STEPs: Self-Supervised Key Step Extraction and Localization from Unlabeled Procedural Videos

Anshul Shah[1] Benjamin Lundell[2] Harpreet Sawhney[2] Rama Chellappa[1]

[1] Johns Hopkins University [2] Microsoft Mixed Reality

{ashah95, rchella4}@jhu.edu {benjamin.lundell,harpreet.sawhney}@microsoft.com

In this Supplementary material, we provide additional empirical studies, analyses and details. We list below the key sections.

1. **Feature extractor** : Details (A.1) and results with additional modalities on Ikea dataset (A.2).
2. **Bootstrapping** : Alternate variants (B.1)
3. **Temporal encoder**: model details (C.1), Avoiding trivial solutions and other baselines (C.2).
4. **Temporal sampling augmentation** : approach (D.1), effect of varying video extent (D.2)
5. **Miscellaneous analyses:** How long to train (E.1), Effect of inter-video alignment (E.2), TC3I with multiple-cue training (E.3), Additional results on loss ablation (E.4)
6. **Key Step extraction:** Sampling variants (F.1), visualizations (F.3)
7. **Practicality of the approach** : Time and resource requirements (G)
8. **Use of > 2 modalities during training** (H)
9. **Additional details:** Datasets (I.1), Additional information on KSL baselines (I.2), Evaluation protocols (I.3), Metrics (I.4), Hyperparameters (Sec. I.5)
10. **Code**: Overall flow (J).
11. **Limitations and future work** (K)
12. **Negative Societal Impact** (L)

A. Feature extractor:

A.1. Details

Our approach allows us to use off-the-shelf feature extractors instead of finetuning them. This is especially important in a data-scarce and resource constrained setting. Note that unlike some prior works, we work with average pooled spatial features for quick training and lower storage requirements.

All results except Table 4 in the main paper used Resnet 50 as m_1 and RAFT/OF features as m_2.

Res50: Features are extracted from the conv5c layer of a ResNet-50 [10] backbone. We use ImageNet pretrained models following the prior works [6, 9, 15].

RAFT/OF: RAFT [20] is a model trained for extracting optical flow from a pair of images. We extract motion features from the pre-trained feature encoder (last recurrent update).

For experiments in Table 4:

Gaze: For sensor derived modalities, we consider the on-device hardware/software block as our feature extractor. For example, in the Meccano dataset, gaze data is made available as the x & y location of the gaze mapped to the image coordinates along with a confidence score. Since gaze is recorded at 200Hz compared to 12Hz for the visual stream, we associate each from of the video to 16 ($200/12$) frames thus giving us a $3 \times 16 = 48$ dimensional gaze vector corresponding to each frame.

Depth: Depth maps are encoded using a pretrained ResNet-50 and features extracted from the Conv5c layers are used for training.

A.2. Additional Results

Separating feature extractor step from the SSL lets us use any available off-the-shelf backbone. We next explore alternate feature extractor (Pose) for the IkeaASM dataset Table 1.

Pose: We extract human pose coordinates using OpenPose [3]. The pose coordinates of each joint are stacked and used as input to the temporal encoder. We also experimented with using features from a pre-trained pose-based action recognition model. Specifically, we used features extracted from a FineGym and NTU pretrained PoseC3D [5] model. We empirically found that this approach gave us a similar performance as using pose coordinates and thus we use coordinates for our pose-based experiments in the paper. We use the pose modality for experiments with the Ikea dataset due to the nature of videos (third person, static

Table 1. Comparison between different feature extractors for Ikea dataset. We use multi-cue features for training and evaluate on single/multi-cue features for inference.

Training	Inference	Phase CLS 1.0
Res50+Pose	Res50	30.6
Res50+Pose	Pose	30.3
Res50+Pose	Res50+RAFT	31.9
Res50+RAFT	Res50	31.5
Res50+RAFT	RAFT	30.3
Res50+RAFT	Res50+RAFT	31.7

camera). For the unconstrained videos of the other datasets, we make use of our RAFT-based feature extractor.

In Table 1 we compare the performance of different feature extractors for the task of phase classification on the Ikea dataset. During inference, for a fair comparison to the prior works we use the appearance feature alone unless specified otherwise. We also experiment with use of multiple modalities at inference (like Table 4 main paper) and use a simple concatenation of features (e.g. Res50 + RAFT). We see that use of multiple modalities at inference leads to additional gains.

