United We Stand, Divided We Fall:
UnityGraph for Unsupervised Procedure Learning from Videos

Siddhant Bansal*
CVIT, IIIT, Hyderabad

Chetan Arora
IIT, Delhi

C.V. Jawahar
CVIT, IIIT, Hyderabad

Abstract

Given multiple videos of the same task, procedure learning addresses identifying the key-steps and determining their order to perform the task. For this purpose, existing approaches use the signal generated from a pair of videos. This makes key-steps discovery challenging as the algorithms lack inter-videos perspective. Instead, we propose an unsupervised Graph-based Procedure Learning (GPL) framework. GPL consists of the novel UnityGraph that represents all the videos of a task as a graph to obtain both intra-video and inter-videos context. Further, to obtain similar embeddings for the same key-steps, the embeddings of UnityGraph are updated in an unsupervised manner using the Node2Vec algorithm. Finally, to identify the key-steps, we cluster the embeddings using KMeans. We test GPL on benchmark ProceL, CrossTask, and EgoProceL datasets and achieve an average improvement of 2% *on third-person datasets and* 3.6% *on EgoProceL over the state-of-the-art.*

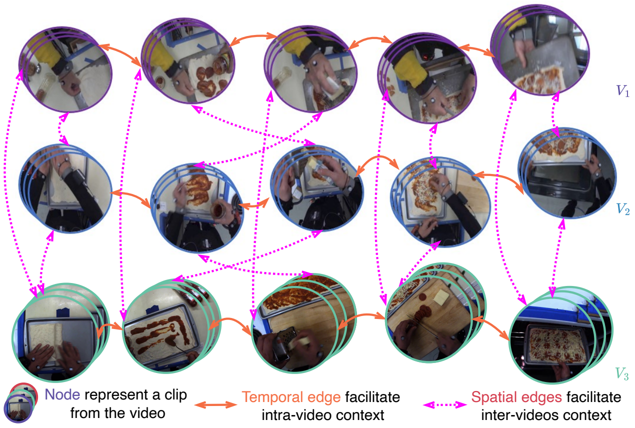

V1
V2
V3
Node represent a clip from the video
Temporal edge facilitate intra-video context
Spatial edges facilitate inter-videos context

Figure 1. **UnityGraph** for three pizza making videos. UnityGraph facilitates procedure learning by creating a unified representation of an arbitrary number of videos from the same category. Here, the nodes represent a clip from the video. Further, the temporal edges connect temporally close frames, allowing intra-video context, whereas the spatial edges connect semantically similar frames across the videos, enabling inter-videos context.

1. Introduction

Motivation: Consider developing a robot capable of assembling a phone in a factory. Hard coding the sequence of steps required to piece together the phone will require years of effort. Instead, it would be useful if a robot could observe a person fabricating the phone multiple times and learns from it! Driven by this objective, we focus on unsupervised procedure learning from videos. Broadly, the task involves identifying the key-steps and their order via multiple videos of the same activity.

Applications: A framework that can distill the steps required to perform a task from multiple demonstrations could help develop robots for manufacturing pipelines, create assistive machines , or monitor and guide a novice learning a new task. Formally, given multiple videos of a task, procedure learning deals with **(a)** identifying the key-steps and **(b)** their order to perform the task [3, 15, 16, 50].

Difference from action segmentation/detection: As shown in Figure 1, procedure learning deals with multiple videos of a task. In contrast, action-based tasks [32] deal with a single video, hence losing the capability to determine repetitive key-steps across the videos. Secondly, these tasks do not consider the order of individual events, which is often crucial for identifying key-steps, and/or procedures/recipes. For example, action-based tasks do not capture the difference in the order of key-steps in V2 and V3 (Fig. 1). Other efforts for video understanding using instructional videos aim at procedure planning [63], procedure sequence verification [44], and instructional video summarisation [41]. Furthermore, as procedure learning deals with localising the key-steps, it differs from the video alignment task [8, 14]. Therefore, considering the utility of procedure learning and its distinctness from existing tasks, we devise the Graph-based Procedure Learning (GPL) framework.

Our approach: GPL is a three-staged unsupervised framework for procedure learning. The first stage of GPL consists

*Corresponding author: siddhant.bansal@research.iiit.ac.in

of UnityGraph, shown in Figure 1. UnityGraph is a graph that models an arbitrary number of videos from the same task. For creating UnityGraph at clip-level, the video clips are first passed through a pre-trained I3D ResNet-50 [5, 22] to get a node. The nodes are later connected based on **(a)** semantic similarity across videos (spatial edges) and **(b)** temporal closeness in the same video (temporal edges). Due to this structure, UnityGraph captures both the inter-video and intra-videos context. This sets UnityGraph apart from the previous approaches that estimate the procedure using one [15, 16, 27, 31, 52] or two [3] videos.

Justification of the approach: As a graph enables us to create a spatial edge between two nodes from different videos irrespective of the key-step order (Figure 1), it enables us to overcome previous works' key-step ordering constraints [31, 52]. Also, the range of granularity (number of frames) to create the nodes is controllable, allowing us to test various configurations. Finally, as shown in Figure 1, the edges capture two forms of relationships **(a)** temporal across the same video and **(b)** semantic across the videos, enabling us to model inter-video and intra-videos context.

Difference from existing techniques: Most works explore procedure learning in a supervised [40,47,64] or weakly supervised [4, 6, 12, 24, 34, 35, 45, 46, 65] setting. Supervised methods require frame-level key-step annotations, making them unscalable [3]. On the other hand, weakly supervised learning methods require an ordered or unordered list of key-steps. Creating the lists requires viewing the videos or defining heuristics leading to scalability issues [15, 16]. Instead, the second stage of GPL enhances UnityGraph's node embeddings in an unsupervised manner using Node2Vec.

Similar works: The works closely related to ours employ various methods to create frame-level features to identify the procedure. Kukleva *et al.* [31] use the signal provided by the relative timestamp of the frame. Elhamifar *et al.* [15] discover and utilise the attention features from individual frames. Bansal *et al.* [3] solve the problem in a self-supervised manner by utilising the signal from corresponding frames among the videos of the same task. These works exploit different attributes of videos to extract the signal. However, they fall short in creating a graph-based representation to utilise the relationship between all the frames across the input videos. In this work, we propose UnityGraph, which first creates a clip-level representation and then captures the correspondences between the key-steps across videos. Our **major contributions** are:

(1) We propose the Graph-based Procedure Learning (GPL) framework. Contrary to existing graph-based frameworks, GPL does not require node or edge annotations, enabling unsupervised procedure learning.

(2) We create a novel graph representation for an arbitrary number of videos: UnityGraph. UnityGraph captures **(a)** temporal relationship in the same video and **(b)** semantic relationship across the videos.

