Supplementary Materials: Unsupervised Procedure Learning via Joint Dynamic Summarization

Ehsan Elhamifar
Khoury College of Computer Sciences
Northeastern University
eelhami@ccs.neu.edu

Zwe Naing
Khoury College of Computer Sciences
Northeastern University
naing.z@husky.neu.edu

1. Computational Complexity Analysis

Notice that at each iteration of the greedy algorithm, we need to compute $f(\Lambda \cup \{i\})$ for every $i \in \{1, \dots, M\} \backslash \Lambda$, where for any Γ we have

$$\begin{aligned} f(\Gamma) &\triangleq \max_{\{\boldsymbol{r}_\ell \subseteq \Gamma^{T_\ell}\}_{\ell=1}^{L}} \sum_{\ell=1}^{L} \frac{1}{T_\ell} \sum_{t=2}^{T_\ell} w_{t-1,t}^{(\ell)} \\ &= \sum_{\ell=1}^{L} \frac{1}{T_\ell} \times \max_{\boldsymbol{r}_\ell \subseteq \Gamma^{T_\ell}} \Big(\sum_{t=2}^{T_\ell} w_{t-1,t}^{(\ell)} \Big), \end{aligned} \tag{1}$$

for $w_{t-1,t}^{(\ell)}$ defined in the main paper. To compute $f(\Lambda \cup \{i\})$, we need to run dynamic programming. This requires performing T_ℓ maximizations for video ℓ, where each maximization requires computing a table with $|\Lambda + 1|^2$ pairs. Given that $|\Lambda|^2$ pairs have already been computed in the previous iteration of the greedy, we need to compute $O(|\Lambda|)$ new values per video, hence, $O(k \sum_{\ell=1}^{L} T_\ell)$ cost for computing $f(\Lambda \cup \{i\})$ for a fixed i, given that $|\Lambda| \leq k$. At each iteration of the greedy, we need to compute the marginal gain for each i in $\{1, \dots, M\} \backslash \Lambda$, hence, $O(kM \sum_{\ell=1}^{L} T_\ell)$ cost per greedy iteration. Finally, to find k representatives, we need to run the greedy algorithm for k iterations, hence, a total cost of $O(k^2 M \sum_{\ell=1}^{L} T_\ell)$.

2. More Details on ProceL Dataset

Table 1 shows the statistics of the ProceL dataset, for each of the 12 tasks in the dataset. Notice the dataset offers variations in terms of complexity and size of different tasks. While some videos such as 'perform CPR' or 'tie a tie' are shorter, with about 3.5K frames, tasks such as 'setup chromecast', 'replace iPhone battery' and 'replace toilet' are long, with more than 12K frames. The dictionary of tasks such as 'perform CPR' and 'make smoke salmon sandwich' have, respectively, 8 and 9 key-steps, while the tasks of 'change tire' and 'change toilet' have, respectively, 18 and 21 key-steps. In addition, not all key-steps in the dictionary always appear in every video, as shown in the column for average number of key-steps per video. Some tasks show large variations in terms of key-steps. For example, the task 'change tire' has 18 total key-steps, while each video on average contains about 11 key-steps. Similarly, 'change toilet' has 21 total key-steps, while each video on average contains 11 key-steps. On the other hand, some tasks have less variation in terms of the appearance of key-steps. For example, the task 'tie a tie' has 14 total key-steps, while every video on average contains 12 key-steps. Similarly, large variations can be seen across different tasks in terms of the foreground ration, i.e., the ratio of total length of key-steps to the total length of the video. While 'jump-start car' or 'assemble a clarinet' have a small foreground ratio, i.e., contain mostly background subactivities, tasks such as 'make smoke salmon sandwich' or 'change tire' have a large foreground ratio. These factors to a good extent justify the lower/higher performance of most methods on different tasks, as presented in the main paper and below.

To better quantify the variations of videos of each task in terms of having missing or repeated key-steps or same ordering of key-steps, following [1] we compute three scores: frequency of missing key steps, frequency of repeated key steps and consistency of key steps ordering. We follow the definitions in [1] for computing the former two and modify the latter definition to be applicable to our case. Let N be the total number of videos in each task and K be the size of key-step dictionary of each task. We define u_n and g_n as the total number of unique key steps in a video and the total number of annotated key steps (including repeated key steps), respectively. Finally, l_{ij} denotes the longest common subsequence between i-th and j-th videos while c_{ij} is the minimum annotated key steps between the two videos.

Missing Key Steps. The frequency of missing key steps M is defined as the ratio of the unique key steps in videos and the number of key steps in the dictionary. This captures the number of key-steps missing in each video, and the score ranges from 0 to 1. The score will be higher if more videos

Task	# videos	avg # frames	# key-steps	avg # key-steps / video	avg frame size	foreground ratio	missing steps	repeated steps	order consistency
change tire	55	8,832	18	11	1060 x 607	0.88	38.7	7.2	14.0
make coffee	61	7,495	12	7	1029 x 599	0.46	40.7	12.1	22.7
perform cpr	58	3,893	8	6	884 x 515	0.68	31.7	34.1	26.1
jump-start car	60	4,368	14	9	1025 x 602	0.21	37.0	2.4	24.4
repot plant	59	5,693	10	5	1084 x 624	0.52	46.9	34.2	38.0
setup chromecast	49	12,103	12	9	1292 x 727	0.81	22.3	0.9	16.0
assemble clarinet	60	7,087	16	10	962 x 571	0.38	40.4	3.5	35.1
make pbj sandwich	59	5,132	10	6	1037 x 624	0.52	37.1	24.6	25.3
replace iPhone battery	59	12,181	14	10	1255 x 707	0.78	27.1	0.2	10.4
make smoke salmon	41	4,804	9	5	1149 x 674	0.91	47.4	15.3	29.4
tie a tie	70	3,625	14	12	957 x 583	0.71	14.0	3.8	5.8
change toilet	49	13,398	21	11	1137 x 663	0.65	48.1	1.1	16.8

