Why Not Use Your Textbook? Knowledge-Enhanced Procedure Planning of Instructional Videos

Supplementary Material

Unless otherwise mentioned, all the results and analysis are obtained for the CrossTask dataset with input visual features from the S3D network [9] pretrained on HowTo100M [8]. We organize the Supplementary Materials as follows:

A. Further Experimental Results

A.1 Additional Results on COIN and NIV

COIN. We present the full results of our method in comparison to several baseline approaches on the COIN dataset across different planning horizons in Table S.1. While the results demonstrate that our method outperforms the majority of previous literature, it does not secure the top position when considering small planning horizons (T=3 or T=4) for the COIN dataset. This can be attributed to the fact that the COIN dataset typically consists of only 3.9 actions per video on average, a scenario where advanced sequence-level procedural knowledge is not essential. The advantage of our method becomes increasingly pronounced as the planning horizon expands. With a larger planning horizon (T=5), our method significantly outperforms the previous state-of-the-art (SOTA) methods, achieving a performance gain of 6.16 on the most strict metric, Success Rate (SR), compared to SkipPlan [7]. Our method's utilization of plan recommendations from the Probabilistic Procedure Knowledge Graph (P^{2}KG) effectively reduces the complexity of long-horizon planning.

NIV. Previous works did not provide results for the NIV dataset concerning long-horizon procedural planning; in Table S.2. we evaluate our model's performance in comparison to the PDPP model [14] specifically on the NIV dataset for long-horizon procedural planning (T=5 or T=6). (For results regarding short-horizon planning, please refer to Table 3 in the main paper.) Our model demonstrates superior performance in terms of the SR, mAcc, and mIoU metrics.

A.2 Parameter Sensitivity Analysis on the Number of Plan Recommendations

Our proposed method involves querying a probabilistic procedure knowledge graph; this process extracts the most probable graph paths, which have specified start and end steps. These paths are subsequently considered as recommended procedural plans and are employed as additional input conditions for the model, with the aim of improving its overall performance in procedure planning. Consequently, the number of plan recommendations, denoted as R in the main paper, emerges as a novel hyper-parameter in our methodology.

We have conducted a parameter sensitivity analysis to examine the effect of the number of paths selected from the probabilistic procedure knowledge graph to be given as conditions to the planning model on the procedure planning performance. Table S.3 displays the parameter sensitivity of our method with respect to R on the CrossTask dataset for the planning horizon T=4. To maintain simplicity in implementation, in cases where R exceeds one, we aggregate the features of the top R graph paths through a linear weighting process (i.e., weighted summation). This results in consistent feature dimensions compared to when R=1. The weighting scheme is as follows:

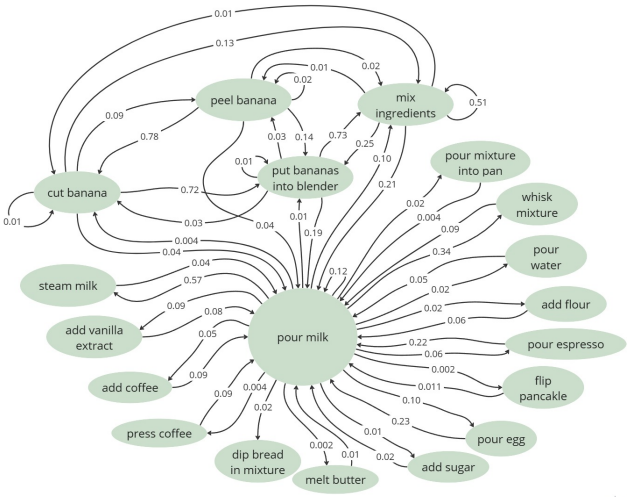
$R = 1 \rightarrow weights : 1$
$R = 2 \rightarrow weights : \frac{2}{3}, \frac{1}{3}$
$R = 3 \rightarrow weights : \frac{3}{5}, \frac{1}{5}, \frac{1}{5}$
$R = 4 \rightarrow weights : \frac{4}{7}, \frac{1}{7}, \frac{1}{7}, \frac{1}{7}$
$R = 5 \rightarrow weights : \frac{5}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}$

where the weights of the graph paths, starting with the most probable one, are provided above. The weights are empirically determined to emphasize the top one probable path by assigning it greater weight, and equal weights are distributed among the remaining graph paths.