(3) To identify the background frames, we propose to detect hand-object interactions in egocentric videos. This leads to an improvement of 1.1% on EgoProceL.

(4) We perform experiments on benchmark EgoProceL, ProceL, and CrossTask datasets and achieve on average 2% improvement on third-person datasets (ProceL and CrossTask) and 3.6% improvement on EgoProceL over the state-of-the-art. We will release the code for the work upon acceptance.

2. Related Works

Graph-based Representation for Video Understanding: Previous works have used graphs for action localisation [26, 59, 62], task completion [25], video super-resolution [60], grounding in instructional videos [23], question answering [54], and action recognition [7, 28, 37, 55]. Hussein *et al.* [26] utilise graphs to analyse human activity from a single video. G-TAD [59] proposes a graph where video clips are nodes, correlations between the nodes form edges, and actions associated with context are used to create target sub-graphs. Khan and Cuzzolin [28] propose to create a graph consisting of nodes based on the action tubes and edges encoding the relationships between the action tubes. However, generating action tubes is computationally expensive and not reliable [28]. Contrary to previous works, GPL proposes UnityGraph to discover key-steps across multiple videos and is the first to utilise graphs for procedure learning. Furthermore, UnityGraph is a task-level graph compared to the video-level graph in [59], enabling us to discover key-steps across videos. Finally, in contrast to the supervised setting in previous works, GPL learns embeddings in an unsupervised manner.

Representation Learning for Procedure Learning: Previous works on procedure learning have developed methods to learn frame-level features [15, 16, 31, 52]. Kukleva *et al.* [31] learn the representation space by using relative timestamps of the frames. On the other hand, Vidal *et al.* [52] predict the future frame and its timestamps. Elhamifar *et al.* [15] learn and employ attention features for individual frames. Bansal *et al.* [3] exploit temporal correspondences across the videos to generate the signal and learn frame-level embeddings. However, these methods fall short in modelling either temporal or spatial relationships. In contrast, GPL consists of UnityGraph that **(a)** represents videos at the clip-level and **(b)** forms edges between semantically similar and temporally close frames.

Multi-modal Procedure Learning: A sub-set of previous works utilises multi-modal data for procedure learning, for example, narrated text and videos [2, 9, 13, 18, 38, 48, 50, 61, 66]. These works have to assume an alignment between the videos and the text [2, 38, 61], which is inaccurate in most cases [15, 16]. Furthermore, they utilise an imperfect Auto-

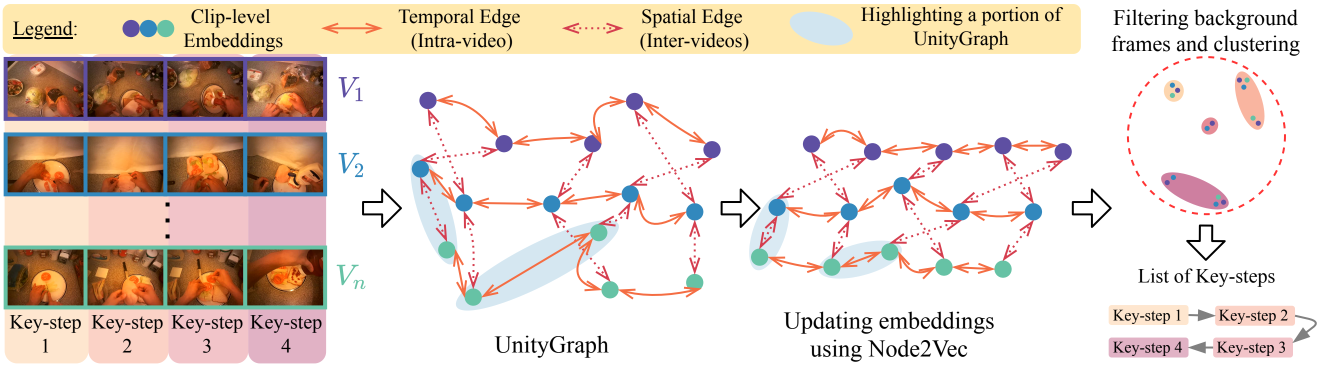

Legend:
Clip-level Embeddings
Temporal Edge (Intra-video)
Spatial Edge (Inter-videos)
Highlighting a portion of UnityGraph
Filtering background frames and clustering
V_1
V_2
V_n
Key-step 1
Key-step 2
Key-step 3
Key-step 4
UnityGraph
Updating embeddings using Node2Vec
List of Key-steps
Key-step 1
Key-step 2
Key-step 4
Key-step 3

Figure 2. **Graph-based Procedure Learning (GPL) framework.** Given multiple videos of the same task, we create UnityGraph. Using the Node2Vec algorithm, we exploit the structure of UnityGraph to enhance the node embeddings in an unsupervised manner. For example, the temporal and spatial clips that were originally far in the embedding space are closer after Node2Vec (highlighted in blue). Finally, we cluster the embeddings using KMeans and filter the background frames to obtain the key-steps required to perform the task.

matic Speech Recognition system to generate text, leading to the requirement of manually cleaning the text, and therefore, is unscalable. To overcome these issues, following [3], we employ only visual modality as our input to GPL, making our framework highly scalable.

Learning Key-step Ordering: Most of the previous works do not capture different key-step ordering to perform the task. They either assume strict ordering [16, 31, 52] or do not predict the order [15,50]. However, we observe that subjects perform the same task in multiple ways (Fig. 1), motivating us to capture different ways to accomplish a task. Therefore, GPL aims to create a key-step order for each video and infer the relevant ordering to perform the task.

Self-Supervised Representation Learning for Videos: Recent works have explored various pretext tasks for learning the representation in a self- or unsupervised manner. For example, utilizing temporal coherence and order as signals [17, 33, 39, 57, 58], predicting successive frames [1, 11, 21, 29, 51, 53] or identifying the arrow of time [56]. Video representation learning methods mentioned generate signals from a limited number of videos. However, our goal is to identify the key-steps from multiple videos. To this end, we explore graphs for procedure learning and propose GPL consisting of UnityGraph to improve video understanding.

3. Graph-based Procedure Learning (GPL)

Autonomously inferring the key-steps required to perform a task opens up the possibility of creating a variety of autonomous, guidance, and assistive systems. The majority of the previous works generate self-supervised signals from either single or a couple of videos. However, to better discover the procedure, getting the signal across all the videos is crucial. To this end, we utilise the capability of graphs to represent abstract video data. As shown in Figure 2, the first part of GPL framework consists of the proposed UnityGraph. It is a novel graph representation for an arbitrary number of videos of a task (Section 3.1).

