Procedure Completion by Learning from Partial Summaries

Zwe Naing
naing.z@husky.neu.edu
Ehsan Elhamifar
e.elhamifar@northeastern.edu

MCADS Lab
Khoury College of Computer Sciences
Northeastern University
Boston, MA, USA

Abstract

We address the problem of procedure completion in videos, which is to find and localize all key-steps of a task given only a small observed subset of key-steps. We cast the problem as learning summarization from partial summaries that allows to incorporate prior knowledge and learn from incomplete key-steps. Given multiple pairs of (video, subset of key-steps), we address the problem by learning representations of input data and finding the remaining key-steps that generalizes well to key-step discovery in new videos. We propose a loss function on the parameters of a network that promotes to recover unseen key-steps that together with the observed key-steps optimize a desired subset selection criterion. To tackle the highly non-convex learning problem, involving both discrete and continuous variables, we develop an efficient learning algorithm that alternates between representation learning and recovering unseen key-steps while incorporating prior knowledge, via a greedy algorithm. By extensive experiments on two instructional video datasets, we show the effectiveness of our framework.

1 Introduction

There exists a large number of instructional videos on the web. YouTube has over 2 billion video search results for the phrase ‘how to’, with more than 500,000 results for tasks such as ‘how to perform CPR’ or ‘how to setup Chromecast’. These instructional videos provide great resource for procedure learning, which is to learn the sequence of key-steps to achieve a certain task. Procedure learning can be used to teach autonomous agents perform complex tasks [48] or help humans in achieving tasks [41], build large instruction knowledge bases, or generate succinct instructions in time constrained settings [33].

The majority of existing work on procedure learning have focused on segmentation of instructional videos when an ordered list of key-steps in each video is given [4, 6, 8, 23, 29, 38, 39, 53] or on unsupervised discovery of key-steps from narrations/text [1, 9, 32, 42, 49] or from visual data [11, 14, 15, 20, 26, 40]. In practice, however, we may have prior knowledge about some of the key-steps of a task, e.g., we may know only the few first steps but not the remaining ones, i.e., we may know how to start a task but do not know how to fully accomplish it. Thus, the goal would be to discover and localize the remaining key-steps using available data, while incorporating the prior knowledge. We refer to this task as procedure completion. Notice that unsupervised procedure learning methods cannot take advantage

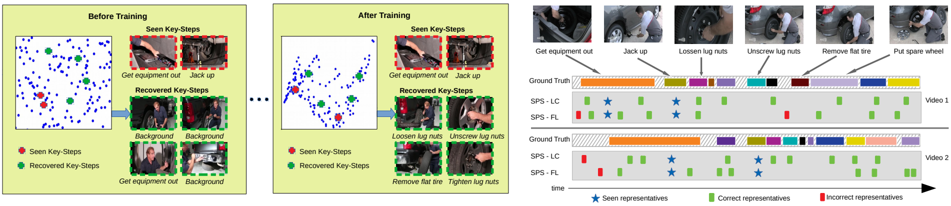

Figure 1: Left: The observed and recovered representatives by our SPS method for the task 'changing tire' from the Inria dataset are shown. While without training, we pick wrong representatives from background, after training we successfully pick the key-steps. Right: Discovered key-steps by SPS for linear coding-based (L) and clustering-based (C) subset selection for two videos, where only two key-steps are observed.

of provided key-steps [1, 14, 15, 26, 40], while supervised methods [52] treat the given key-steps as the entire procedure steps, hence cannot find the remaining ones.

Paper Contributions. In this paper, we formulate and address the problem of procedure completion in videos. We cast the problem as subset selection with partial summaries that allows to incorporate prior knowledge and learn from partial key-steps. More specifically, given multiple pairs of (video, subset of key-steps), we cast the problem as learning representations of input data and finding the remaining key-steps under this representation while generalizing well to new videos. To do so, we propose a loss function on the parameters of a network that promotes to recover unseen key-steps that together with the seen ones optimize a desired subset selection criterion, see Figure 1. To tackle the highly non-convex learning problem, involving both discrete and continuous variables, we develop an efficient learning algorithm that alternates between representation learning (continuous variables) and recovering the unseen key-steps (discrete variables) given prior knowledge, via a greedy algorithm. By experiments on two instructional video datasets, we show the effectiveness of our framework.

Prior Work on Procedure Learning. Depending on the type of supervision, the existing work on learning from instructional videos can be divided into three main categories. The first group of work assumes that annotations of the key-steps (also referred to as procedure steps) are given in videos and the goal is to learn how to segment new videos [52] or anticipate future key-steps [41]. To reduce the costly and unscalable annotation requirement, the second group of work on weakly supervised learning assumes that each video is accompanied with an ordered or unordered list of key-steps appearing in it, and the goals are to localize the key-steps in videos and learn a model for each key-step [4, 8, 23, 38, 53].

Unsupervised procedure learning methods, on the other hand, have focused on exploiting the structure of videos of the same task in order to discover and localize key-steps [1, 15, 20, 26, 40, 42]. Several work have addressed understanding procedures from narration or text [1, 10, 32, 42, 49]. However, reliably obtaining text from spoken natural language using Internet videos still requires manual cleaning the automatic speech recognition results. Moreover, existing methods assume that the text and visual information are aligned [1, 32, 49], which could be violated in real videos. Thus, to learn reliable visual models of key-steps, recent work have focused on learning key-steps directly from visual data [15, 26, 40], using a Mallows model [40], joint sequential summarization [15], embedding and clustering of visual features [26] and self-supervised DNN learning [14]. Unsupervised procedure learning, however, remains a challenging problem, due to large variations in unconstrained videos with substantial amount of background irrelevant activities.