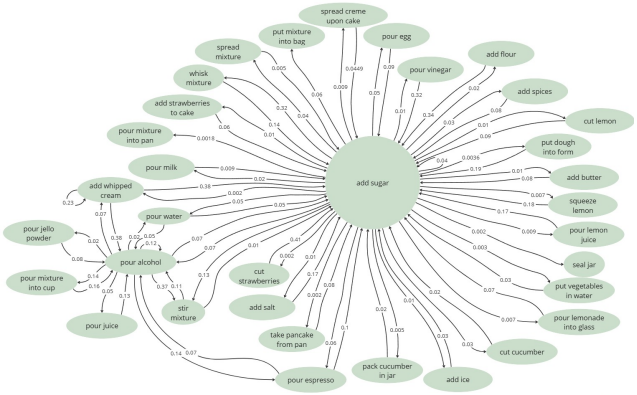
Our method, with R=2, achieves the best results in SR and mIoU (see Table S.3). In general, performance of our method initially increases and then decreases as R continues to increase. The reason for larger values of R yield-

Models	COIN (T=3)			COIN (T=4)			COIN (T=5)		
	$SR^{\uparrow}$	$mAcc^{\uparrow}$	$mIoU^{\uparrow}$	$SR^{\uparrow}$	$mAcc^{\uparrow}$	$mIoU^{\uparrow}$	$SR^{\uparrow}$	$mAcc^{\uparrow}$	$mIoU^{\uparrow}$
Random	< 0.01	< 0.01	2.47	< 0.01	< 0.01	2.32	-	-	-
Retrieval	4.38	17.40	32.06	2.71	14.29	36.97	-	-	-
DDN [4]	13.90	20.19	64.78	11.13	17.71	68.06	-	-	-
P^{3}IV [15]	15.40	21.67	76.31	11.32	18.85	70.53	4.27	10.81	68.81
E3P [13]	19.57	31.42	84.95	13.59	26.72	84.72	-	-	-
PDPP [14]	19.42	43.44	50.03	13.67	42.58	49.84	13.02	**43.36**	50.96
SkipPlan [7]	**23.65**	**47.12**	**78.44**	**16.04**	**43.19**	**77.07**	9.90	38.99	**76.93**
Ours (R=2)	20.25	39.87	51.72	15.63	39.53	53.27	**16.06**	40.72	56.15

Table S.1. **Performance of baselines and ours on the COIN dataset.** Our method excels at handling challenging planning scenarios that demand longer planning horizons (T); with T=5, ours achieves a performance gain of 6.16 on the most strict metric, Success Rate (SR)



(a)



(b)

Figure S.1. **Probabilistic Procedure Knowledge Graphs** (P^{2}KG) around the nodes 'peel banana' and 'add whipped cream' respectively up to a two-node depth. P^{2}KG effectively captures the task-sharing steps, variability in transition probabilities between steps, implicit temporal and causal relationships of steps, as well as the existence of numerous viable plans given an initial step and an end step

Models	NIV (T=5)			NIV (T=6)		
	$SR^{\uparrow}$	$mAcc^{\uparrow}$	$mIoU^{\uparrow}$	$SR^{\uparrow}$	$mAcc^{\uparrow}$	$mIoU^{\uparrow}$
PDPP [14]	18.95	37.26	87.50	14.94	41.02	93.70
Ours (R=2)	**21.58**	**39.79**	**91.66**	**17.53**	**43.62**	**93.75**

Table S.2. **Performance of baselines and ours on the NIV dataset.** Our method demonstrates superior performance on all metrics

Model	$SR^{\uparrow}$	$mAcc^{\uparrow}$	$mIoU^{\uparrow}$
PDPP [14]	18.69	52.44	62.38
Ours (R=1)	20.38	55.54	64.03
Ours (R=2)	**21.02**	56.08	**64.25**
Ours (R=3)	20.22	**56.19**	63.15
Ours (R=4)	20.76	55.63	64.22
Ours (R=5)	20.37	55.43	63.93

Table S.3. **Parameter sensitivity study on the CrossTask dataset.** Performance of our method initially increases and then decreases as R continues to increase. An excessively large value for R may lead to the inclusion of less prominent paths in the conditions given to the planning model

ing lower SR is attributed to the influence of less prominent paths being included in the conditions given to the planning model. Further tuning of the weighting scheme could potentially yield even more superior results which we leave for future work.