The initial features of UnityGraph are created using a pre-trained I3D ResNet-50 [5, 22]. To further improve the features, as shown in Figure 2, the next step in GPL involves updating the embeddings using the Node2Vec algorithm in an unsupervised manner. Once the embeddings are learned, they are clustered using the KMeans algorithm (Section 3.2). The final step of GPL involves ordering the discovered clusters based on the average timestamps of the constituting frames (Section 3.2).

Notations: As shown in Figure 2, GPL takes in n untrimmed videos of the same task, denoted by $V = \{V_i : i \in \mathbb{N}, 1 \leq i \leq n\}$. Note that the n videos can have a different number of frames. A video V_x with m frames is divided into multiple clips using a sampling rate, stride, and window size of σ, ω, and ψ, respectively. The clips are then passed through a pre-trained I3D ResNet-50, denoted as f_θ (with parameters θ), used to generate node-level embeddings of dimension d for UnityGraph. The clips for video V_x with z clips are denoted as $V_x = \{c_x^1, c_x^2, \ldots, c_x^z\}$ and the video's node-level embeddings are denoted as $f_\theta(V_x) = \{v_x^1, v_x^2, \ldots, v_x^z\}$. Furthermore, we assume K key-steps in a task where K is a hyperparameter.

3.1. Representing Videos using UnityGraph

Assumptions: We make the following design choices when creating UnityGraph: **(a)** To compensate for the high frame rate and long action duration, we create UnityGraph's nodes at the clip level. **(b)** Using a 3D CNN, each clip is converted to an embedding. The motivation here is that the sampled clip either contains one action or none. **(c)** To keep the problem tractable, we assume the objects and actions are semantically similar across the task's videos.

3.1.1 Creating UnityGraph's nodes and edges

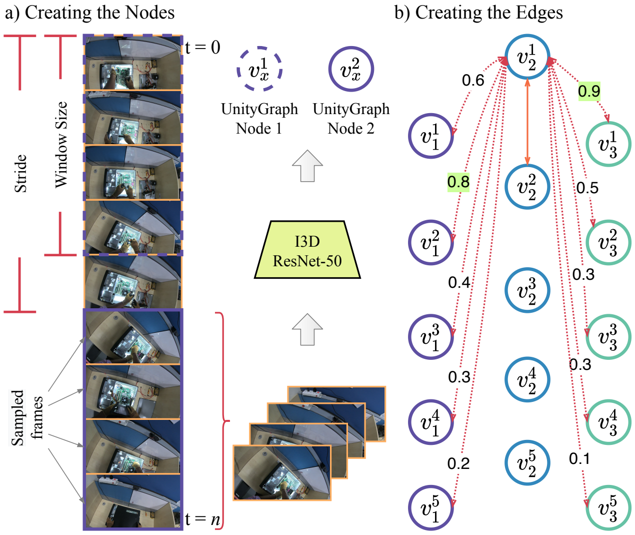


Figure 3. a) Given window size (ψ), stride (ω), and sampling rate (σ), a clip from a video is passed through a pre-trained I3D ResNet-50 to generate the node's embedding. b) We consider nodes from three videos (V_1, V_2, V_3). For brevity, we show similarity score between v_2^1 and all the nodes in V_1 and V_3. The edges with highest semantic similarity (marked in green) are retained.

Figure 3 summarises the creation of UnityGraph. A node v_x^1 in UnityGraph is the embedding of a clip c_x^1 for video V_x. The embedding is a d dimensional vector created using an I3D ResNet-50 [5, 22]. For example, for V_1 with $z = 5$, the nodes are created as:

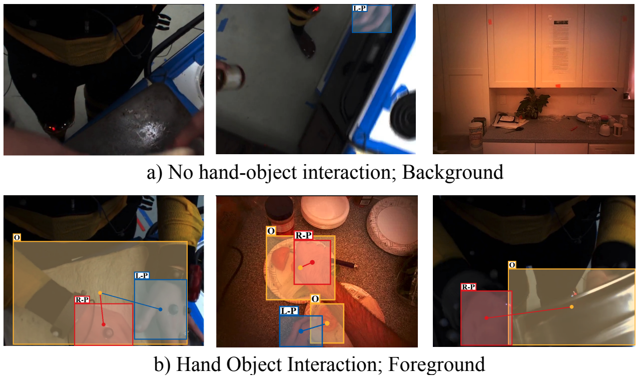
$$v_1^i = f_\theta(c_1^i), \text{where } i \in \{1, \ldots, z\} \tag{1}$$

Creating nodes in this way helps with **(a)** converting a volume of frames (the clip) to an embedding and **(b)** comparing and modifying the embeddings to understand the procedure. Figure 3 (a) summarises this process.

Once the nodes are created, the graph is completed by creating edges between them. The edges are created at two levels **(a)** spatial that facilitate inter-videos connection and **(b)** temporal that facilitate intra-video connection.

To better understand the process, consider two videos (V_1, V_2) from Figure 3 (b). Let us focus on creating an edge between the first node (v_2^1) from V_2 and nodes from V_1. The goal is to find the node in V_1 having the highest semantic similarity with v_2^1. To this end, we calculate the cosine similarity (S_C) between v_2^1 and all the nodes in V_1:

$$S_C(v_2^1, v_1^i) = \frac{\sum_{j=1}^{d} v_{2j}^1 v_{1j}^i}{\sqrt{\sum_{j=1}^{d}(v_{2j}^1)^2}\sqrt{\sum_{j=1}^{d}(v_{1j}^i)^2}}, \quad \text{where } i \in \{1, \ldots, z\}. \tag{2}$$


a) No hand-object interaction; Background

b) Hand Object Interaction; Foreground

Figure 4. We use the hand-object detection model from [49]. a) Frames not containing hand-object interaction. Second image in the first row contains a hand without an interaction with an object, hence, background. b) Frames containing hand-object interaction and contribute towards understanding the procedure.

The spatial edge is created between the node with the highest similarity score:

$$\text{Edge}(v_2^1, v_1^i) = \begin{cases} 1, \text{if} \max(S_C(v_2^1, v_1^i)) \\ 0, \text{otherwise} \end{cases}, \quad \text{where } i \in \{1, \ldots, z\}. \tag{3}$$

To create the temporal edges, we connect the neighboring nodes from the same video. Let us consider creating temporal edges for V_2:

$$\text{Edge}(v_2^i, v_2^j) = \begin{cases} 1, \text{if } |i - j| = 1 \\ 0, \text{otherwise} \end{cases}, \quad \text{where } i, j \in \{1, \ldots, z\}. \tag{4}$$

To summarise, UnityGraph consists of nodes created using Equation (1). The nodes are spatially connected using Equation (3) and temporally connected using Equation (4).