B. Bootstrapping

B.1. Alternate variants

In the main paper, we discussed our proposed technique of improving the set of positives by using raw features to bootstrap the temporal windows. We noted (Section 3.3, main paper) that our final window used $\tilde{\mathcal{W}}$ was a union of the σ-window and the one obtained through bootstrapping. We refer to this approach as 'Union-Window'. In this section we show our results with some alternate variants that we tried.

Only the sampled window: In this approach we discard the σ-window for the loss computation and use only the bootstrapped window $\mathcal{W}'$.

Union Window, only modifying negative set : Here, the positives come from the σ-window but instead of using the complement as negatives, we also remove the potential false negatives obtained using $\tilde{\mathcal{W}}$.

In Table 2, we compare these three variants. We see that our approach of using the union window for both positive and negative set shows the best performance. Using the sampled window performs since it doesn't necessarily impose the temporal constraints for the loss function and ignore the σ-window. Use of the union window only to generate the negative set doesn't modify the false negatives as positives and shows poor performance.

Table 2. Comparing bootstrap variants on Meccano dataset. We compare variants for using the bootstrapped window. We notice that using the union window for defining both positives and negatives leads to the best results.

Variant	F-1	IoU
No bootstrap	32.1	14.9
Only sampled window	34.4	16.3
Union Window for negative	34.6	16.3
Union Window for positive and negative	**36.4**	**18.0**

C. Temporal Encoder

C.1. Model details

We train a separate temporal encoder per modality. Each temporal encoder is a multi-headed transformer model. The input to the temporal encoder are raw features $p_t^{m_i} \in \mathbb{R}^{D_i}$. We use a two layer vanilla transformer with two heads. We do not use causal masks. The temporally adapted sequence ($\tilde{q}_t^{m_i} \in \mathbb{R}^D$) is then passed through a two layer MLP to obtain features ($q_t^{m_i} \in \mathbb{R}^D$) used for computing the loss. D is set to 128.

C.2. Analysis : Positional Encoding

In Table 3, we analyse different ways of encoding temporal information into the model. Since transformers on its own does not have any information about the relative position of various steps in the temporal dimensions, we add a sinusoidal position encoding [21] to the input sequence before adapting it with the transformer. Since our model will be subsequently trained with losses that enforce temporal consistency, any learnable embedding before adding position encoding might learn to ignore the raw features (Table 3 *STEPs w/ MLP before PosEnc*). Hence, we add this embedding to the raw feature directly to avoid a collapse of the learned representations. Note that removing the positional encoding performs worse (*STEPs w/o PosEnc*) which shows the importance of encoding temporal information into the feature sequence. We see the same trend when using a loss with and without bootstrapping. Our final approach with bootstrapping outperforms other baselines. While this baseline still trains a temporal encoder, another potential baseline is to use the raw features directly for evaluation. This baseline (*Raw features*) performs much worse showing the importance of adapting the features through our training loss. .

D. Temporal Sampling augmentation

D.1. Sampling approach

Most works in Video-SSL work with short clips. We work with procedures which are minutes long, often with steps that might repeat in the future. Our approach trains a light-weight temporal encoder on pre-extracted features thus enabling use

Table 3. Comparing different variants of positional encoding on Meccano dataset. We see the benefits of using a positional encoding during training to impart ordering information. The positional encoding is added directly to raw features thus precluding the model from learning trivial embeddings. We see similar trends for model w & w/o the bootstrapping window.

Approach	F1	IoU
TC3I [1]	18.1	7.8
Raw Features	20.1	8.6
MC2 loss		
STEPs w/o PosEnc	27.7	12.7
STEPs w/ MLP before PosEnc	28.8	12.9
STEPs	34.2	16.6
BMC2 loss		
STEPs w/o PosEnc	32.2	14.7
STEPs w/ MLP before PosEnc	32.8	15.3
STEPs	**36.4**	**18.0**

of long-range dependencies. Note that since our approach works with features instead of raw videos/images, we cannot employ any input-space augmentations like random resize crop, color jitters, blurring etc during SSL training. We apply a temporal sampling augmentation but on features and do not generate multiple 'views' of the video. Instead, we extract positives and negatives from a single sample of the video. Our sampling strategy during training borrows ideas from TSN [22] but applies it to feature sequences. Given the complex video, we first randomly select a start t_{start} and end frame t_{end}. These are sampled based on a hyperparameter of how much temporal extent of the video we want to cover during training. Next, the selected extent is uniformly divided into N chunks. A feature frame from each chunk is randomly sampled which is then used in the subsequent model. Note that the corresponding time stamp of the sampled frame is utilized in positional encoding Appendix C.2. The flow is presented in Fig. 1.