In this paper, we address the new setting in which each video is partially labeled with a small subset of key-steps, with the goal of learning reliable visual models for key-steps and localizing all key-steps in videos.

Prior Work on Subset Selection. Subset selection is the task of finding a small subset of most informative data points from a large dataset. It involves design and optimization of objective functions that characterize the informativeness of selected data points, referred to as representatives. Different criteria have been studied in the literature, including (sequential) facility location [11, 12, 13, 36, 46], maximum cut [30, 34], maximum marginal relevance [5, 31], sparse coding [16, 18] and DPPs [7, 27, 43]. Given that almost all subset selection criteria are, in general, non-convex and NP-hard, approximate methods, such as greedy algorithms for optimizing graph-cuts and (sequential) facility location [11, 24], sampling from Determinantal Point Process (DPP) [27, 28] as well as convex relaxation-based methods [2, 17, 35] have been studied in the literature.

The majority of existing research on subset selection falls into the unsupervised category, where one finds representatives of a dataset by optimizing the above criteria. On the other hand, supervised subset selection has been more recently studied in the literature [7, 19, 21, 22, 37, 43, 45, 47, 50, 51], with the goal of learning from human-generated summaries. More specifically, given a training set composed of multiple datasets and their ground-truth representatives, [47] learns data representations whose input to facility location recovers ground-truth representatives, [22, 45] learn a mixture of submodular functions, and [7, 19, 21, 43, 50] learn a DPP kernel to obtain ground-truth representatives by running subset selection on training data.

Despite its importance, the problem of learning from partial ground-truth summaries has not been addressed in the literature. Applying conventional supervised subset selection to these data results in learning criteria whose optimization on training data will return only the seen representatives, ignoring the fact that many representatives are yet undiscovered, hence, generalizing poorly to test datasets. In the paper, we develop subset selection methods that, for multiple datasets with partial ground-truth representatives, recover the remaining representatives, learn a subset selection criterion whose optimization over training data recovers both seen and unseen representatives, and generalize well to new datasets.

2 Procedure Completion

In this section, we develop a procedure completion framework by formulating the problem as subset selection using partial summaries and proposing an efficient algorithm to solve it. Assume we have a collection of L videos, $\{Y^{(\ell)}\}_{\ell=1}^{L}$, where $Y^{(\ell)} \triangleq \left[y_1^{(\ell)} \dots y_{N_\ell}^{(\ell)}\right] \in \mathbb{R}^{d \times N_\ell}$ corresponds to the matrix of N_ℓ segments/frames from the video ℓ and $y_j^{(\ell)} \in \mathbb{R}^d$ denotes the feature vector of the j-th segment/frame in the video ℓ. Assume we observe some of the ground-truth key-steps for each video, gathered in $\{\mathcal{S}_\ell\}_{\ell=1}^{L}$, where $\mathcal{S}_\ell$ denotes indices of the observed key-steps in video ℓ. The remaining key-steps, denoted by $\{\mathcal{U}_\ell\}_{\ell=1}^{L}$, are unknown. Given a specific subset selection model, $\mathcal{M}$, our goal is to find a transformation of data under which the subset selection model recovers the observed and unobserved key-steps in the videos. We denote by Λ_ℓ the set of all representatives from the video ℓ, i.e.,

$$\Lambda_\ell \triangleq \mathcal{S}_\ell \cup \mathcal{U}_\ell, \;\; \forall \ell = 1, 2, \dots, L, \tag{1}$$

Let $f_\theta : \mathbb{R}^d \to \mathbb{R}^{d'}$ be transformation of data, parametrized by θ, which in our case corresponds to the unknown parameters of a deep neural network. Let

$$f_i^{(\ell)} \triangleq f_\theta(y_i^{(\ell)}) \in \mathbb{R}^{d'}, \;\; F^{(\ell)} \triangleq f_\theta(Y^{(\ell)}) \in \mathbb{R}^{d' \times N_\ell} \tag{2}$$

denote, respectively, the transformation of the frame/segment i in the video ℓ and transformation of the video ℓ, where $f_\theta(\cdot)$ is applied to each column. Finally, we let $F_{\Lambda_\ell}^{(\ell)}$ be the submatrix of $F^{(\ell)}$ whose columns are indexed by Λ_ℓ.

2.1 Proposed Formulation

Given videos with partial key-steps $\left\{(Y^{(\ell)}, \mathcal{S}_\ell)\right\}_{\ell=1}^{L}$, to find the network parameters θ and the unseen key-steps $\{\mathcal{U}_\ell\}_{\ell=1}^{L}$, we propose to minimize a loss function $\mathcal{L}(\theta, \Lambda_1, \dots, \Lambda_L)$ that enforces three desired properties: i) we recover unseen key-steps that together with the observed ones optimize the subset selection criterion, $\mathcal{M}$; ii) the seen and the recovered unseen key-steps capture all modes in the learned embedding space; iii) the transformation extracts meaningful feature representation of data that generalizes well to future test videos.