3.1.2 Detecting background frames

The procedure learning datasets majorly consists of background frames [3, 65], making it difficult to determine the procedure. We observe that a majority of the background frames involve people searching for objects, reading instructions, and waiting for an automated step to finish. Furthermore, as shown in Fig. 4, these activities do not involve hand-object interaction. Therefore, we argue that the frames lacking hand-object interactions represent the background and we propose to use Shan *et al.*'s hand-object interaction model [49] to filter out such frames in egocentric videos.

3.2. Identifying Key-steps and their Order

Updating and Clustering UnityGraph's Embeddings: As illustrated in Figure 2, using default embeddings ob-

Table 1. **Hyper-parameter values** for different components of the GPL framework. Here, "**Ego**" refers to first-person, "**ThP**" refers to third-person, and "Ablation table" refers to the table containing quantitative results for the respective hyper-parameter

Hyper-parameter	Notation	Value (**Ego**)	Value (**ThP**)	Ablation Table
Sampling Rate	σ	8	4	Table 4
Stride	ω	5	10	Table 4
UnityGraph's Window Size	ψ	10	10	Table 4
Walk Count	α	100	50	Table 5
Walk Length	γ	100	50	Table 5
Node2Vec's Window Size	β	12	10	Table 5
Return Parameter	p	1.0	1.0	Table 6
In-out Parameter	q	1.0	1.0	Table 6
No. of Key-steps	K	7	7	Table 7
No. of Videos	n	$\max(n)$	$\max(n)$	Table 9
Embedding Dimension	d	400	400	–

tained from the pre-trained network can result in embeddings lying far from each other. To improve the embeddings in an unsupervised manner, we update them using the Node2Vec algorithm [19]. The updated embeddings are clustered using KMeans to discover the key-steps.

Utility of Node2Vec: Node2Vec [19] learns embeddings while preserving neighborhood information by simulating biased random walks. UnityGraph connects semantically similar frames from multiple videos and temporally close frames from the same video. Node2Vec exploits these connections (refer to Figure 1 in supplementary) in an unsupervised manner to improve the embeddings. In contrast, DeepWalk [43] utilises uniform random walks and falls short of capturing this structure [19].

Identifying the Order of Key-steps: Once we have the clusters of key-steps, we follow [3] to determine their order. For each clip, the normalized time is calculated [3, 31]. Based on the cluster clips' normalized time, the average time for the cluster is calculated. The clusters are then arranged in an increasing order of time to generate the order of key-steps. This approach has two advantages **(a)** it allows each video to have its own key-step order and **(b)** does not require providing key-step ordering information.

Complexity Analysis: We extract information simultaneously from multiple offline videos and do not optimise for time. Hence, GPL's time complexity is exponential in the number of videos. Considering we have n videos of the same length (l), then the time complexity is $O(n^2l^2)$.

Table 2. **Procedure Learning from Third-person Videos**. Comparison between state-of-the-art methods and GPL on third-person datasets [16, 65]. Results in **bold** and underline are the highest and second highest in a column, respectively. **P**, **R**, and **F** represent precision, recall, and F-score, respectively

	ProceL [16]			CrossTask [65]		
	P	**R**	**F**	**P**	**R**	**F**
Uniform	12.4	9.4	10.3	8.7	9.8	9.0
Alayrc *et al.* [2]	12.3	3.7	5.5	6.8	3.4	4.5
Kukleva *et al.* [31]	11.7	30.2	16.4	9.8	35.9	15.3
Elhamifar *et al.* [15]	9.5	26.7	14.0	10.1	**41.6**	16.3
Fried *et al.* [18]	–	–	–	–	28.8	–
Shen *et al.* [50]	16.5	**31.8**	21.1	15.2	35.5	21.0
CnC [3]	20.7	22.6	21.6	22.8	22.5	22.6
GPL-2D (*ours*)	21.7	23.8	22.7	24.1	23.6	23.8
UG-I3D (*ours*)	21.3	23.0	22.1	23.4	23.0	23.2
GPL (*ours*)	**22.4**	24.5	**23.4**	**24.9**	24.1	**24.5**

4. Dataset and Evaluation Methodology

Evaluation: Unless otherwise mentioned, we evaluate the proposed GPL framework following [3]. We take the mean of the scores over all the key-steps and report F1 and IoU Scores. F1-Score is the harmonic mean of the precision and recall scores. For precision, we calculate the ratio between the number of frames having correct key-steps prediction and the number of frames assigned to the key-steps. For recall, the denominator is the number of ground truth key-step frames across all the key-steps of the video. Following [2, 3, 15, 16, 31, 50], we obtain the one-to-one mapping between the ground truth and prediction using the Hungarian algorithm [30].

Experimental Setup: We use features from the final layer of 3D ResNet-50 [22] pre-trained on Kinetics 400 [5] provided by PyTorch [42]. To keep feature extraction tractable, we reshape the short side of the video frame to 256, while maintaining the aspect ratio. For detecting the hand-object interactions, we use the 'handobj_100K+ego' model provided by [49]. We create and manipulate graphs using NetworkX [20]. Furthermore, Table 1 contains the hyper-parameter values obtained for egocentric and third-person view after an extensive ablation study.

Baselines: (a) Random: Here, the labels are obtained by randomly sampling predictions from a uniform distribution with K values representing K key-steps. **(b) CnC** [3]: This work generates frame-level embeddings by learning an embedding space that exploits temporal correspondences across a couple of videos. **(c) GPL-2D**: Here, to compare between clip- and frame-level features, we create UnityGraph nodes utilizing features from ResNet-50 initialised on ImageNet and further use Node2Vec to update the features. **(d) UG-I3D**: Here, we do not update the embeddings using Node2Vec. Instead, **U**nity**G**raph consisting of nodes embeddings from I3D ResNet-50 [5, 22].