D.2. Effect of temporal extent

Given a temporal extent β, t_{start} and t_{end} are sampled from $[1, \cdots, T]$ such that $t_{end} - t_{start} \simeq \beta T$ All experiments in the main paper use $\beta = 1$ which implies $t_{start} = 1$ and $t_{end} = T$. In Table 4, we experiment with different temporal extents and the effect on performance on the Meccano dataset. We note that $\beta = 1$ gives the best performance.

E. Miscellaneous Analyses

E.1. How long to train?

We train all Meccano models for 300 epochs (Appendix I.5). In this section we explore the effect of longer training of our models. We train our model for longer on the Meccano dataset. With a learning rate drop at 400 epochs, the model obtains F1 score of 37.8 and IoU of 18.25 at 500 epochs which shows that the model gradually improves even beyond 300 epochs but the marginal gains are low as is often observed in self supervised learning.

Table 4. Comparing temporal extents. We sample our features from varying temporal extents of the video. β denotes the extent of the whole video used for sampling. Using $\beta = 1$ shows the best performance.

Temporal Extent (β)	F-1	IoU
0.2	24.7	10.7
0.4	26.7	11.2
0.6	30.6	13.9
0.8	35.1	17.0
1.0	**36.4**	**18.0**

E.2. Effect of inter-video alignment

Our approach shows strong performance compared to prior works without using any alignment-based loss. This lets us train even on a single video. An additional video-alignment objective is complementary to our contributions. Upon including inter-video alignment loss (soft-DTW [9]), we notice that the IoU improves by 0.7 with no improvement in F1 score.

E.3. TC3I with multiple modalities

Use of multiple modalities during training can benefit other approaches too. We experiment with including multi-cue training with the publicly available implementation of TC3I. Using RAFT (Optical flow) cues during training improves IoU on Meccano from 7.8 to 8.6. We notice that STEPs still faares much better since it can capture long range temporal dependencies and benefit from bootstrapping of raw features.

E.4. Additional results on loss ablation

In Table 5, we present the effect of various components of our approach on four egocentric datasets. We see that our proposed losses brings an improvement to the final performance in most cases.

F. Key Step extraction

F.1. Using alternate sampling approach for key step extraction

Separating representation learning from generation allows us to easily use alternate approaches to extract key steps. We experiment with the approach ILS-SUMM [18] which implements iterative local search for unsupervised video summarization. They recover an optimal summary by

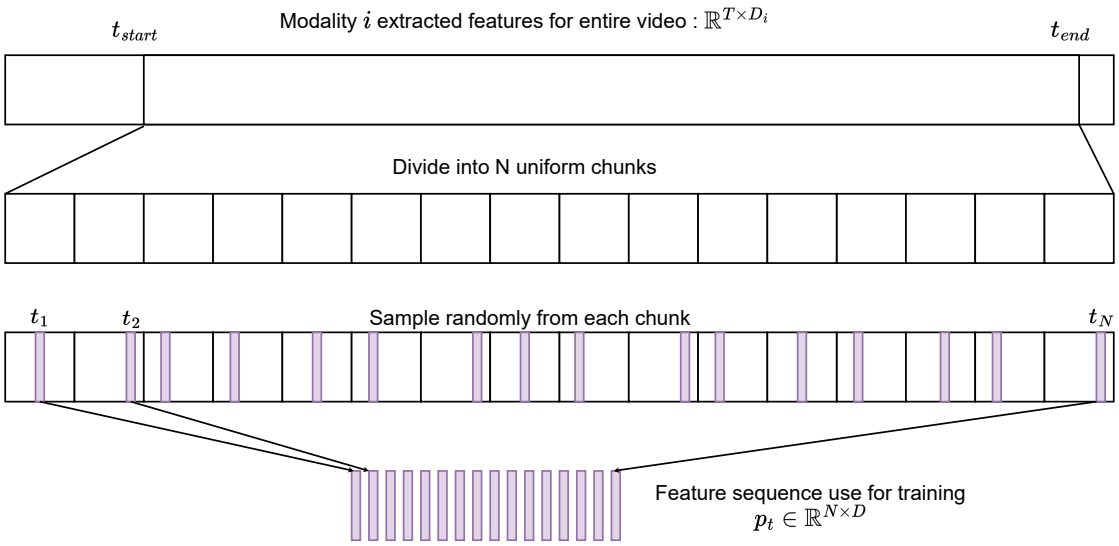

Figure 1. Temporal sampling. We illustrate the sampling applied during training. We first set a temporal extent. This is divided into N equal sized chunks. A feature frame is sampled at random from each chunk which is used to contruct the input sequence.