Finally, a consistent trend of higher performance of $R = 2$ over $R = 1$ as T increases is not always guaranteed. In Table 1 and Table 2 of the main paper, KEPP with R=2 has a weaker performance than $R = 1$ when T is 5, but stronger than $R = 1$ when T is 3, 4 or 6. This may arise from the stochastic nature of the diffusion model and the variability

(a) Add Oil to Your Car task

(b) Build Simple Floating Shelves task

(c) Make French Toast task

(d) Make Meringue task

Figure S.2. **Qualitative Analysis of Different Procedural Tasks.** In the illustration, 'P^{2}KGC =1' and 'P^{2}KGC =2' indicates the first and second paths obtained from the probabilistic procedure knowledge graph respectively

in the extent to which the top-2 graph plans offer additional beneficial information across different values of T.

B. More Thorough Analysis

B.1 More Visualizations of the Probabilistic Procedure Knowledge Graph

Figure S.1 (a) and (b) show the sub-graphs in our probabilistic procedure knowledge graph (P^{2}KG) around the nodes 'peel banana' and 'add whipped cream' up to a two-node depth respectively. P^{2}KG effectively captures the task-sharing steps, variability in transition probabilities between steps, implicit temporal and causal relationships of steps, as well as the existence of numerous viable plans given an initial step and an end step.

B.2 More Qualitative Results

Figure S.2 presents more qualitative results of our method compared with PDPP [14] across a range of procedural tasks, with a planning horizon set at T=4. In scenarios depicted in Figure S.2 (a), (c), and (d), the PDPP model erroneously predicts the initial or final steps based on the given visual states, resulting in procedural planning failures. Our approach, however, incorporates a specialized step model designed to accurately predict these crucial first and last steps. Furthermore, Figure S.2 highlights how our method benefits from the plan recommendations provided by the probabilistic procedure knowledge graph. These recommendations offer invaluable insights into feasible plans identified during training, thereby significantly enhancing our method's capability in procedure planning at test time.

B.3 Training Efficiency

Figure S.3 compares the training convergence of our planning model with PDPP [14]. We have plotted the loss values across the training epochs and indicated the epoch of convergence determined by the early stopping scheme. Our model, with R=2 and R=1, converged at the 30th and 40th epochs, respectively, while the previous state-of-the-art method, PDPP, converged at the 60th epoch. Our model exhibited significantly faster training convergence compared to PDPP. The training efficiency of our method is achieved by leveraging the probabilistic procedure knowledge graph, which provides valuable context to facilitate the model's learning process.