Datasets: Contrary to previous works that use either first- or third-person datasets for procedure learning, we perform experiments on both views. For third-person procedure learning, we choose standard benchmark datasets,

Table 3. **Results on egocentric view** on EgoProceL, GPL outperforms previous work on most of the tasks. This highlights the effectiveness of video representation generated using the proposed UnityGraph and Node2Vec for updating the embeddings based on node neighborhoods. Note that EgoProceL is a recent dataset for egocentric procedure learning, due to this, there is only one approach (CnC [3]) to fairly compare with. Furthermore, as other methods have been specifically designed around third-person datasets, we compare with them on those datasets in Table 2. Here, CnC and GPL-2D (with Node2Vec) utilize features from ResNet-50 initialised on ImageNet whereas, UG-I3D (without Node2Vec) and GPL (with Node2Vec) utilize features from I3D ResNet-50 initialised on Kinetics-400. Results in **bold** and underline are the highest and second highest in a column, respectively

	EgoProceL											
	CMU-MMAC		EGTEA G.		MECCANO		EPIC-Tents		PC Assembly		PC Disassembly	
	F1	IoU	F1	IoU	F1	IoU	F1	IoU	F1	IoU	F1	IoU
Random	15.7	5.9	15.3	4.6	13.4	5.3	14.1	6.5	15.1	7.2	15.3	7.1
CnC [3]	22.7	11.1	21.7	9.5	18.1	7.8	17.2	8.3	25.1	12.8	27.0	14.8
GPL-2D (*ours*)	21.8	11.7	23.6	14.3	18.0	8.4	17.4	8.5	24.0	12.6	**27.4**	**15.9**
UG-I3D (*ours*)	28.4	15.6	25.3	14.7	18.3	8.0	16.8	8.2	22.0	11.7	24.2	13.8
GPL (*ours*)	**31.7**	**17.9**	**27.1**	**16.0**	**20.7**	**10.0**	**19.8**	**9.1**	**27.5**	**15.2**	26.7	15.2

CrossTask [65] and ProceL [16]. CrossTask consists of 213 hours of videos from 18 primary tasks (2763 videos). ProceL consists of 47.3 hours of videos from 12 diverse tasks (720 videos). To demonstrate the efficiency of the proposed GPL framework, we evaluate it in the first-person EgoProceL [3] dataset. It consists of 62 hours of egocentric videos of 130 subjects performing 16 tasks. Please note that we *do not* alter the key-step ordering provided by these datasets. Hence, they are identical to the previous works [2, 3, 15, 18, 31, 50].

5. Experiments and results

Results on Third-person View: Table 2 compares state-of-the-art methods and GPL on two third-person datasets [16, 65]. We obtain the results for the previous works from [3, 50]. Here, for a fair comparison, the framework is evaluated using the metrics laid out in [15, 31, 50]. We observe that [15, 31] assigns majority of the frames to one key-step. Due to this, the recall is high, however, the precision decreases, lowering the F-score significantly.

Results on Egocentric View: Table 3 compares state-of-the-art and GPL on the EgoProceL dataset. The results for tasks in CMU-MMAC [10] and EGTEA G. [36] have been averaged and reported. Note that EgoProceL is a recent dataset for egocentric procedure learning, hence, there is only one approach [3] to fairly compare with. We compare methods developed on third-person datasets in Table 2.

The results obtained highlight **(a)** generalisation capabilities of GPL. As GPL obtains high results on tasks using objects with high variability in size, different locations, and a variety of lighting conditions. **(b)** Effectiveness of using UnityGraph for modeling temporal and spatial relationships across the videos by creating a single representation for an arbitrary number of videos. **(c)** The results consistently increase upon using Node2Vec for updating the embeddings of UG-I3D. This further justifies our hypothesis, shown in Figure 2, that Node2Vec improves the embeddings and helps in inferring the procedure. **(d)** Utility of using clips for creating UnityGraph. GPL-2D performs comparable to previous frame-level techniques (*e.g.* [3]). However, as clips allow averaging the frame-level noise, collating information from clips performs the best.

Qualitative Results: Figure 5 shows the qualitative results obtained using the baselines and the proposed GPL framework on two tasks from EgoProceL.

6. Ablation Study

To determine optimal hyper-parameters, unless otherwise mentioned, we perform ablation on two challenging tasks: PC Assembly [3] and Change Tire [16]. Due to different attributes of first- and third-person videos, we select one set of hyper-parameters for each of the views. Finally, unless otherwise mentioned, for egocentric videos, UnityGraph is created using background frame detection.

Creating UnityGraph: In Table 4, we check values for frame sampling rate (σ), stride (ω), and window size (ψ). As egocentric videos have high motion variability, the maximum scores are obtained for a high sampling rate of 8. Also, the stride (5) and window size (10) are lowest for egocentric videos enabling the creation of nodes with high variability in less time. For third-person videos, the maximum scores are obtained for a low sampling rate of 4. This is because third-person cameras are fixed and do not have high variability in scenes. Furthermore, the stride of 10 and window size of 10 works best as it allows to sample sparsely.

Learning and clustering the embeddings: In Table 5, we explore various values for Walk Count (α), Walk Length (γ), and Window Size (β). For egocentric videos, we achieve best results for α as 100, γ as 100, and β as 12.

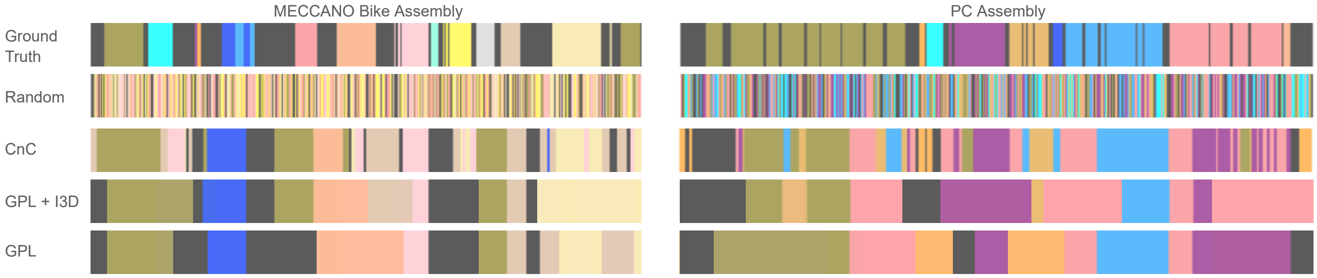


Figure 5. **Qualitative Results** for one video each of Bike and PC Assembly. Each color for a task denotes one key-step and gray sections are the background. First row contains the ground truth label, second row contains the results obtained by randomly predicting the key-steps, third row shows results obtained using CnC [3], fourth row highlight the results using UnityGraph's node generated using I3D ResNet-50, and the last row shows results obtained for the GPL framework. As can be seen, the segments obtained from GPL are more coherent upon using Node2Vec to update UnityGraph's embeddings. This highlights the efficacy of both, UnityGraph and Node2Vec

Table 4. **Hyper-parameters for creating UnityGraph**. Here, the results are obtained upon changing various parameters for creating UnityGraph. **R**, and **F** represent recall, and F-score, respectively

