applied for proposal score and offsets, respectively.

We formulate the proposal generation as a classification problem and proposal offset as a regression problem. The segment proposals are classified as procedure segment or non-procedure segment with binary cross-entropy loss applied. During training, the segment proposals having at least 0.8 IoU (Intersection over Union) 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 separately for training. Then for the positive samples, we regress the proposed length and center offsets to the ground-truth ones from a relative scale. Given a ground-truth segment with center $c_o$ and length $l_o$, the target offsets $(\theta_c, \theta_l)$ w.r.t. anchor (center $c_a$ and length $l_a$) are given by:

$$\theta_c = \frac{c_o - c_a}{l_a}, \quad \theta_l = \log \frac{l_o}{l_a}. \quad (1)$$

Smooth $l_1$-loss (Ren et al. 2015) is applied in a standard way. For inference, the proposed offsets adjust the anchor location towards the final prediction location.

**Sequential Prediction**

Contrary to spatial objects, video procedure segments, by their nature, have strong temporal dependencies and yet ambiguous temporal boundaries. Therefore, we treat them differently. Recently, modeling frame-level temporal dependency in video has been explored (Zhang et al. 2016). However, memorizing dependencies over enormous frames is still challenging for recurrent models to date (Singh et al. 2016). In contrast, we propose to learn segment-level dependency because the number of proposal segments could be fewer so learning dependencies over segments are more tractable. By leveraging the segment-level dependency, we predict the sequence of procedure segments while dynamically determine the number of segments to propose.

We use long short-term memory (LSTM) for sequential prediction due to its state-of-the-art performance in sequence modeling (Xu et al. 2016; Zhang et al. 2016). The input of LSTM is constructed from three parts: 1) Proposal Vector $S$: max-pooled proposal scores from the proposal module, fixed over time; 2) Location Embedding $B_t$: a set of vectors that discretely encode the locations of ground-truth or previously generated segments; 3) Segment Content $C_t$: the visual features of the ground-truth or previously generated segments. The tuple $(S, B_t, C_t)$, $t = 1, 2, \ldots, N$, is concatenated as the input to LSTM at each time step $t$. Intuitively, when we learn to choose a few winners from a pool of candidates, we need to know who and how good they are (Proposal Vector), what they look like (Segment Content) and the target candidates (location embedding). We will detail each component after introducing the overall model first.

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

$$\log p(\epsilon_1, \ldots, \epsilon_N | S) = \sum_{t=1}^{N} \log p(\epsilon_t | S, B_{t-1}, C_{t-1}, \epsilon_0, \ldots, \epsilon_{t-1}), \quad \text{where} \ \epsilon_0 \ \text{is the special} \ <\text{start}> \ \text{segment token,} \ B_0 \ \text{is the em-}$$
tered proposals (score and offsets), we flatten the proposal
during training, each ground-truth
segment candidates in a video.
encoded the location and confidence information of all likely
segment is represented by a one-hot vector where the index
of one matches to the nearest proposal candidate as illus-
ated in Fig. 6. This discrete representation of location is
easier to learn than continuous location values. Through a
trated in Fig. 6. This discrete representation of location is
encoded for the visual content for
selected candidate, i.e., who we have and who we need next.

tool for video temporal structure learning. It is infeasible for general video structure inference.
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.

Loss Function
The loss function for procedure segmentation network consists of three parts, the binary cross-entropy loss for pro-
duceness classification, the smooth 1/1-loss (Ren et al. 2015) for offset regression and the cross-entropy loss for sequen-
tial prediction. The formulations are as follows:

\[ L = \alpha_r L_{reg} + \alpha_s L_{seq} \]  
\[ L_{cla} = -\frac{1}{U_p + U_n} \sum_{i=1}^{U_p} \log(S_{i}^{(pos)}) + \sum_{i=1}^{U_n} \log(1 - S_{i}^{(neg)}) \]  
\[ L_{reg} = \frac{1}{U_p} \sum_{i=1}^{U_p} ||\mathbf{B}_i - \mathbf{B}_{i}^{(gt)}||_{smooth-1} \]  
\[ L_{seq} = -\frac{1}{N} \sum_{t=1}^{N} \log(P_{t}^{T} \mathbf{1}_{t}^{(gt)}) \]
where \( U_p \) and \( U_n \) are the number of positive and nega-
tive samples, respectively, \( S_{i}^{(pos)} \) and \( S_{i}^{(neg)} \) represents their
scores, \( \mathbf{B}_i^{(gt)} \) is the ground-truth boundary corresponding to
positive sample \( i \), \( \mathbf{B}_{i} \) is the softmax output of LSTM at time
\( t \) and \( \mathbf{1}_{t}^{(gt)} \) is one-hot vector of ground-truth segment index.
Discount factors \( \alpha_r \) and \( \alpha_s \) are applied to balance the contri-
butions of the regression loss and sequential prediction loss,
respectively. Empirically, equally weighting each part, i.e.
\( \alpha_r = \alpha_s = 1 \), yields good results.