Notice that the set of all (seen and unseen) key-steps, Λ_ℓ, must well encode the video ℓ. However, the representativeness depends on the particular subset selection criterion, $\mathcal{M}$, which we use. More specifically, we can define the encoding loss function as

$$\mathcal{L}_{enc,\mathcal{M}}(\theta, \Lambda_1, \dots, \Lambda_L) \triangleq \sum_{\ell=1}^{L} d_{\mathcal{M}}(F_{\Lambda_\ell}^{(\ell)}, F^{(\ell)}) \tag{3}$$

where $d_{\mathcal{M}}(\cdot,\cdot)$ measures the goodness of the transformed key-steps $F_{\Lambda_\ell}^{(\ell)}$ for representing the ℓ-th video, $F^{(\ell)}$, under the subset selection model $\mathcal{M}$. In the paper, we advocate using a linear encoding loss that allows having smooth assignment of transition frames between key-steps. More specifically, we assume that each data point can be written as a linear combination of representatives. Hence, the encoding of a subset of frames/segements can be measured by how well they reconstruct the entire video, i.e.,

$$d_{\mathcal{M}}(F_{\Lambda_\ell}^{(\ell)}, F^{(\ell)}) \triangleq \frac{1}{N_\ell} \sum_{j=1}^{N_\ell} \min_{z_j^{(\ell)}} \left(\|\tilde{f}_j^{(\ell)} - \tilde{F}_{\Lambda_\ell}^{(\ell)} z_j^{(\ell)}\|_2^2 + \rho \|z_j^{(\ell)}\|_2^2 \right). \tag{4}$$

The regularization via $\rho \geq 0$ prevents picking redundant (linearly dependent) representatives. Here, $\tilde{f}_j^{(\ell)}$ and $\tilde{F}_{\Lambda_\ell}^{(\ell)}$ are vectors and matrices with normalized columns, in order to remove the undesired effect of different scalings of data (without normalization, we tend to choose columns with large norms as they require small coefficients to reconstruct other points).

On the other hand, the (seen and unseen) key-steps of a video should ideally capture all distribution modes of the feature representation, under the learned transformation. Enforcing diversity leads to the natural desired property of key-steps being well spread. Thus, we consider the loss function

$$\mathcal{L}_{enc,\mathcal{M}}(\theta, \Lambda_1, \dots, \Lambda_L) + \alpha_1 \mathcal{L}_{div}(\theta, \Lambda_1, \dots, \Lambda_L) \tag{5}$$

where $\alpha_1 \geq 0$ is a regularization parameter and the dissimilarity loss, $\mathcal{L}_{dis}$, is define as

$$\mathcal{L}_{dis}(\theta, \Lambda_1, \dots, \Lambda_L) \triangleq -\sum_{\ell=1}^{L} \sum_{i \in \Lambda_\ell} \sum_{\substack{j \in \Lambda_\ell \\ j \neq i}} \|f_i^{(\ell)} - f_j^{(\ell)}\|_2^2, \tag{6}$$

which promotes to embed key-steps in a space where the pairwise Euclidean distances between them are maximized. Notice one could use other loss functions, such as the cosine dissimilarity. However, (6) has the desired property of finding a good Euclidean embedding for data, which also worked well in our experiments. The encoding loss also prevents arbitrary scaling of transformed data by a large value to make the dissimilarity loss smaller.

Given that the key-steps constitute only a small part of the data, learning representation using $\{(Y^{(\ell)}, \mathcal{S}_\ell)\}_{\ell=1}^{L}$ by minimizing the two previous losses is prone to overfitting. The overfitting is more severe in the studied setting, as we are given only a subset of all key-steps. To prevent overfitting and to learn representations that preserve local similarities in data, we take advantage of the available data $\{Y^{(\ell)}\}_{\ell=1}^{L}$ and find representations via autoencoders, which are used to reconstruct the original data. Thus, we propose to solve

$$\begin{aligned} &\min_{\theta,\Lambda_1,\ldots,\Lambda_L} \mathcal{L}(\theta,\Lambda_1,\ldots,\Lambda_L) \triangleq \mathcal{L}_{enc,\mathcal{M}} + \alpha_1 \mathcal{L}_{dis} + \alpha_2 \mathcal{L}_{ae} \\ &\text{s.t.} \quad |\mathcal{U}_\ell| \leq k_\ell, \;\; \Lambda_\ell = \mathcal{S}_\ell \cup \mathcal{U}_\ell, \;\; \forall \ell = 1,\ldots,L, \end{aligned} \tag{7}$$

where $\mathcal{L}_{ae}$ is the autoencoder loss, $\alpha_1, \alpha_2 \geq 0$ are regularization parameters which control the trade-off between the three terms, k_ℓ denotes the budget on the number of unseen key-steps. In the next subsection, we propose an algorithm to efficiently optimize (7).