Sampling Rate	Stride	Window Size	PC Assembly			Change Tire		
			R	**F**	**IoU**	**R**	**F**	**IoU**
4	5	10	23.0	22.6	12.5	23.1	20.7	12.6
8	5	10	**28.8**	**27.2**	**15.1**	23.8	21.0	12.8
4	10	10	20.0	19.9	11.0	**26.2**	**23.2**	**13.9**
8	10	10	28.1	26.8	15.0	23.7	21.1	12.7
4	15	10	25.0	24.3	13.5	23.8	21.4	13.1
8	15	10	25.7	26.6	14.3	23.2	21.0	12.9
4	5	15	22.2	21.7	11.9	24.1	21.0	12.4
8	5	15	21.1	20.6	10.7	21.2	19.6	11.9
4	10	15	25.2	24.0	13.2	23.9	21.6	13.2
8	10	15	23.4	22.3	12.3	25.1	22.8	13.6
4	15	15	23.7	22.9	12.8	22.7	20.1	12.2
8	15	15	22.7	21.8	11.9	24.2	21.8	13.3

Table 5. **Hyper-parameters for learning the embeddings**. Here, the results are obtained upon varying Node2Vec's parameters. **R**, and **F** represent recall, and F-score, respectively

Walk Count	Walk Length	Window Size	PC Assembly			Change Tire		
			R	**F**	**IoU**	**R**	**F**	**IoU**
50	50	8	28.7	27.2	15.1	20.6	19.2	11.6
100	50	8	23.0	22.3	12.1	20.6	19.2	11.6
50	100	8	28.3	26.6	14.7	23.8	21.1	12.9
100	100	8	28.1	26.1	14.3	25.1	21.9	13.1
50	150	8	28.2	26.6	14.7	23.9	21.1	12.9
100	150	8	28.0	26.0	14.3	25.0	21.7	13.0
50	50	10	28.8	27.3	15.1	**26.1**	**22.6**	**13.4**
100	50	10	27.9	26.3	14.5	24.3	21.9	13.2
50	100	10	28.8	27.2	15.1	23.8	21.0	12.8
100	100	10	23.1	22.5	12.1	24.4	21.7	13.2
50	150	10	23.0	22.4	12.0	24.4	22.0	13.3
100	150	10	27.7	26.2	14.4	23.9	21.1	12.9
50	50	12	28.9	27.5	15.2	24.8	21.7	12.9
100	50	12	24.6	23.4	12.5	24.4	21.8	13.2
50	100	12	27.8	26.3	14.3	24.4	21.1	12.7
100	100	12	**29.0**	**27.5**	**15.2**	25.6	21.9	13.1
50	150	12	28.7	27.1	14.8	24.4	21.1	12.7
100	150	12	28.8	27.3	15.0	24.6	21.8	13.2

Table 6. **Hyper-parameters for walks over UnityGraph**. Here, we perform multiple walks to analyse the hyper-parameters for Node2Vec. **R**, and **F** represent recall, and F-score, respectively

Return Parameter	In-out Parameter	PC Assembly			Change Tire		
		R	**F**	**IoU**	**R**	**F**	**IoU**
0.1	0.5	28.7	26.9	14.8	24.5	21.8	13.2
0.1	1.0	28.4	26.7	14.7	24.5	21.8	13.2
0.5	0.1	24.2	23.0	12.3	20.5	18.9	11.4
0.5	1.0	28.5	26.8	14.7	20.6	19.0	11.5
1.0	0.1	24.6	24.6	14.5	**25.6**	**21.9**	13.0
1.0	0.5	28.8	27.3	15.1	20.4	18.9	11.4
1.0	1.0	**29.0**	**27.5**	**15.2**	**25.6**	**21.9**	**13.1**

Due to the high variability of scenes in egocentric videos, the walk count required to update the embeddings are high. This also leads to having a high walk length and a high number of frames in a window. Instead, for third-person videos, we require comparatively less walk count (50), walk length (50), and window size (10). As the majority of the videos in third-person datasets are from the internet, they skip a large number of repetitive portions of the task [3]. Due to this, the number of walks and the length are less. Furthermore, these datasets contain multiple non-relevant frames (explanation/animation) that should be circumvented.

In Table 6, we analyses the return parameter that controls the likelihood of immediately revisiting a node in the walk, and in-out parameter that allows the search to differentiate between inward and outward nodes [19]. For both views, we obtain the highest results for p and q as 1.0.

Number of key-steps and background frames: Table 7 contains results obtained upon varying K (number of key-steps) for the GPL framework. The results follow the trend in the previous work [3] and are highest for $K = 7$. The results decrease significantly as the values of K increase.

In Section 3.1.2, we discussed the presence of a high number of background frames in procedure learning datasets [65]. To address this issue, we employ a

Table 7. **Tuning** K. Here, the results are obtained for various values of K. **R**, and **F** represent recall, and F-score, respectively

K	PC Assembly			Change Tire		
	R	**F**	**IoU**	**R**	**F**	**IoU**
7	**29.0**	**27.5**	**15.2**	**25.6**	**21.9**	**13.1**
10	19.5	20.4	10.4	17.6	16.9	10.2
12	18.8	20.5	10.2	14.4	14.2	8.5
15	19.7	22.5	10.7	13.5	13.6	7.4

Table 8. **Detecting background frames**. Results are obtained upon filtering frames that do not contain hand-object interaction. Results improve for categories with subjects working in an unrestricted space. **R**, and **F** represent recall, and F-score, respectively

Hand-Object Interaction	PC Assembly			Greek Salad		
	R	**F**	**IoU**	**R**	**F**	**IoU**
Not Checked	**29.5**	**27.6**	14.4	25.4	22.3	12.7
Checked	29.0	27.5	**15.2**	**34.9**	**26.5**	**21.4**

hand-object interaction detection method on first-person videos. Frames that exhibit hand-object interaction are categorized as foreground, while the rest are considered background. The effectiveness of background frame filtration is demonstrated in Table 8, showcasing improved results for categories involving open spaces. For instance, in the Greek Salad dataset [36], the subjects work in an unrestricted kitchen environment, as opposed to PC Assembly [3], where they work in a confined space with hands being visible for the majority of the time. It should be noted that hand-object detection in third-person videos fails due to small hands and constant hand visibility. Hence, we exclusively utilize hand-object interaction for background filtering in egocentric videos. The enhanced results for third-person videos are solely attributed to GPL. Results for other categories in EgoProceL are in the supplementary materials.

Number of videos for creating UnityGraph: Table 9 contains results obtained by increasing the number of videos used to create UnityGraph. The objective here is to assess the effectiveness of the GPL framework in relation to the number of videos utilized. Here, we specifically select tasks with a number of videos that are powers of two. We then generate $\frac{n}{\text{Video Count}}$ graphs and concatenate the results to maintain consistency with the other experiments. As shown in Table 9, the highest F-score and IoU are achieved when using the maximum number of videos. This supports our main claim that employing UnityGraph to create a unified representation for all task videos enables capturing both the temporal relationships within individual videos and the semantic relationships across multiple videos. Furthermore, as the dataset size increases, GPL, in conjunction with UnityGraph, consistently achieves high-performance results.