Experiments and Results
In this section, we benchmark our new dataset on proce-
dure segmentation with competitive baselines and our pro-
methds under standard metrics. We also show abla-
tion studies, qualitative results and analysis on the procedure
structure learned by our approach.

Baselines. We compare our methods against state-of-the-
art methods in video summarization and action proposal due
to lack of direct baselines in our new problem. These meth-
ods include: 1) Video Summarization LSTM (vsLSTM)
(Zhang et al. 2016), 2) Segment CNN for proposals (SCNN-
prop) (Shou, Wang, and Chang 2016). The major difference
between ProcNets and vsLSTM is, our model learns the
segment-level temporal dependency while vsLSTM learns

Relations to Other Models. To the best of our knowledge,
we are the first to apply segment-level sequential modeling
on category-independent procedure segments. The proposed
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 (Ren et al. 2015) or action detection (Shou, Wang,
and Chang 2016) is NMS. This approach fails to capture
the temporal structure or segment dependencies of instructional
videos. We consider it as a baseline in our experiment along
with our sequential prediction model.
the frame-level temporal dependency. SCNN-prop is the proposal module of action detector SCNN, which achieves state-of-the-art performance in action proposal.\(^2\) In addition, we also evaluate a uniform segment baseline (denoted as Uniform). Two variants of ProcNets are evaluated, one with all the modules (ProcNets-LSTM) and one that replaces sequential prediction with NMS (ProcNets-NMS). Finally, note that we compare with no action segmentation methods since these approaches require an action pool and directly model the finite action states (e.g., with HMM) which requires the “grammar” of the video procedure; both of these needs violate the core assumptions in this paper.

**Metrics.** For procedure segmentation, we adopt two standard metrics for evaluating segment proposals: Jaccard (Bojanowski et al. 2014) and mean Intersection over Union (mIoU). In Jaccard measure, the maximal intersection over prediction between all the final proposals and each ground-truth segment is computed and averaged. The individual Jaccard for each video is then averaged as the overall Jaccard. mIoU replaces the intersection over prediction in Jaccard with intersection over union (IoU). Hence, mIoU penalizes all the misalignment of segments while Jaccard only penalizes the partition of proposal beyond the ground truth. All the methods except for ProcNets-LSTM output 7 segments per video, determined by the average number of segments in the training set. Note that the average number of proposals from ProcNets-LSTM is also around 7, makes that a fair comparison. Inspired by the average recall metric in action proposal (Escorida et al. 2016), we also report the proposal averaged recall, precision and F1 score but with limited segments (10 per video), as motivated in Introduction section.

**Data Preprocessing.** To preserve the overall information in the videos, we uniformly down-sample 500 frames for each video in YouCook2. The average sample rate is 1.58 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 frame-wise ResNet-34 feature (He et al. 2016),\(^3\) pretrained on both ImageNet (Deng et al. 2009) and MSCOCO caption (Lin et al. 2014; Zhou et al. 2016). Hence, each video is represented as a sequence of image spatial features. Local motion features are not used in our study; they may further improve performance.

**Implementation and Training Details.** The sizes of the temporal conv. kernels (also anchor length) are from 3 to 123 with an interval of 8, which covers 95% of the segment durations in training set. The 16 explicit anchors centered at each frame, i.e., stride for temporal conv. is 1. We randomly select \(U = 100\) samples from all the positive and negative samples respectively and feed in negative samples if positive ones are less than \(U\). Our implementation is in Torch. All the LSTMs have one layer and 512 hidden units. For hyper-parameters, the learning rate is \(4 \times 10^{-5}\). We use the Adam optimizer (Kingma and Ba 2014) for updating weights with \(\alpha = 0.8\) and \(\beta = 0.999\). Note that we disable the CNN

\(^2\) New results comparing DAPs and SCNN-prop; https://github.com/escorcia/daps/wiki

\(^3\) Torch implementation of ResNet by Facebook; https://github.com/facebook/fb.resnet.torch

---

**Table 2: Results on temporal segmentation. Top two scores are highlighted. See text for details.**

<table>
<thead>
<tr>
<th>Method (%)</th>
<th>validation Jaccard</th>
<th>mIoU</th>
<th>test Jaccard</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>41.5</td>
<td>36.0</td>
<td>40.1</td>
<td>35.1</td>
</tr>
<tr>
<td>vsLSTM</td>
<td>47.2</td>
<td>33.9</td>
<td>45.2</td>
<td>32.2</td>
</tr>
<tr>
<td>SCNN-prop</td>
<td>46.3</td>
<td>28.0</td>
<td>45.6</td>
<td>26.7</td>
</tr>
<tr>
<td>ProcNets-NMS (ours)</td>
<td>49.8</td>
<td>35.2</td>
<td>47.6</td>
<td>33.9</td>
</tr>
<tr>
<td>ProcNets-LSTM (ours)</td>
<td>51.5</td>
<td>37.5</td>
<td>50.6</td>
<td>37.0</td>
</tr>
</tbody>
</table>