Remark 1 *We can use other loss functions for the encoding. In the experiments, we compare our linear encoding loss against the facility location loss [24], which is a clustering-based subset selection criterion. It finds an assignment of each point to one representative so that the total encoding cost of the data via representatives is minimized, i.e.,*

$$d_{\mathcal{M}}(F_{\Lambda_\ell}^{(\ell)}, F^{(\ell)}) \triangleq \frac{1}{N_\ell} \sum_{j=1}^{N_\ell} \min_{i \in \Lambda_\ell} \|f_j^{(\ell)} - f_i^{(\ell)}\|_2^2. \tag{8}$$

2.2 Proposed Optimization Algorithm

The loss function in (7) is non-convex, involving both continuous and discrete variables, where in addition to the non-convexity with respect to θ imposed by using a network, the unknown optimization variables θ and $\{\mathcal{U}_\ell\}_{\ell=1}^{L}$ depend on each other. In fact, to find all unseen key-steps $\{\mathcal{U}_\ell\}_{\ell=1}^{L}$, by minimizing $\mathcal{L}$, we must use the transformed data points $\{F^{(\ell)}\}_{\ell=1}^{L}$, which requires knowing the optimal representation parameters θ. On the other hand, to find the optimal embedding parameters θ, we need to have access to all key-steps $\{\mathcal{U}_\ell \cup \mathcal{S}_\ell\}_{\ell=1}^{L}$.

Alternating Minimization. To tackle the above problem, we use an alternating optimization framework, where starting from an initialization of the parameters θ, at each iteration, we first find the optimal set of key-steps given a fixed representation and then update the parameters given the recovered key-steps. We initialize θ by pretraining the autoencoder via minimizing $\mathcal{L}_{ae}$ using all datasets $\{Y^{(\ell)}\}_{\ell=1}^{L}$. Algorithm 1 shows the steps of the method. Finding θ for a fixed $\{\mathcal{U}_\ell\}_{\ell=1}^{L}$ is done by minimizing the loss in (7), where all three losses are being optimized over. On the other hand, finding the unseen key-steps given the parameters requires conditioning on the observed key-steps.

Recovering Unseen Key-Steps: To find the unseen key-steps, we use a greedy approach, motivated by (approximate) submodular function maximization [24], which has been shown to work well for general non-submodular functions as well [3]. Notice that optimizing over unseen key-steps only depends on $\mathcal{L}_{enc}$ and $\mathcal{L}_{dis}$. Thus, for simplicity of notation, for a fixed $\theta = \bar{\theta}$, we define

$$J(\Lambda_1,\ldots,\Lambda_L) \triangleq \mathcal{L}_{enc}\big(\bar{\theta}, \{\Lambda_\ell\}_{\ell=1}^{L}\big) + \alpha_1 \mathcal{L}_{dis}\big(\bar{\theta}, \{\Lambda_\ell\}_{\ell=1}^{L}\big). \tag{9}$$

Algorithm 1 : Procedure Completion

```
Input: Video datasets $\{Y^{(\ell)}\}_{\ell=1}^{L}$ and seen key-steps $\{\mathcal{S}_\ell\}_{\ell=1}^{L}$
Initialize $\theta$ by pretraining the autoencoder using $\{Y^{(\ell)}\}_{\ell=1}^{L}$;
while (Not Converged) do
    For fixed $\theta$, recover unseen key-steps $\{\mathcal{U}_\ell\}_{\ell=1}^{L}$ using Algorithm 2;
    For fixed $\{\mathcal{U}_\ell\}_{\ell=1}^{L}$, set $\Lambda_\ell = \mathcal{S}_\ell \cup \mathcal{U}_\ell$ for $\ell = 1, \dots, L$ and update $\theta$ by minimizing the loss in (7);
end while
Output: Optimal parameters $\theta$ and all key-steps $\{\Lambda_\ell\}_{\ell=1}^{L}$.
```

Given the definitions of the encoding and dissimilarity losses (3) and (6), respectively, we can decompose (9) over individual videos as

$$\begin{aligned} J(\Lambda_1, \dots, \Lambda_L) &\triangleq \sum_{\ell=1}^{L} J^{(\ell)}(\Lambda_\ell), \\ J^{(\ell)}(\Lambda_\ell) &\triangleq d_{\mathcal{M}}\big(F^{(\ell)}_{\Lambda_\ell}, F^{(\ell)}\big) + \alpha_1 dis\big(F^{(\ell)}_{\Lambda_\ell}\big), \end{aligned} \tag{10}$$

where $dis\big(F^{(\ell)}_{\Lambda_\ell}\big)$ is the dissimilarity score for the video ℓ and is computed using the term inside the first summation in (6). Hence, for θ, the minimization over unseen key-steps decomposes into finding unseen key-steps from each video independent from the others.

To minimize $J^{(\ell)}(\Lambda_\ell)$ over unseen key-steps, we take a greedy approach by considering an active set Γ_ℓ that is incrementally grown to select at most k_ℓ unseen key-steps. Initializing Γ_ℓ to $\mathcal{S}_\ell$, at each iteration, we add the data point from the dataset ℓ that minimizes the cost function the most compared to only using Γ_ℓ. More specifically, we find the point $i^* \in \{1, \dots, N_\ell\} \backslash \Gamma_\ell$ for which the gain, $\delta_{\Gamma_\ell}(i) \triangleq J^{(\ell)}(\Gamma_\ell) - J^{(\ell)}(\Gamma_\ell \cup i)$, is maximum. Thus, including the point i^* gives the largest decrease in the loss function. We include i^* in the active set and repeat the process until a budget on the number of unseen key-steps is met or the gain becomes insignificant, i.e., becomes smaller than a predefined threshold. Algorithm 2 shows the steps of the greedy method.

3 Experiments

We evaluate the performance of different algorithms, as a function of the number of observed key-steps, on Inria [1] and Breakfast [25] datasets.