Table 9. **Number of Videos**. Here, the results are obtained upon systematically increasing the number of videos for creating UnityGraph. **R**, and **F** represent recall, and F-score, respectively

Video Count	Bacon and Eggs [36]			Tie-Tie [16]		
	R	**F**	**IoU**	**R**	**F**	**IoU**
4	23.6	20.0	11.4	20.9	18.4	11.2
8	25.0	22.1	12.1	21.3	18.8	11.2
16	**27.8**	**23.1**	**12.6**	20.1	17.6	10.7
32	–	–	–	20.2	18.0	10.8
64	–	–	–	**23.5**	**19.7**	**11.4**

Remark on hyper-parameter selection: Determining the optimal values poses a challenge due to the unsupervised nature of the problem. To address this, we conduct experiments using both first- and third-person views and perform an extensive ablation study. We present a set of hyper-parameters for each view in Table 1. In an effort to achieve generalizable hyper-parameter tuning, we perform the ablation study on a single task from each view. It is worth noting that the EgoProceL dataset [3] contains videos from multiple sources, resulting in significant domain variation. Nonetheless, the superior performance of GPL over existing approaches on three datasets demonstrates that the chosen hyper-parameters are applicable across different datasets.

Assumptions and Limitations: Proposed GPL framework exhibits limitations that stem from certain assumptions. Firstly, UnityGraph relies on subjects using similar objects for identical key-steps, which might lead to inaccuracies when dissimilar objects are employed. Additionally, while our method filters background frames based on hand-object interactions, cases like the 'check booklet' from MECCANO challenge this assumption. We're actively working to address these limitations in future iterations of our work to enhance UnityGraph's robustness and accuracy.

7. Conclusion

Procedure learning is an important direction toward creating systems capable of assisting humans. Contrary to current approaches, we propose the graph-based procedure learning (GPL) framework. GPL consists of UnityGraph that creates a unified representation for multiple videos of the same task. UnityGraph allows us to model both, temporal and spatial information. The results obtained and the ablation performed demonstrate the capability of a graph-based approach for procedure learning.

Acknowledgments: The work was supported in part by the Department of Science and Technology, Government of India, under DST/ICPS/Data-Science project ID T-138. The authors thank Makarand Tapaswi and Charu Sharma for their Topics in Deep Learning course which motivated the paper's central idea.

References

[1] U. Ahsan, Chen Sun, and Irfan Essa. DiscrimNet: Semi-Supervised Action Recognition from Videos using Generative Adversarial Networks. In *Computer Vision and Pattern Recognition Workshops (CVPRW) 'Women in Computer Vision (WiCV)'*, 2018. 3

[2] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Ivan Laptev, Josef Sivic, and Simon Lacoste-Julien. Unsupervised learning from Narrated Instruction Videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 5, 6

[3] Siddhant Bansal, Chetan Arora, and C.V. Jawahar. My View is the Best View: Procedure Learning from Egocentric Videos. In *European Conference on Computer Vision (ECCV)*, 2022. 1, 2, 3, 4, 5, 6, 7, 8

[4] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly Supervised Action Labeling in Videos under Ordering Constraints. In *European Conference on Computer Vision (ECCV)*, 2014. 2

[5] João Carreira and Andrew Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3, 4, 5

[6] Chien-Yi Chang, De-An Huang, Yanan Sui, Li Fei-Fei, and Juan Carlos Niebles. D3TW: Discriminative Differentiable Dynamic Time Warping for Weakly Supervised Action Alignment and Segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[7] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[8] Richard W. Conners and Charles A. Harlow. A Theoretical Comparison of Texture Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1980. 1

[9] Dima Damen, Teesid Leelasawassuk, Osian Haines, Andrew Calway, and Walterio Mayol-Cuevas. You-Do, I-Learn: Discovering Task Relevant Objects and their Modes of Interaction from Multi-User Egocentric Video. In *British Machine Vision Conference (BMVC)*, 2014. 2

[10] F. De La Torre, J. Hodgins, A. Bargteil, X. Martin, J. Macey, A. Collado, and P. Beltran. Guide to the Carnegie Mellon University Multimodal Activity (CMU-MMAC) database. In *Robotics Institute*, 2008. 6

[11] Ali Diba, Vivek Sharma, L. Gool, and R. Stiefelhagen. DynamoNet: Dynamic Action and Motion Network. In *International Conference on Computer Vision (ICCV)*, 2019. 3

[12] Li Ding and Chenliang Xu. Weakly-Supervised Action Segmentation with Iterative Soft Boundary Assignment. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[13] Hazel Doughty, Ivan Laptev, Walterio Mayol-Cuevas, and Dima Damen. Action Modifiers: Learning From Adverbs in Instructional Videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[14] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal Cycle-Consistency Learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[15] Ehsan Elhamifar and Dat Huynh. Self-supervised Multi-task Procedure Learning from Instructional Videos. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 3, 5, 6

[16] Ehsan Elhamifar and Zwe Naing. Unsupervised Procedure Learning via Joint Dynamic Summarization. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 3, 5, 6, 8

[17] Basura Fernando, Hakan Bilen, E. Gavves, and Stephen Gould. Self-Supervised Video Representation Learning with Odd-One-Out Networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[18] Daniel Fried, Jean-Baptiste Alayrac, P. Blunsom, Chris Dyer, S. Clark, and Aida Nematzadeh. Learning to Segment Actions from Observation and Narration. In *Association for Computational Linguistics (ACL)*, 2020. 2, 5, 6

[19] Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. 5, 7

[20] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, 2008. 5

[21] Tengda Han, Weidi Xie, and Andrew Zisserman. Video Representation Learning by Dense Predictive Coding. In *Workshop on Large Scale Holistic Video Understanding, ICCV*, 2019. 3

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3, 4, 5

[23] De-An Huang*, Shyamal Buch*, Lucio Dery, Animesh Garg, Li Fei-Fei, and Juan Carlos Niebles. Finding "It": Weakly-Supervised, Reference-Aware Visual Grounding in Instructional Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[24] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Connectionist Temporal Modeling for Weakly Supervised Action Labeling. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[25] De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural Task Graphs: Generalizing to Unseen Tasks From a Single Video Demonstration. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[26] Noureldien Hussein, Efstratios Gavves, and Arnold W. M. Smeulders. VideoGraph: Recognizing Minutes-Long Human Activities in Videos. *ArXiv*, abs/1905.05143, 2019. 2