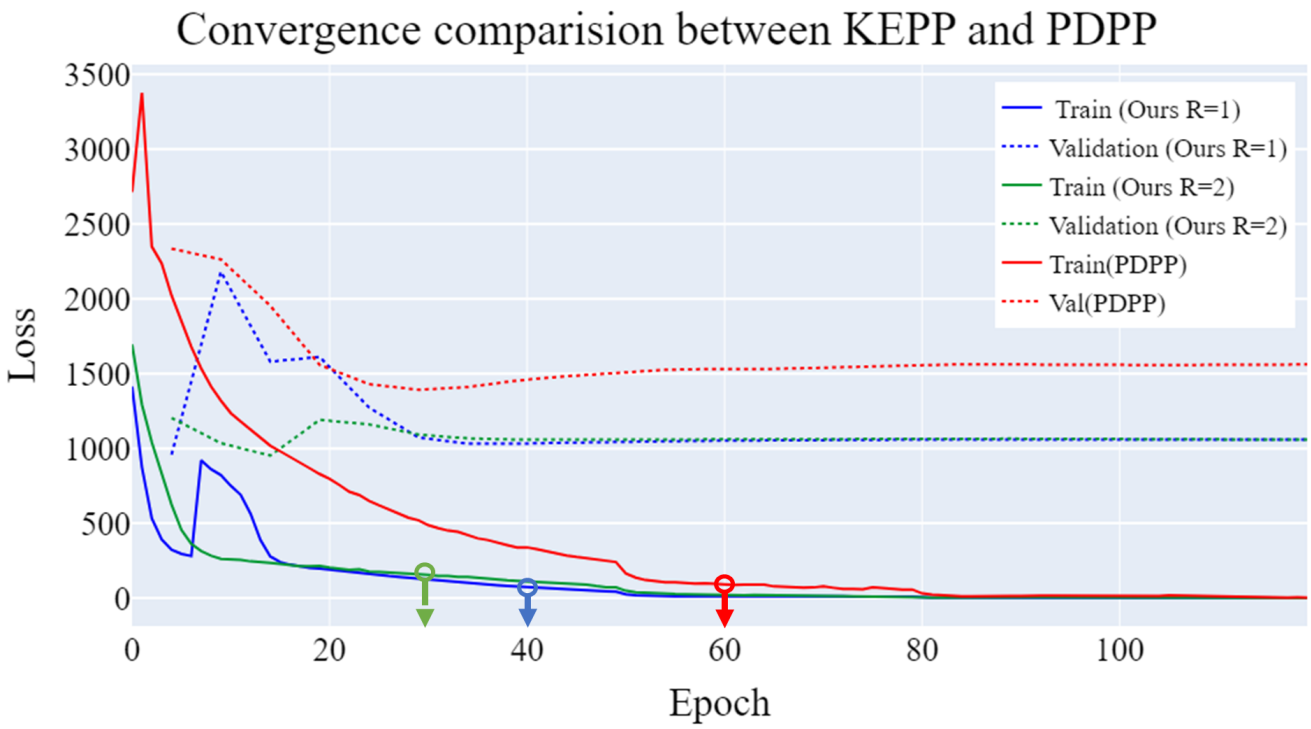
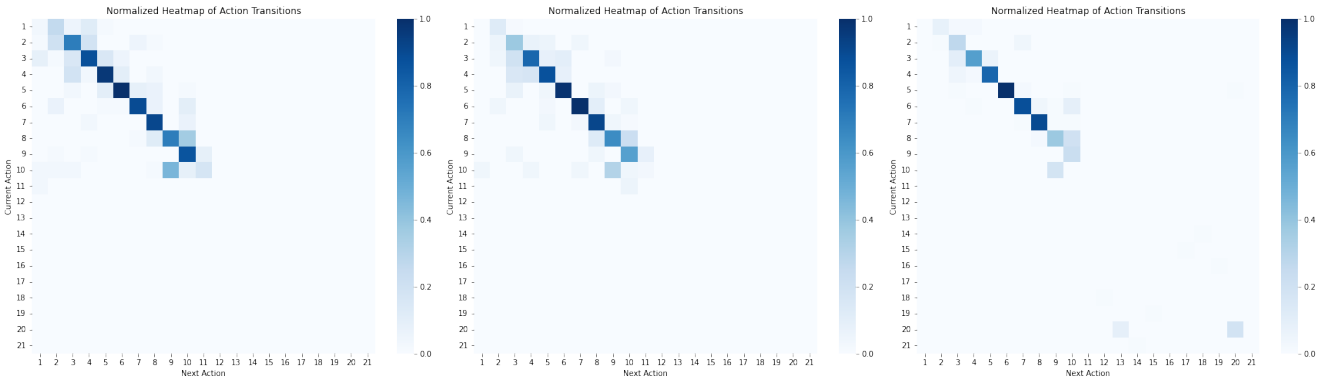


Figure S.3. **Training efficiency comparison between PDPP [14] and our model (KEPP).** These training and validation profiles demonstrate that our model exhibited significantly faster training convergence compared to PDPP. Our method with R=2 has faster convergence compared to R=1. The arrows indicate the epochs of convergence determined by the early stopping scheme



(a) Train (b) Test (c) Ours (R=1)