3.1 Algorithms and Baselines

We refer to our method as Summarization with Partial Summaries (SPS). We compare our method against dppLSTM [50], a baseline, referred to as Uniform, and running subset selection on features learned by an LSTM Autoencoder, which we refer to as Unsupervised. Since dppLSTM is a supervised method, we train dppLSTM using the seen key-steps in the training data. For the Uniform baseline, we select k key-steps uniformly at random from all available segments. To demonstrate the effectiveness of our proposed algorithm, we also compare against the unsupervised method, where we run subset selection on the features obtained by the same network used by SPS, but without fine-tuning via our loss function. In fact, when using linear coding, Unsup-LC corresponds to SMRS [16], and when using facility location, Unsup-FL corresponds to Kmedoids [36] and.

Algorithm 2 : Unseen Key-Step Recovery

```
Input: Set function $\{J^{(\ell)}\}_{\ell=1}^{L}$, seen key-steps $\{\mathcal{S}_\ell\}_{\ell=1}^{L}$, budgets $\{k_\ell\}$.
for $\ell = 1,\ldots,L$ do
    Initialize: $\Gamma_\ell = \mathcal{S}_\ell$;
    for $j = 1,\ldots,k_\ell$ do
        for $i \in \{1,\ldots,N_\ell\} \backslash \Gamma_\ell$ do
            Compute gain $\delta_{\Gamma_\ell}(i) = J^{(\ell)}(\Gamma_\ell) - J^{(\ell)}(\Gamma_\ell \cup i)$;
        end for
        Compute $i^* = \mathrm{argmax}_i\, \delta_{\Gamma_\ell}(i)$;
        if $\left(\delta_{\Gamma_\ell}(i^*) \geq \varepsilon\right)$ then
            Update $\Gamma_\ell \leftarrow \Gamma_\ell \cup \{i^*\}$;
        else
            Break;
        end if
    end for
    Set $\Lambda_\ell = \Gamma_\ell$;
end for
Output: Optimal key-step sets $\{\Lambda_\ell\}_{\ell=1}^{L}$.
```

3.2 Datasets

The Inria dataset [1] contains five instructional tasks, such changing tire or making coffee, where each task has 30 videos. The Breakfast dataset [25] has 10 cooking activities by 52 individuals, such as making cereals or frying eggs, where each activity has approximately 200 videos, corresponding to different camera views of the same person doing the same task. On both datasets, the ground-truth annotations provide the frame localization for all key-steps as well as background. A main difference between the two datasets is the amount of background activities (not associated with any key-step). The Inria dataset consists of a large amount of background actions, while the Breakfast dataset has a very small amount background at the beginning and end of each video. We use the entire Inria dataset and the first two splits of the Breakfast dataset for training and evaluation. In our experiments, we divide the videos of each task into 80% for training and 20% for testing.

3.3 Feature Extraction

We perform the subset selection at the segment level, where we divide each video into segments of 10-frame length. We extract visual features from the conv5 and fc2 layers of the VGG16 network, respectively, for the Inria and Breakfast datasets to capture the content of each frame. The conv5 features have already been shown to work well for the Inria dataset [1] and, in our experiments, fc2 features worked best for all methods on the Breakfast dataset. The visual features are then given as input to an LSTM autoencoder [44], which captures the dynamics of the input segment. We set the length of the input sequence of the LSTM to 10, thereby obtaining features for every segment.

3.4 Implementation Details

We use the LSTM Autoencoder with one hidden layer with 2,048 and 4,096 units for Inria and Breakfast, respectively. We use the Adam optimizer with the learning rate of 1e-4 and batch size of 256 during pre-training and run for 1,000 epochs until the reconstruction loss converges. We report the results of the running the facility location and linear coding on features obtained by the pretrained networks, which we refer to as Unsup-FL and Unsup-LC, respectively. Our method, referred to as SPS-FL and SPS-LC, fine-tunes the network

	Inria			**Breakfast**			
Seen Key-Steps	$K = 2$	$K = 4$	$K = 6$	20%	40%	60%	80%
Uniform	26.4	35.2	40.0	45.9	46.3	47.5	48.2
Unsup–FL	24.1	31.6	38.9	43.0	43.6	46.7	47.8
Unsup–LC	26.7	35.1	41.0	48.3	48.3	48.8	49.1
dppLSTM	21.8	32.3	46.7	40.6	42.3	47.1	47.6
SPS–FL	35.1	49.8	59.5	43.0	43.9	46.7	48.1
SPS–LC	**47.9**	**58.0**	**62.1**	**51.8**	**53.1**	**53.3**	**54.7**

Table 1: F1 score (%) comparison for different number/fraction of seen ground-truth key-steps.

using our loss function.

For our method, during training, we fine-tune the pretrained weights of the LSTM Autoencoder on the training set of each task via backpropagation. For training on Inria, we use Adam optimizer with learning rate of 1e-4 for both linear coding and facility location models and fine-tune the weights for 25 epochs (the loss converges around 25 epochs). We perform backpropagation in each epoch using all training videos. We set the hyperparameters α_1 and α_2 to one. In addition, the regularization parameter ρ is set to zero for the linear coding model. For training on Breakfast, we set the learning rate of Adam optimizer to 1e-5, for both version of our method. Given that Breakfast contains many redundant video for each task, e.g., multiple viewpoints of a person demonstrating a cooking activity, we perform backpropagation using SGD for each epoch by selecting 25 videos randomly without replacement. We train for 15 epochs for each task in the Breakfast dataset. We set α_1 and α_2 to 1e-5 and 1, respectively, and set ρ to 1e-4.