Table 5. Effect of different losses. In this experiment, we evaluate the effect of our different loss terms on the four egocentric datasets. We see that our loss terms leads competitive performance compared to the baseline.

Approach	Multi-Cue training	BMC2	*CMU-MMAC* F1	*CMU-MMAC* IoU	*EGTEA G.* F1	*EGTEA G.* IoU	*Meccano* F1	*Meccano* IoU	*EPIC-Tents* F1	*EPIC-Tents* IoU
Random	-	-	15.7	5.9	15.3	4.6	13.4	5.3	14.1	6.5
Uniform	-	-	18.4	6.1	20.1	6.6	16.2	6.7	16.2	7.9
Bansal et al. [1]			22.7	**11.1**	21.7	9.5	18.1	7.8	17.2	8.3
STEPs	✗	✗	26.2	10.7	29.5	12.2	29.9	14.1	37.1	20.0
STEPs	✗	✓	25.0	9.6	29.6	**12.4**	33.1	15.8	37.7	19.0
STEPs	✓	✗	27.7	11.0	**30.8**	**12.4**	32.0	15.3	41.2	**21.9**
STEPs	✓	✓	**28.3**	**11.4**	29.0	11.6	**36.4**	**18.0**	**42.2**	**21.4**

solving a constraint optimization problem on the total summary duration. The approach leads to plausible key steps (Fig. 2) which show the effectiveness of our learning algorithm. We use our proposed clustering & sampling steps since they provide more user control for key step extraction focused on task-based videos while ILS-SUMM provides fewer controls. For example, the k-Means clustering algorithm can easily be swapped with an alternative algorithm like FINCH [17] which can automatically detect the number of clusters. In Figure 3, we show results by swapping the k-Means in Algorithm 1 with FINCH. For an easy comparison with other presented results, we sort and visualize 10 key steps which have the least distance to the cluster centers. We see that use of this alternate clustering algorithm gives plausible key-steps. The example shows that our algorithm allows for easy customization based on user requirements.

F.2. Alternative KSE metrics

Prior works like [12] proposed metrics for evaluating video summarization systems. We explore the evaluation of representation and diversity metrics from [12] for the extracted key-steps on Meccano dataset. We compare these scores to key steps extracted using TC3I's features for the same number of steps extracted. We notice that our approach leads to a better representation score (53.1 vs 48.2) and higher min-disparity (diversity) score (6.52 vs 4.54). Since we are dealing with procedural videos rather than generic home-videos, we believe that KSL is a more apt proxy for KSE. Following prior works, we evaluate on KSL metric in this paper.

F.3. Additional KS visualizations

In Fig. 4 and Fig. 5, we visualize the extracted key steps for a video of Meccano and EPIC-Tents respectively. We observe that the extracted key steps are plausible for the task of assembling a toy bike and assembling a tent respectively.

Figure 2. Alternative sampling approaches. We use our extracted features with ILS-SUMM to extract a 10 step summary of a person assembling a Kallax Shelf Drawer. The approach leads to plausible key steps showing the efficacy of our features. We use our proposed clustering & sampling steps since they provide much more user control for key step extraction focused on task-based videos.

Figure 3. Alternative clustering algorithm. We replace k-Means with FINCH [17] and visualize the top-10 key steps based on distance to the cluster centers. While this clustering approach seems to miss a step, we see that the results are still plausible. Our algorithm allows for easy customization based on user requirements.