Table 1: Statistics of the ProceL dataset across different tasks.

have missing key-steps,

$$M \triangleq 1 - \frac{\sum_{n=1}^{N} u_n}{KN}. \tag{2}$$

Repeated Key Steps. Similarly, the frequency of repeated key steps R quantifies the repetitions of key steps in videos. It is defined as the ratio of unique key steps and total number of annotated key steps in videos. The score is between 0 and 1, and higher score implies more repetitions of key steps across videos,

$$R \triangleq 1 - \frac{\sum_{n=1}^{N} u_n}{\sum_{n=1}^{N} g_n}. \tag{3}$$

Order Consistency. Finally, order consistency O is defined the average over the consistencies between distinct pairwise videos. In turn, the consistency between a pair of videos is defined as the ratio of the longest common subsequence and the minimum number of annotated key-steps of two videos. This score is also between 0 and 1 and higher score means higher order consistency between pairs of videos,

$$O \triangleq 1 - \sum_{\substack{i,j \in 1:N \\ i \neq j}} \frac{l_{ij}}{c_{ij}}. \tag{4}$$

Table 1 shows that 'tie a tie' has the smallest number of missing key-steps, while 'make smoke salmon' or 'change toilet' have large missing steps. On the other hand, 'replace iPhone battery' and 'set up chromecast' have the smallest number of repeated steps, while 'repot plant' and 'perform cpr' have the largest number of repeated steps, on which our method does well given that it can handle repeated steps. Finally, the tasks 'repot a plant', 'assemble clarinet' and 'perform cpr' are very consistent in their ordering, while 'tie a tie', 'change tire' and 'setup chromecast' videos are less consistent and more challenging for procedure learning.

Figure 1 shows the ground-truth annotations for three sample videos in ProceL from the tasks 'change tire', 'replace iPhone battery' and 'assemble clarinet'. These ground-truth annotations, as discussed in the main paper,

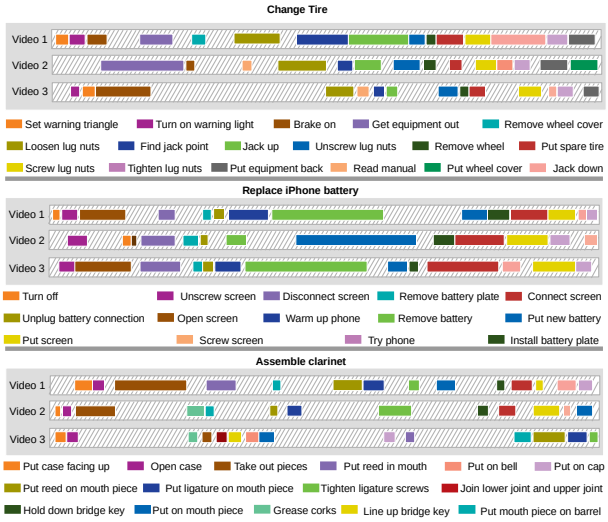

Figure 1: Annotations of key-steps of three videos from the tasks 'change tire', 'replace iPhone battery' and 'assemble clarinet' in the ProceL dataset.

were gathered by consensus between two human annotators, who both watched all the videos. Notice that some key-steps are very short, such as 'unplug battery connection' in 'replace iPhone battery', while some are long, such as 'remove battery' in the same task. Also, 'assemble clarinet' has a larger background compared to other two tasks. Notice that the annotations are, roughly speaking, coarse level not detailing fine-grain descriptions of key-steps, e.g., 'unplug battery connection' does not detail what finer steps need to be done to achieve this key-step.

3. More Information on Feature Extraction

To have a fair comparison to prior work, we use the same features as in [1]. More specifically, for each video segment, we extract two types of features: 1000-dimensional appearance descriptors, which capture the objects present in the scene, and 2000-dimensional motion descriptors, which capture actions across frames. We build the final feature

vector of each superframe by concatenating the two descriptors followed by applying Hellinger normalization.

To obtain appearance descriptors, we take 512-dimensional conv5 layer activations of the VGG16 network applied to frames of the videos of the same task. We then build visual-bag-of-features descriptors by applying KMeans with 1000 centroids on the VGG features and computing the mean of the cluster assignments of frames in each superframe to build its appearance descriptor. For motion descriptors, we extract histogram of optical flows (HOF) features of each superframe using [2], where we set the maximum trajectory length to the length of the superframe. We apply KMeans with 2000 centroids to the HOF features and average the cluster assignments of frames in each superframe to build its motion descriptor. The two types of descriptors provide complementary information, capturing semantic content and motion profiles of each segment.

References

[1] J. B. Alayrac, P. Bojanowski, N. Agrawal, I. Laptev, J. Sivic, and S. Lacoste-Julien, "Unsupervised learning from narrated instruction videos," 2016. 1, 2

[2] H. Wang and C. Schmid, "Action recognition with improved trajectories," *International Conference on Computer Vision*, 2013. 3