Figure S.4. **Step Transition Heatmaps on the CrossTask dataset of (a) the training set, (b) the testing set, and (c) procedure plan predictions made by our method on the test set**. The i-row-j-column depicts the probability of the transition from i-th action to j-th action. Darker color indicates higher probability. Please refer to Section B.4 for details regarding task and action names

B.4 Analysis by the Step Transition Heatmap

Figure S.4 visualises the step transition matrices of the ground-truth training procedure plans, ground-truth testing procedure plans, and procedure plan predictions made by our method with R=1 on the test set, for the 'Change a Tire' task on the CrossTask dataset. The i-row-j-column depicts the probability of the transition from i-th action step to j-th action step. Darker color of the step transition heatmap indicates higher probability. The steps depicted in the heatmaps in order are: "brake on":1, "get things out":2, "start loose":3, "jack up":4, "unscrew wheel":5, "withdraw wheel":6, "put wheel":7, "screw wheel":8, "jack

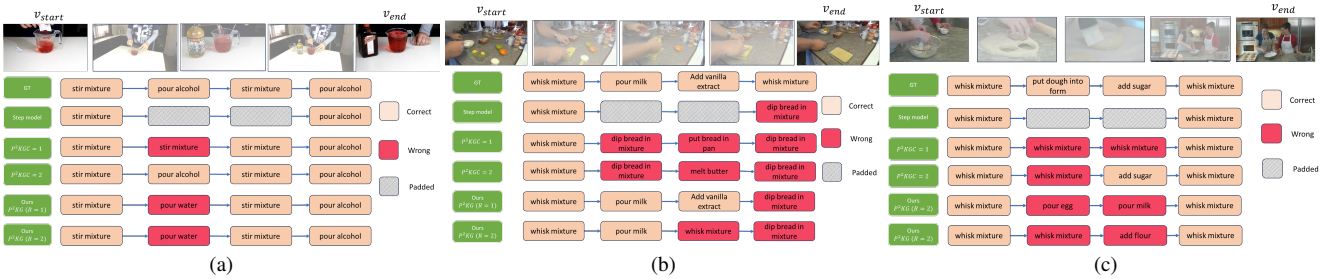


Figure S.5. **Failure Cases**: (a) when repetitive smaller action sequences exist within the procedure plan, (b) when a_1 and a_T have not been accurately predicted by the step model, and (c) when the start and end action steps are the same, and the probabilistic procedure knowledge graph fails to provide valuable plan recommendations because it overly encodes the repetitive actions

down":9, "tight wheel":10, "put things back":11, "remove funnel":12, "lower jack":13, "put funnel": 14, "wipe off dipstick": 15, "close cap":16, "insert dipstick":17, "pour oil":18, "pull out dipstick":19, "raise jack":20, "remove cap":21.

Out of the 21 actions illustrated, first 11 actions belong to the 'Change a Tire' task while other actions are from several other tasks but highly relevant to the 'Change a Tire' task (e.g., 'raise jack":20 and "lower jack":13). When comparing Figure S.4 (a) and (b), the distributions of action step transitions differ between training and testing. When comparing Figure S.4 (b) and (c), our method, which utilizes graph-structured training knowledge, reasonably predicts the step transition matrix on the test set.

B.5 Limitations and Failure Cases

Figure S.5 illustrates three distinct failure case patterns in our model: (a) failure in prediction when repetitive smaller action sequences exist within the procedure plan, (b) failure in accurately predicting a_1 and a_T from the step model, and (c) failure in generating valuable probabilistic procedure knowledge graph paths because the graph overly encodes the repetitive actions when the start and end action steps are the same. Consistent with prior studies, cases involving repetitive actions or action sequences continue to pose challenges. Future studies could consider addressing these challenging scenarios.

Below, we present analyses regarding how often are errors caused by incorrect $\hat{a}_1$ and $\hat{a}_T$, or the graph. On CrossTask♣ test data (T=4, R=2), 49.94% of predicted plans have incorrect $\hat{a}_1$ or $\hat{a}_T$, and 63.23% of error instances feature incorrect $\hat{a}_1$ or $\hat{a}_T$. If we use ground-truth a_1 and a_T at inference, success rate improves from 21.02 into 36.58. When analysing the errors of procedure knowledge graph, in 34% of correctly predicted plans, the top-1 graph plan mismatches the ground-truth plan, and in 87% of error instances, the top-1 graph plan mismatches the ground-truth plan.