G. Practicality of the approach

The use of off-the-shelf feature extraction without fine-tuning enables fast transformer encoder training. We can train a model on 17 Meccano videos for 300 epochs in under 6 minutes on a single Nvidia A5000 GPU. While we use a 24GB GPU, for our set of hyperparameters, the training run needs only $\sim$ 3.5GB GPU memory. We use a single GPU for all our experiments. Our small memory footprint (1.8M parameters for our model compared to 23M in the Res50 encoder alone) along with the use of off-the-shelf features allows training with large temporal extents unlike approaches which rely on finetuning. We note that precomputing of features is also quite fast. We use standard pipelines for feature extraction. For example, using a single GPU, it takes less than 15 minutes to extract Resnet50 appearance features for *all* frames ($> 300k$) of Meccano dataset. Further, some cues like gaze do not need any feature extraction and can be

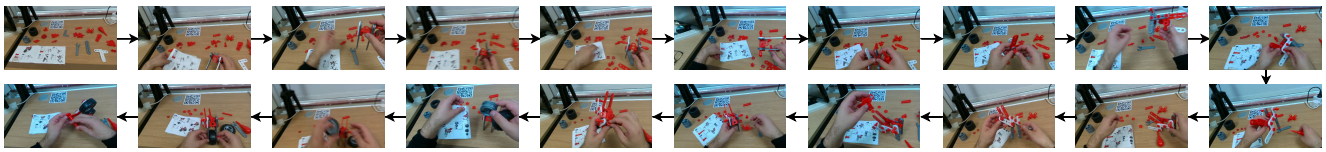

Figure 4. KS Visualization for Assembing a Toy Bike (Meccano dataset). We use the features extracted using our approach and extract a 20 step summary of assembling a toy model of a motorbike. We use the cluster and sample approach to generate these. We notice that the keys steps are plausible.

Figure 5. KS Visualization for Assembing a tent (EPIC-Tents dataset). We use the features extracted using our approach and extract a 20 step summary of assembling a tent. We use the cluster and sample approach to generate these. We notice that the keys steps are plausible.

Table 6. Training with $N > 2$ modalities

Method	Training	Inference	F1	IoU
Training using 3 modalities				
STEPs	RGB+OF+Gaze	RGB	37.8	18.7
STEPs	RGB+OF+Depth	RGB	38.7	19.0
STEPs	RGB+Gaze+Depth	RGB	37.8	18.7
Training using 4 modalities				
STEPs	RGB+OF+Gaze+Depth	RGB	**40.2**	**19.8**

used in the raw form.

H. Use of multiple modalities during training

In the main paper we use 2 modalities for all of our experiments. In this section, we train our approach using > 2 modalities while performing inference on the RGB/appearance modality alone for an easy comparison. We present our results in the Table 6. We observe that use of additional modalities benefits from longer training and we train these models for 700 epochs with a LR drop at 600. For simplicity, in these experiments, we set

$$\lambda_{uv} = \begin{Bmatrix} 1, & \text{for } u = v \\ 1, & \text{for } u = \texttt{RGB}, v \neq \texttt{RGB} \\ 0, & \text{otherwise} \end{Bmatrix}$$

We see that using additional modalities during training consistently improves performance. In particular, we note that depth and optical flow are especially helpful. Finally, using all four modalities during training gives further improvements and gives the best results.

I. Additional details:

I.1. Dataset details

I.1.1 Egocentric procedure learning datasets

We use four egocentric datasets to validate our approach. We use the publicly available annotations from the authors of CnC [1] for our experiments on these datasets.

CMU-Kitchens/MMAC [4]: This dataset contains recording of subjects performing the tasks involved in cooking and food preparation. A kitchen was built and twenty-five subjects were recorded to cook five different recipes: brownies, pizza, sandwich, salad, and scrambled eggs.

EGTEA Gaze+ [14]: Extended GTEA Gaze+ is a large-scale dataset with modalities like visual, gaze tracking, audio etc. It consists of activities from 86 unique sessions of 32 subjects. The dataset includes seven cooking recipes : Continental Breakfast, Pizza, Bacon and eggs, Cheese burger, Greek salad, Pasta salad and Turkey sandwich.

Meccano Dataset [16]: Meccano is a multimodal egocentric video dataset of people assembling a toy bike. The dataset is captured with multiple modalities like RGB videos, depth videos and gaze signals. We use this dataset for most ablations and analyses.

EPIC-Tents Dataset [11]: This is an outdoor egocentric video dataset of people assembling a camping tent. The dataset is collected from 29 participants.

I.1.2 Third-person procedure learning datasets

ProceL Dataset [8]: consists of videos from 12 procedures like replacing iPhone battery, setting up Chromecast etc. The dataset consists of about 720 videos with 8 key-steps on average. The videos are obtained from YouTube. We use the same experimental setup and features as used in [1] following their official implementation.