[27] Lei Ji, Chenfei Wu, Daisy Zhou, Kun Yan, Edward Cui, Xilin Chen, and Nan Duan. Learning Temporal Video Procedure Segmentation from an Automatically Collected Large Dataset. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022. 2

[28] Salman Khan and Fabio Cuzzolin. Spatiotemporal Deformable Models for Long-Term Complex Activity Detection. In *British Machine Vision Conference (BMVC)*, 2021. 2

[29] Dahun Kim, Donghyeon Cho, and In-So Kweon. Self-Supervised Video Representation Learning with Space-Time Cubic Puzzles. In *AAAI Conference on Artificial Intelligence*, 2019. 3

[30] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 1955. 5

[31] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. Unsupervised learning of action classes with continuous temporal embedding. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 3, 5, 6

[32] Sateesh Kumar, Sanjay Haresh, Awais Ahmed, Andrey Konin, M. Zeeshan Zia, and Quoc-Huy Tran. Unsupervised Action Segmentation by Joint Representation Learning and Online Clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[33] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Unsupervised Representation Learning by Sorting Sequences. In *International Conference on Computer Vision (ICCV)*, 2017. 3

[34] Jun Li, Peng Lei, and Sinisa Todorovic. Weakly Supervised Energy-Based Learning for Action Segmentation. In *International Conference on Computer Vision (ICCV)*, 2019. 2

[35] Jun Li and Sinisa Todorovic. Set-Constrained Viterbi for Set-Supervised Action Segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[36] Yin Li, Miao Liu, and James M. Rehg. In the Eye of Beholder: Joint Learning of Gaze and Actions in First Person Video. In *European Conference on Computer Vision (ECCV)*, 2018. 6, 8

[37] Xingyu Liu, Joon-Young Lee, and Hailin Jin. Learning Video Representations From Correspondence Proposals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[38] J. Malmaud, Jonathan Huang, V. Rathod, Nick Johnston, Andrew Rabinovich, and K. Murphy. What's Cookin'? Interpreting Cooking Videos using Text, Speech and Vision. In *HLT-NAACL*, 2015. 2

[39] Ishan Misra, C. L. Zitnick, and M. Hebert. Shuffle and Learn: Unsupervised Learning Using Temporal Order Verification. In *European Conference on Computer Vision (ECCV)*, 2016. 3

[40] Zwe Naing and Ehsan Elhamifar. Procedure Completion by Learning from Partial Summaries. In *British Machine Vision Conference (BMVC)*, 2020. 2

[41] Medhini Narasimhan, Arsha Nagrani, Chen Sun, Michael Rubinstein, Trevor Darrell, Anna Rohrbach, and Cordelia Schmid. TL;DW? Summarizing Instructional Videos with Task Relevance and Cross-Modal Saliency. In *European Conference on Computer Vision (ECCV)*, 2022. 1

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Neural Information Processing Systems*, 2019. 5

[43] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014. 5

[44] Yicheng Qian, Weixin Luo, Dongze Lian, Xu Tang, Peilin Zhao, and Shenghua Gao. SVIP: Sequence VerIfication for Procedures in Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[45] Alexander Richard, Hilde Kuehne, and Juergen Gall. Action Sets: Weakly Supervised Action Segmentation Without Ordering Constraints. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[46] Alexander Richard, Hilde Kuehne, Ahsan Iqbal, and Juergen Gall. NeuralNetwork-Viterbi: A Framework for Weakly Supervised Video Learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[47] Fadime Sener and Angela Yao. Zero-Shot Anticipation for Instructional Activities. In *International Conference on Computer Vision (ICCV)*, 2019. 2

[48] Ozan Sener, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. Unsupervised Semantic Parsing of Video Collections. In *International Conference on Computer Vision (ICCV)*, 2015. 2

[49] Dandan Shan, Jiaqi Geng, Michelle Shu, and David Fouhey. Understanding human hands in contact at internet scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4, 5

[50] Yuhan Shen, Lu Wang, and Ehsan Elhamifar. Learning To Segment Actions From Visual and Language Instructions via Differentiable Weak Sequence Alignment. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 3, 5, 6

[51] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised Learning of Video Representations Using LSTMs. In *International Conference on Machine Learning (ICML)*, 2015. 3

[52] Rosaura G. VidalMata, Walter J. Scheirer, Anna Kukleva, David Cox, and Hilde Kuehne. Joint Visual-Temporal Embedding for Unsupervised Learning of Actions in Untrimmed Sequences. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021. 2, 3

[53] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating Videos with Scene Dynamics. In *Neural Information Processing Systems*, 2016. 3

[54] Shaojie Wang, Wentian Zhao, Ziyi Kou, Jing Shi, and Chenliang Xu. How to Make a BLT Sandwich? Learning VQA Towards Understanding Web Instructional Videos. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021. 2

[55] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[56] Donglai Wei, Joseph Lim, Andrew Zisserman, and William T Freeman. Learning and Using the Arrow of Time. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[57] Jin woo Choi, Gaurav Sharma, S. Schulter, and Jia-Bin Huang. Shuffle and Attend: Video Domain Adaptation. In *European Conference on Computer Vision (ECCV)*, 2020. 3

[58] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-Supervised Spatiotemporal Learning via Video Clip Order Prediction. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[59] Mengmeng Xu, Chen Zhao, David S. Rojas, Ali Thabet, and Bernard Ghanem. G-TAD: Sub-Graph Localization for Temporal Action Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[60] Chenyu You, Lianyi Han, Aosong Feng, Ruihan Zhao, Hui Tang, and Wei Fan. MEGAN: Memory Enhanced Graph Attention Network for Space-Time Video Super-Resolution. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022. 2

[61] Shoou-I Yu, Lu Jiang, and Alexander Hauptmann. Instructional Videos for Unsupervised Harvesting and Learning of Action Examples. In *ACM International Conference on Multimedia*, 2014. 2

[62] Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan. Graph Convolutional Networks for Temporal Action Localization. In *International Conference on Computer Vision (ICCV)*, 2019. 2

[63] He Zhao, Isma Hadji, Nikita Dvornik, Konstantinos G. Derpanis, Richard P. Wildes, and Allan D. Jepson. P3IV: Probabilistic Procedure Planning From Instructional Videos With Weak Supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[64] Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards Automatic Learning of Procedures From Web Instructional Videos. In *AAAI Conference on Artificial Intelligence*, 2018. 2

[65] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 4, 5, 6, 7

[66] D. Zhukov, J.-B. Alayrac, I. Laptev, and J. Sivic. Learning Actionness via Long-range Temporal Order Verification. In *European Conference on Computer Vision (ECCV)*, 2020. 2