The effectiveness of our method is constrained when procedural knowledge is not required. This is because procedural knowledge is structural information about procedures, often considered as *sequential* knowledge about steps in procedures [16], and KEPP uses graph *paths* as contextual plan recommendations (and graph paths can provide valuable procedural knowledge), scenarios with short plans restrict the efficacy of KEPP, which specializes in long planning horizons–the more challenging cases. KEPP's performance hinges on accurate predictions of $\hat{a}_1$ and $\hat{a}_T$ when T is small, less on the graph for planning; e.g., with T=3, the graph provides only the middle action, and thus procedural knowledge plays a minor role.

The analyses above imply that performance can be greatly boosted with more precise $\hat{a}_1$ and $\hat{a}_T$, and having multiple plan recommendations is advantageous, as the top graph path may not always align. The failure case analyses also highlight the limitations of our approach. Our method depends on the precise prediction of both the initial and final steps by the step model, and it performs more effectively when the probabilistic procedure knowledge graph can offer valuable contextual information.

B.6 Ablations with Flawless Step Model

The table S.4 points out the success rate, mAcc, and mIoU of our plan model for different horizons when the ground truth (GT) a_1 and a_T are used for training and testing the model. This provides an accurate depiction of the accuracy of the plans generated when the step (video perception) model generates perfect start and end steps.

B.7 Zero-shot Planning with the P^{2}KG

We conduct an experiment by using P^{2}KG for procedure planning as a parameter-free zero-shot approach on CrossTask♣ (R=1 since we directly use top graph plan). The success rates are 22.58, 17.74, 10.92, and 5.92 respectively for T=3,4,5,6; and 56.51, 32.40, 19.63, and 12.16 respectively if we assume a flawless video model.

This P^{2}KG-based zero-shot approach could achieve superior performance than current SOTA–for example, Skip-

Plan's results are 28.85, *15.56*, *8.55*, and *5.12*, respectively; see Table 1 and Table 2 in the main paper.

B.8 Further Discussions

How can KEPP handle novel tasks and scale? As mentioned in Sec.4 of the paper, there are trade-offs between using the LLM-generated recommendations and the P^{2}KG recommendations. For instance, the P^{2}KG recommendations are constrained by the training data, limiting their applicability to unseen procedural activities, whereas LLMs generally better generalize to these unseen activities. Nevertheless, it is possible for our proposed method to scale for internet-scale videos given an action list. One could address it by enumerating all admissible actions and mapping the KEPP's output actions to the most semantically-similar admissible actions. KEPP generalizes and scales better with a larger vocabulary in its graph, making internet-scale data beneficial for its development.

Why adapt a diffusion model for step recognition and planning? In the implementation of KEPP, we adapt PDPP [14] which is a diffusion model as the step model and the planning model. We outline our motivations as follows. First, a diffusion based procedure planning method avoids error propagation seen in non-diffusion-based prior models [3, 4, 11]. Secondly, it can be trained without complex multi-objective training processes [7, 13, 15]. Third, randomness is involved both for training and sampling in a diffusion model, which is helpful to model the uncertain action sequences for planning [14]. Furthermore, it is convenient to apply conditional diffusion and projection operations to mitigate uncertainty that might misguide learning. Finally, the learned weights of the diffusion step and planning models could mutually benefit (e.g., initialize the planning model with the learned step model weights, or finetune the step model with the planning model weights). Future work can consider transfer the learned weights of the diffusion step model towards the diffusion planning model.

C. Method and Implementation Details

Our code and trained models are publicly available.

C.1 Diffusion Model Details

Standard Diffusion Model. A standard denoising diffusion probabilistic model [6] tackles data generation by establishing the data distribution $p(x_0)$ through a denoising Markov chain over variables $\{x_N \dots x_0\}$, starting with x_N as a Gaussian random distribution.