Crosstask Dataset [24]: is an instructional video dataset with 18 primary tasks. The dataset consists of 2750 videos with 7 keys-steps on average. We use the same experimental setup and features as used in [1] following their official implementation.

Ikea-Assembly Dataset [2]: comprises of complex furniture assembly tasks performed by multiple people across views. The challenging tasks are composed of multiple sub-tasks like flipping the table, attaching the leg etc. While captured in a controlled setting, the dataset consists of large temporal variations and includes sections where no activity takes place. This dataset closely mimics the practical AR/VR scenario that we discuss in the introduction of the main paper. We follow the same experimental setting as the prior work [9, 15] for a fair comparison. The top-view was used for all experiments. Kallax Shelf-Drawer assembly task split was used unless otherwise specified.

I.1.3 Other datasets

PennActions Dataset [23]: The dataset consists of 13 action categories from the PennAction dataset as used by authors in [6, 9, 15]. The actions are composed of humans doing sports and exercises and are composed of 2-6 phases per action. While this is not strictly a procedure learning datasets, we apply our method to this approach to show the wide applicability.

I.2. More information on KSL baselines

Random: Following [1], we assign a random label to each frame from a uniform distribution with K values. These K values represent K steps.

Uniform: Here, we uniformly divide the video into K chunks and assign a unique label to each chunk. We found this to be a stronger baseline than 'Random'.

Bansal et al [1]: Here we directly report the best results obtained by the most relevant prior work. This approach trains an embedding network using a combination of losses. The frames for evaluation videos are then assigned to K

labels using a ProCut module which relies on soft-clustering. We report their best results for each compared dataset.

I.3. Evaluation protocols

We follow the same evaluation protocols as prior works for a fair comparison. For CMU-MMAC, EGTEA-Gaze+, EPIC-Tents, Meccano, ProceL and CrossTask, we follow the evaluation protocol used in CnC [1]. Videos were sampled at FPS/2. We use the improved metric proposed in by authors of [1] which evaluates the IoU and F1 score per-key step and averages them. Closely following [1], this modified protocol is not used for CrossTask and ProceL datasets where standard protocol [7, 8, 13, 19] of calculating it for all steps together is used. For experiments on IkeaASM and PennActions, we closely follow the protocols laid out in VAVA [15] and LAV [9]. For IkeaASM, evaluation was performed on frames sampled at 8FPS, while for PennActions, they are sampled at original FPS during evaluation.

I.4. Metric details

Here we present additional details on the various metrics used to evaluate our models. For all metrics, a higher score implies a better performance.

Key-step localization: We follow the same experimental setup as [1]. We first find a one-to-one matching between steps in the ground truth and clustering predictions from our method using the Hungarian algorithm. Recall is computed as the ratio of number of frames having the correct key step prediction to the ground truth number of key frames across all key steps. Precision is the ratio of the number of correctly predicted frames and number of frames predicted as key steps. F-1 score is the harmonic mean of recall and precision.

Phase Classification: We calculate the average per-frame phase classification accuracy obtained by training an SVM on the phase labels on our temporally adapted per-frame features. Following prior works [6, 9, 15], we evaluate the model on varying amounts of labels used for SVM training (0.1, 0.5 and 1.0).

Kendall's Tau: This is a statistical measure which is used to determine the temporal alignment between two sequences. Since this metric assumes a strict monotonic order of actions and we report it only for the PennActions dataset. Given two videos, we first sample a pair of frame features q_i, q_j from the first video and retrieve the corresponding closest features in the second video of the same task $q_{i'}, q_{j'}$. The frame indices i, j, i', j' as concordant if $(i-j)(i'-j') > 0$ and discordant other wise. Kendall's Tau is calculated as

$$K.T = \frac{\#\text{concordant pairs} - \#\text{ discordant pairs}}{\binom{n}{2}} \quad (1)$$

Please refer to [6] for further details.

Table 7. Common hyperparameters used for training STEPs models

Hyperparameter	Value
Clustering Algorithm	k-Means
# Clusters	7
λ_{uu}	1
λ_{uv}	1
margin ζ	2.0
Learning rate	1E-3
Train modalities	Res50 + RAFT-OF
Inference modalities	Res50
Number of heads	2
Dimension of hidden layers	128
Number of transformer layers	2
Temporal extent β	1.0
Optimizer	Adam

I.5. Hyperparameter details

We use PyTorch for training our models. In Table 7, we list the common per-dataset hyperparameters used for our experiments in Tables 1, 2 & 3 of the main paper. Following are some additional training and implementation details.