3.5 Evaluation Metrics

To investigate the performance of our framework as a function of the number of observed key-steps in videos, we reveal 2, 4 and 6 key-steps for each task in the Inria dataset and 20%, 40%, 60% and 80% for each task in the Breakfast dataset[1]. We pick the middle segment in the ground-truth of each selected key-step as the observed key-step. For all methods, we set the budget k_ℓ to the number of unseen ground-truth key-steps in each video.

For evaluation, we report segment-wise precision, action-wise recall and F1 score [25]. These metrics measure the performance of finding a representative for each key-step and the correctness of video segmentation based on assignments of segments to representatives. The action-wise recall of a video is defined as the number of correctly localized key-steps divided by the number of key-steps in the video. A key-step is correctly localized if at least one representative segment overlaps with the ground-truth of the key-step. To compute the segment-wise precision, we assign each segment of a video to its closest representative and divide the number of segments in ground-truth key-steps assigned to a correctly localized representative by the total number of segments in all key-steps of the video. The F1 score is the harmonic mean between action-wise recall and segment-wise precision.

3.6 Experimental Results

Table 1 shows the average F1 score of different algorithms on Inria and breakfast datasets, respectively, for different number/percentage of seen ground-truth key-steps. On Inria, both

[1] We set the minimum number of seen key steps to 2 for the Breakfast dataset so that there are at least two revealed key-steps when videos have very few key steps.

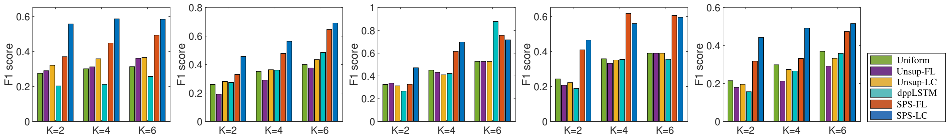

Figure 2: F1 scores on the Inria dataset for different number of observed key-steps, K. Tasks from left to right: change tire, make coffee, perform CPR, repot plant, jump-start car.

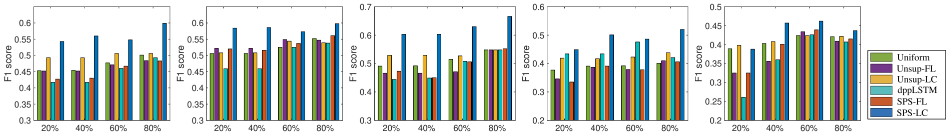

Figure 3: F1 scores on 5 activities from the Breakfast dataset for different percentages of observed ground-truth key-steps. Tasks from left to right: milk, juice, cereals, scrambled eggs, pancake.

versions of our method significantly outperform other algorithms for all values of K. dppLSTM performs poorly for $K = 2,4$ and does better than Unsup and Uniform for $K = 6$. This is expected, since dppLSTM learns to output only seen key-steps that when K is small, leads to poor learning and generalization. Notice that in both datasets, SPS-LC improves over Unsup-LC (and similarly for FL), showing the effectiveness of our proposed loss function, which fine-tunes the network weights learned via Unsup. In both datasets, SPS-LC performs better than SPS-FL, while the latter does not perform as well on Breakfast. This is due to the fact that FL is a clustering-based subset selection that imposes and prefers clear clustering of data, which does not hold well in the beginning of training and particularly on Breakfast that has visually similar segments. We believe visually similar segments in Breakfast is the reason that dppLSTM does not work well for more seen key-steps. Uniform on Breakfast does well, since its videos have very small background, hence, randomly selecting a segment has a higher chance of picking an unseen key-step.

Figures 2 and 3 shows the F1 scores of different methods for each task in the Inria and Breakfast datasets, respectively. Notice that on Inria both versions of our method, i.e., SPS-LC and SPS-FL, perform better than their unsupervised counterparts and the supervised dppLSTM in all tasks. On Breakfast, SPS-LC performs significantly better than other methods, while the performance of SPS-FL is close to its unsupervised version. We believe this is due to high visual similarity of key-steps in the dataset, which makes learning for a clustering-based subset selection hard. Table 2 (left) shows the average precision and recall of our method on Inria for different values of K. Notice that the precision and recall for SPS-FL is much lower for $K = 2$ compared to SPS-LC, since clustering of data using only 2 seen key-steps with a large number of unseen ones at the beginning of training has a large error, which propagates into subsequent steps. However, both models improve the action-wise recall by a large margin as we increase the number of seen key-steps. For larger seen key-steps, both models are effective.

Figure 4 (left) shows the precision and F1 score of SPS-LC on the task CPR from Inria for $K = 2$, demonstrating that within 10 epochs the scores significantly improve and does not change much afterwards (notice the oscillation as we do not optimize the scores directly).