**Table 3: Ablation study on LSTM input. We remove either Proposal Vector (as -Proposal Vec), Location Embedding (as -Location Emb) or Segment Content (as -SegmentFeat).**

<table>
<thead>
<tr>
<th></th>
<th>Jaccard</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full model</td>
<td>50.6</td>
<td>37.0</td>
</tr>
<tr>
<td>-Proposal Vec</td>
<td>47.6</td>
<td>36.1</td>
</tr>
<tr>
<td>-Location Emb</td>
<td>46.2</td>
<td>35.1</td>
</tr>
<tr>
<td>-SegmentFeat</td>
<td>49.0</td>
<td>36.4</td>
</tr>
</tbody>
</table>

fine-tuning which heavily slows down the training process.

**Procedure Segmentation Results**

We report the procedure segmentation results on both validation and testing sets in Tab. 2. The proposed ProcNets-LSTM model outperforms all other methods by a huge margin in both Jaccard and mIoU. SCNN-prop (Shou, Wang, and Chang 2016) suffers in our sequential segmentation task result from the lack of sequential modeling. vsLSTM (Zhang et al. 2016) models frame-level temporal dependency and shows superior results than SCNN-prop. However, our model learns segment-level temporal dependency and yields better segmentation results, which shows its effectiveness. Uniform baseline shows competitive results and the possible reason is, in instruction videos, generally procedures span the whole video which favors segments that can cover the majority of video. For the rest experiments, all the results are on testing set.

**Ablation study on sequential prediction.** The input of the sequence modeling LSTM is the concatenation of three parts: Proposal Vector, Location Embedding and Segment Content. We remove either one of them as the ablation study. Results are shown in Tab. 3. Unsurprisingly, the proposal scores (Proposal Vector) play a significant role in determining the final proposals. When this information is unavailable, the overall performance drops by 6% on Jaccard relatively. The Location Embedding encodes the location information for ground-truth segments and is the most important component for procedure structure learning. Jaccard and mIoU scores drop by 8.7% and 5.1% relatively when location embedding is not available. The segment visual feature has less impact on the sequence prediction, which implies the visual information represented in the video appearance feature is noisy and less informative.

**Proposal localization accuracy.** We study the proposal localization problem when each model proposes 10 seg-
Table 4: Results on segment localization accuracy. 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>vsLSTM</td>
<td>22.1</td>
<td>24.1</td>
<td>23.0</td>
</tr>
<tr>
<td>SCNN-prop</td>
<td>28.2</td>
<td>23.2</td>
<td>25.4</td>
</tr>
<tr>
<td>ProcNets-NMS</td>
<td>37.1</td>
<td>30.4</td>
<td>33.4</td>
</tr>
</tbody>
</table>

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

Qualitative results. We demonstrate qualitative results with videos from YouCook2 test set (see Fig. 7). The model can accurately localize some of the segments and predict their lengths. Moreover, the number of segments proposed is adapted to individual videos and the model learns to propose fewer segments at the beginning and the end of the video, where usually no cooking processes happen. 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.

Analysis on temporal structure learning. We conduct additional experiments to evaluate the temporal structure learning capability of ProcNets. 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 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). We believe reasons are two. 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.

Conclusion

We introduce a new problem called procedure segmentation to study human consensus on how a procedure is structured from unconstrained videos. Our proposed ProcNets take frame-wise video features as the input and predict procedure segments exist in the video. We evaluate the model against competitive baselines on the newly collected large-scale cooking video dataset with standard metrics and show significant improvements. Besides, ProcNets are capable of inferring the video structure by video content and modeling the temporal dependencies among procedure segments. For future work, there are two extensions of the current work. The first one is dense video captioning. The other one is weakly supervised segmentation, which is to first align the weak audio/subtitle signal with the video and then train our model with the aligned annotation.

Acknowledgement. This work has been supported in part by Google, ARO W911NF-15-1-0354, NSF CNS 1463102, NSF CNS 1628987, NSF NRI 1522904 and NSF BIGDATA 1741472. This article solely reflects the opinions and conclusions of its authors and neither Google, ARO nor NSF. We sincerely thank our colleagues Vikas Dhiman, Madan Ravi Ganesh and Ryan Szeto for their helpful discussions. We also thank Mingyang Zhou, Yichen Yao, Haonan Chen, Carolyn Busch and Dali Zhang for their efforts in constructing the dataset.

References


Kuehne, H.; Richard, A.; and Gall, J. 2017. Weakly supervised learning of actions from transcripts. CVIU.