Batch size: We use a batch size of 4 for all datasets except CrossTask, ProceL and PennActions which use a batch size of 16.

Temporal window size: We use a window size of 10s for all datasets except PennActions. Due to the small temporal lengths in that dataset, we instead use a window size of 4 frames for that dataset.

Epochs: We train models on Meccano and EGTEA for 300 epochs and on EPIC-tents and CMU-Kitchens for 150 epochs. Models on CrossTask and ProceL were trained for 100 epochs due to large size of the dataset. Models on Ikea were trained for 300 epochs while those for PennActions were trained for 400 epochs.

Modality for bootstrapping: We obtain the best results when we use RGB modality for bootstrapping on Meccano, EPIC, and ProceL datasets. Raw RAFT features were used for CrossTask while a concatenation of RGB and RAFT were used for CMU Kitchens, EGTEA, Ikea and PennActions.

Number of chunks: We use 1024 chunks for all datasets except CrossTask, ProceL and PennActions. Both CrossTask and ProceL use 512 chunks. Due to the very short lengths of videos in PennActions, we determine number of chunks as 0.8 of average video length in that dataset.

Average over runs: Since most of the datasets we work with are small in size, we report all results for Meccano, EPIC-Tent, CMU-Kitchens, EGTEA and Ikea as average of 3 training runs from random seeds.

Key step extraction details For our key step visualizations, we first cluster the video features. For k-Means, we set num-

Algorithm 1 Overall flow

Input: Video dataset V
Output: Adapted features $\tilde{q}$ and Key Steps a_k for each video

1. Extract and store raw features
$P_i = \texttt{FeatureExtractor}_i(V)$

# 2. Train STEPs model using pre-extracted raw features
for epoch $ep = 1, \ldots, L$ **do**
  # Temporally sample features and create a minibatch
  $p_1, \cdots, p_i = \texttt{TemporalSampling}(P_1, \cdots, P_i)$
  # Forward pass through per-modality temporal encoders
  $\tilde{q}_1, \cdots, \tilde{q}_i = \texttt{f}_1(p_1), \cdots, \texttt{f}_i(p_i)$
  # Project features using per-modality MLP and L2 normalize
  $q_1, \cdots, q_i = \texttt{Project}_1(\tilde{q}_1), \cdots, \texttt{Project}_i(\tilde{q}_i)$
  # Obtain bootstrap window using σ-window and raw features
  $\tilde{\mathcal{W}} = \texttt{BootstrapWindow}(p_i, \sigma)$
  # Calculate BMC2 loss and backpropagate
  $\texttt{loss} = \texttt{BMC2}(q_1, \cdots, q_i, \tilde{\mathcal{W}})$
end for

3. Evaluate for downstream task
Extract Key Steps $\{a_k\}$ for video v using learned temporal encoder f_i
$\{a_k\} = \texttt{KeySteps}(v, f_i)$
or evaluate for Key Step Localization on dataset V using learned temporal encoder f_i
$\texttt{IoU}, \texttt{F1} = \texttt{KeyStepLocalization}(V, f_i)$
return $\{a_k\}_{k=1}^{k=K}, \tilde{q}$

ber of clusters as the number of phases in the task. We use a background rejection ratio of 0.1. Threshold time between steps (γ) is chosen as 2 seconds. A step is sampled from each sub-segment based on distance to the cluster center. For visualization we display top10/top20 key-steps based on the distance to center for each sampled key step. For KS extraction results on the Ikea dataset, we show the crop around the person instead of the whole frame for clarity. The models were trained and evaluated were *not* evaluated using the person-crop.

J. Overall flow

We illustrate the overall flow of our approach in Algorithm 1.

K. Limitations and Future work

Our approach is the first step towards Key Step extraction for AR/VR applications where many of the modalities are available for free through on device sensors/modules. While this is generally true, extracting and parsing these modalities requires pre-trained feature extractors and/or domain knowledge. To reduce storage requirements, we work with average pooled features which precludes the model from using spatial attention. Next, while our approach can work with even a few videos, the performance gap suggests that incorporating recent advances in few-shot learning, or use of highly contextualized embeddings like CLIP can help further improve the performance. Finally, Key steps for a task can be very subjective and vary based on application. When deploying these models, key steps might have to be validated via device usage experiments and the model appropriately tuned.