Ablation Studies. Table 2 (right) shows the effect of different components of the loss function in (7) on the F1 score on Inria. Notice that simply training the autoencoder without ground-truth key-steps, as well as only optimizing dissimilarity loss or encoding do not do well. Adding encoding or dissimilarity loss to the autoencoder improves the performance for

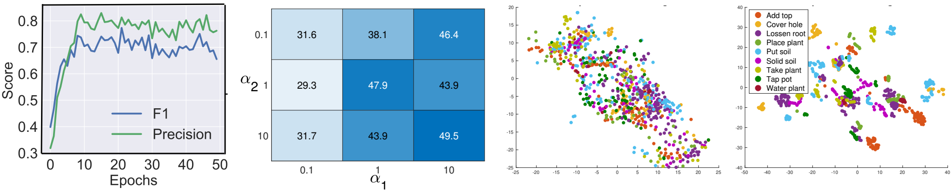

Figure 4: Left: F1 score of SPS-LC as a function of training epochs for $K = 2$ on the task CPR from Inria. Middle Left: Effect of hyperparameters α_1 and α_2 on F1 score. Middle Right and Right: T-SNE visualization of feature representations from five videos of the task 'Repot Plant' with two observed key-steps before (left) and after (right) training using SPS-LC. Circles of different colors correspond to segments from different key steps. The data distribution changes after training where segments belonging to the same key step become better clustered.

	K	Segment-wise precision	Action-wise recall
SPS-FL	2	30.1	45.0
	4	41.4	64.7
	6	47.3	83.0
SPS-LC	2	43.6	57.0
	4	48.8	75.4
	6	49.7	86.1

	SPS-LC	SPS-FL
$\mathcal{L}_{ae}$	26.7	24.1
$\mathcal{L}_{enc}$	28.1	25.9
$\mathcal{L}_{dis}$	27.8	26.2
$\mathcal{L}_{ae} + \mathcal{L}_{enc}$	33.4	23.7
$\mathcal{L}_{ae} + \mathcal{L}_{dis}$	32.0	28.1
$\mathcal{L}_{ae} + \mathcal{L}_{enc} + \mathcal{L}_{dis}$	47.9	35.1

Table 2: Left: Segment-wise precision (%) and action-wise recall (%) of our method on Inria for different numbers of observed key-steps, K. Right: Effect of different loss functions on F1 score (%) on Inria for $K = 2$.

SPS-LC. Finally, using all losses significantly improves the F1 score of both linear coding and clustering-based variants of our method. This is because the dissimilarity loss encourages to select key-steps that are visually dissimilar, while without it, the method tends to select representatives that belong to the same key step.

Effect of Hyperparameters. The second plot in Figure 4 shows the effect of hyperparameters α_1 and α_2 of our loss function on the average F1 score (%) for SPS-LC with $K = 2$ on Inria. When we have enough and almost equal emphasis on the diversity and reconstruction losses, our method performs well, demonstrating the importance of both losses. Notice that the performance is very low for small α_1, which shows the importance of the dissimilarity term in our loss function.

Qualitative Results. The third and fourth plots in Figure 4 shows the T-SNE plots for five videos from the task 'repot plant' in Inria before and after training by our method. Notice that different key-steps, which are initially mixed, become better clustered after training by our method. Finally, Figure 1 shows the ground-truth and key-steps discovered by our method for two training videos from the task 'change tire' in Inria. Notice thar SPS-LC correctly 7 key-steps in both video 1 and 2, while only finding 1 key-step from background in video 1. Also the results show we might recover more than one representative for the same key-step. This often happens when a key-step has large visual appearance variation within a video. Handling such cases would be the subject of future studies.

4 Conclusions

We addressed the problem of procedure completion by learning from a small observed subset of key-steps. We developed a subset selection method from partial summaries and an efficient optimization algorithm, by simultaneously learning representations of input videos and recovering remaining key-steps. By experiments on real instructional videos, we showed the effectiveness of our framework.

Acknowledgements

This work is partially supported by DARPA Young Faculty Award (D18AP00050), NSF (IIS-1657197), ONR (N000141812132) and ARO (W911NF1810300). Zwe Naing performed the research during his research assistantship in MCADS Lab at Northeastern University.

References

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

[2] P. Awasthi, A. S. Bandeira, M. Charikar, R. Krishnaswamy, S. Villar, and R. Ward. Relax, no need to round: Integrality of clustering formulations. *Conference on Innovations in Theoretical Computer Science (ITCS)*, 2015.

[3] A. A. Bian, J. M. Buhmann, A. Krause, and S. Tschiatschek. Guarantees for greedy maximization of non-submodular functions with applications. *International Conference on Machine Learning*, 2017.

[4] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Weakly supervised action labeling in videos under ordering constraints. *European Conference on Computer Vision*, 2014.

[5] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. *SIGIR*, 1998.

[6] C. Y. Chang, D. A. Huang, L. Fei-Fei Y. Sui, and J. C. Niebles. D3tw: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[7] W. L. Chao, B. Gong, K. Grauman, and F. Sha. Large-margin determinantal point processes. *Uncertainty in Artificial Intelligence*, 2015.

[8] Li Ding and Chenliang Xu. Weakly-supervised action segmentation with iterative soft boundary assignment. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[9] H. Doughty, I. Laptev, W. Mayol-Cuevas, and D. Damen. Action modifiers: Learning from adverbs in instructional videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[10] Xinya Du, Bhavana Dalvi Mishra, Niket Tandon, Antoine Bosselut, Wen tau Yih, Peter Clark, and Claire Cardie. Weakly-supervised action segmentation with iterative soft boundary assignment. *Annual Meeting of the North American Association for Computational Linguistics*, 2019.

[11] E. Elhamifar. Sequential facility location: Approximate submodularity and greedy algorithm. *International Conference on Machine Learning*, 2019.