L. Negative Societal Impact

The paper proposes an approach to extract key steps for unlabeled procedural videos. As such, we do not perceive any negative ethical or societal impact since experiments were done on using publicly available datasets and models. That said, while deploying such models in the wild, consent of all individuals must be taken to avoid leaking any potentially sensitive information.

References

[1] Siddhant Bansal, Chetan Arora, and CV Jawahar. My view is the best view: Procedure learning from egocentric videos. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIII*, pages 657–675. Springer, 2022. 3, 4, 6, 7

[2] Yizhak Ben-Shabat, Xin Yu, et al. The ikea asm dataset: Understanding people assembling furniture through actions, objects and pose. In *IEEE/CVF WACV*, pages 847–859, 2021. 6

[3] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017. 1

[4] Fernando De la Torre, Jessica Hodgins, Adam Bargteil, Xavier Martin, Justin Macey, Alex Collado, and Pep Beltran. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. 2009. 6

[5] Haodong Duan, Yue Zhao, Kai Chen, Dahua Lin, and Bo Dai. Revisiting skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2969–2978, 2022. 1

[6] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *IEEE/CVF CVPR*, pages 1801–1810, 2019. 1, 6, 7

[7] Ehsan Elhamifar and Dat Huynh. Self-supervised multi-task procedure learning from instructional videos. In *ECCV*, pages 557–573. Springer, 2020. 7

[8] Ehsan Elhamifar and Zwe Naing. Unsupervised procedure learning via joint dynamic summarization. In *Proceedings of the IEEE/CVF ICCV*, pages 6341–6350, 2019. 6, 7

[9] Sanjay Haresh, Sateesh Kumar, Huseyin Coskun, Shahram N Syed, Andrey Konin, Zeeshan Zia, and Quoc-Huy Tran. Learning by aligning videos in time. In *IEEE/CVF CVPR*, pages 5548–5558, 2021. 1, 3, 6, 7

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[11] Youngkyoon Jang, Brian Sullivan, Casimir Ludwig, Iain Gilchrist, Dima Damen, and Walterio Mayol-Cuevas. Epic-tent: An egocentric video dataset for camping tent assembly. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 6

[12] Vishal Kaushal, Rishabh Iyer, Khoshrav Doctor, Anurag Sahoo, Pratik Dubal, Suraj Kothawade, Rohan Mahadev, Kunal Dargan, and Ganesh Ramakrishnan. Demystifying multi-faceted video summarization: Tradeoff between diversity, representation, coverage and importance. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 452–461. IEEE, 2019. 4

[13] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. Unsupervised learning of action classes with continuous temporal embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12066–12074, 2019. 7

[14] Yin Li, Miao Liu, and James M Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *Proceedings of the European conference on computer vision (ECCV)*, pages 619–635, 2018. 6

[15] Weizhe Liu, Bugra Tekin, Huseyin Coskun, Vibhav Vineet, Pascal Fua, and Marc Pollefeys. Learning to align sequential actions in the wild. *IEEE/CVF CVPR*, 2022. 1, 6, 7

[16] Francesco Ragusa, Antonino Furnari, Salvatore Livatino, and Giovanni Maria Farinella. The meccano dataset: Understanding human-object interactions from egocentric videos in an industrial-like domain. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1569–1578, 2021. 6

[17] M. Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2019. 4, 5

[18] Yair Shemer, Daniel Rotman, and Nahum Shimkin. Ils-summ: Iterated local search for unsupervised video summarization. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 1259–1266. IEEE, 2021. 3

[19] Yuhan Shen, Lu Wang, and Ehsan Elhamifar. Learning to segment actions from visual and language instructions via differentiable weak sequence alignment. In *IEEE/CVF CVPR*, 2021. 7

[20] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 1

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[22] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *IEEE transactions on pattern analysis and machine intelligence*, 41(11):2740–2755, 2018. 3

[23] Weiyu Zhang, Menglong Zhu, and Konstantinos G Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *Proceedings of the IEEE international conference on computer vision*, pages 2248–2255, 2013. 6

[24] Dimitri Zhukov, Jean-Baptiste Alayrac, Ivan Laptev, and Josef Sivic. Learning actionness via long-range temporal order verification. In *ECCV*, pages 470–487. Springer, 2020. 6