[12] E. Elhamifar and M. C. De-Paolis-Kaluza. Online summarization via submodular and convex optimization. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[13] E. Elhamifar and M. C. De-Paolis-Kaluza. Subset selection and summarization in sequential data. *Neural Information Processing Systems*, 2017.

[14] E. Elhamifar and D. Huynh. Self-supervised multi-task procedure learning from instructional videos. *European Conference on Computer Vision*, 2020.

[15] E. Elhamifar and Z. Naing. Unsupervised procedure learning via joint dynamic summarization. *International Conference on Computer Vision*, 2019.

[16] E. Elhamifar, G. Sapiro, and R. Vidal. See all by looking at a few: Sparse modeling for finding representative objects. *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[17] E. Elhamifar, G. Sapiro, and S. S. Sastry. Dissimilarity-based sparse subset selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[18] E. Esser, M. Moller, S. Osher, G. Sapiro, and J. Xin. A convex model for non-negative matrix factorization and dimensionality reduction on physical space. *IEEE Transactions on Image Processing*, 21(7):3239–3252, 2012.

[19] J. Gillenwater, A. Kulesza, E. Fox, and B. Taskar. Expectation-maximization for learning determinantal point processes. *Neural Information Processing Systems*, 2014.

[20] Karan Goel and Emma Brunskill. Learning procedural abstractions and evaluating discrete latent temporal structure. *International Conference on Learning Representation*, 2019.

[21] B. Gong, W. Chao, K. Grauman, and F. Sha. Diverse sequential subset selection for supervised video summarization. *Neural Information Processing Systems*, 2014.

[22] M. Gygli, H. Grabner, , and L. Van Gool. Video summarization by learning submodular mixtures of objectives. *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[23] D. A. Huang, L. Fei-Fei, and J. C. Niebles. Connectionist temporal modeling for weakly supervised action labeling. *European Conference on Computer Vision*, 2016.

[24] Andreas Krause and Daniel Golovin. Submodular function maximization. Cambridge University Press, 2014.

[25] H. Kuehne, A. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human. *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[26] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. Unsupervised learning of action classes with continuous temporal embedding. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[27] A. Kulesza and B. Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5, 2012.

[28] C. Li, S. Jegelka, and S. Sra. Efficient sampling for k-determinantal point processes. *Conference on Artificial Intelligence and Statistics*, 2016.

[29] J. Li, P. Lei, and S. Todorovic. Weakly supervised energy-based learning for action segmentation. *International Conference on Computer Vision*, 2019.

[30] H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010.

[31] H. Lin and J. Bilmes. A class of submodular functions for document summarization. *Annual Meeting of the Association for Computational Linguistics*, 2011.

[32] Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, and Kevin Murphy. What's cookin'? interpreting cooking videos using text, speech and vision. *NAACL*, 2015.

[33] Nina Mishra, Ryen W White, Samuel Ieong, and Eric Horvitz. Time-critical search. *International ACM SIGIR conference on Research & development in information retrieval*, 2014.

[34] R. Motwani and P. Raghavan. Randomized algorithms. *Cambridge University Press, New York*, 1995.

[35] A. Nellore and R. Ward. Recovery guarantees for exemplar-based clustering. *Information and Computation*, 2015.

[36] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14, 1978.

[37] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid. Category-specific video summarization. *European Conference on Computer Vision*, 2014.

[38] A. Richard, H. Kuehne, and J. Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[39] A. Richard, H. Kuehne, A. Iqbal, and J. Gall. Neuralnetwork-viterbi: A framework for weakly supervised video learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[40] Fadime Sener and Angela Yao. Unsupervised learning and segmentation of complex activities from video. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[41] Fadime Sener and Angela Yao. Zero-shot anticipation for instructional activities. *International Conference on Computer Vision*, 2019.

[42] Ozan Sener, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Unsupervised semantic parsing of video collections. *IEEE International Conference on Computer Vision*, 2015.

[43] A. Sharghi, A. Borji, C. Li, T. Yang, and B. Gong. Improving sequential determinantal point processes for supervised video summarization. *European Conference on Computer Vision*, 2018.

[44] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015.

[45] S. Tschiatschek, R. Iyer, H. Wei, and J. Bilmes. Learning mixtures of submodular functions for image collection summarization. *Neural Information Processing Systems*, 2014.

[46] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes. Using document summarization techniques for speech data subset selection. *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2013.

[47] C. Xu and E. Elhamifar. Deep supervised summarization: Algorithm and application to learning instructions. *Neural Information Processing Systems*, 2019.

[48] Yezhou Yang, Yi Li, Cornelia Fermüller, and Yiannis Aloimonos. Robot learning manipulation action plans by watching unconstrained videos from the world wide web. *AAAI*, 2015.

[49] Shoou-I Yu, Lu Jiang, and Alexander Hauptmann. Instructional videos for unsupervised harvesting and learning of action examples. *ACM International Conference on Multimedia*, 2014.

[50] K. Zhang, W. Chao, F. Sha, and K. Grauman. Video summarization with long short-term memory. *European Conference on Computer Vision*, 2016.

[51] K. Zhang, W. L. Chao, F. Sha, and K. Grauman. Summary transfer: Exemplar-based subset selection for video summarization. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[52] Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. *AAAI*, 2018.

[53] D. Zhukov, J. B. Alayrac, R. G. Cinbis, D. Fouhey, I. Laptev, and J. Sivic. Cross-task weakly supervised learning from instructional videos. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.