In the forward diffusion phase, Gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is progressively added to the initial, unaltered data x_0, transforming it into a Gaussian random distribution. Each noise addition step is mathematically defined as:

$$x_n = \sqrt{\bar{\alpha}_n} x_0 + \epsilon \sqrt{1 - \bar{\alpha}_n} \tag{1}$$

$$q(x_n|x_{n-1}) = \mathcal{N}\left(x_n; \sqrt{1-\beta_n} x_{n-1}, \beta_n \mathbf{I}\right) \tag{2}$$

where $\bar{\alpha}_n = \prod_{s=1}^{n} (1 - \beta_s)$ represents the noise magnitude, and $\{\beta_n \in (0,1)\}_{n=1}^{N}$ denotes the pre-defined ratio of Gaussian noise added in each step.

Conversely, the reverse denoising process transforms Gaussian noise back into a sample. Each denoising step is mathematically defined as:

$$p_\theta(x_{n-1}|x_n) = \mathcal{N}(x_{n-1}; \mu_\theta(x_n, n), \Sigma_\theta(x_n, n)) \tag{3}$$

where μ_θ is parameterized as a learnable noise prediction model $\epsilon_\theta(x_n, n)$, optimized using a mean squared error (MSE) loss $L = \|\epsilon - \epsilon_\theta(x_n, n)\|^2$, and Σ_θ is calculated using $\{\beta_n\}_{n=1}^{N}$. During training, the model selects a diffusion step $n \in [1, N]$, calculates x_n via Eq. 1, then the learnable model estimates the noise and computes the loss based on the actual noise added at step n. After training, the diffusion model generates data akin to x_0 by iteratively applying the denoising process, starting from random Gaussian noise.

Conditioned Projected Diffusion Model. As we incorporate conditional information into the data distribution, these conditional guides can be altered during the denoising process. However, modifying these conditions can lead to incorrect guidance for the learning process, rendering conditional guides ineffective. To tackle this issue, the Conditioned Projected Diffusion Model has been introduced to ensure that guided information remains unaffected during denoising. In this approach, Wang et al. [14] introduce a condition projection operation within the learning process. Specifically, they enforce that visual observation conditions and additional condition dimensions remain unchanged during both training and inference by assigning them to their initial values. We have adapted their Conditioned Projected Diffusion Model as the architecture for both our step model and planning model in KEPP.

C.2 Implementation of KEPP

The step model for a_1 and a_T predictions and the planning model for procedure plan full-sequence prediction are trained separately. As mentioned in the main paper, for the step model, intermediate actions are padded in order to accurately predict a_1 and a_T. We use a U-Net based Conditioned Projected Diffusion Model [14] $f_\theta(x_n, n)$ as the learnable model architecture for both of the step model and the planning model, and the training loss is:

$$L = \sum_{n=1}^{N} (f_\theta(x_n, n) - x_0)^2 \tag{4}$$

where x_0 denotes the initial, unaltered input and N denotes the total number of diffusion steps.

Model	T=3			T=4			T=5			T=6		
	$SR^{\uparrow}$	$mAcc^{\uparrow}$	$mIoU^{\uparrow}$	$SR^{\uparrow}$	$mAcc^{\uparrow}$	$mIoU^{\uparrow}$	$SR^{\uparrow}$	$mAcc^{\uparrow}$	$mIoU^{\uparrow}$	$SR^{\uparrow}$	$mAcc^{\uparrow}$	$mIoU^{\uparrow}$
Ours	33.38	60.79	63.89	21.02	56.08	64.15	12.74	51.23	63.16	9.23	50.78	65.56
Ours with GT a_1, a_T	57.50	84.85	80.75	37.17	77.01	78.58	20.51	67.66	73.97	15.08	65.27	74.37

Table S.4. Comparison of the performance when ground truth (GT) a_1 and a_T are used for training and testing the planning model in our KEPP system (R=2) on the CrossTask$^{\clubsuit}$ dataset

In line with the approach taken in [14], given that a_1 and a_T are the most relevant actions for the provided input visual states, we have modified the training loss by implementing a weighted MSE loss. This involves multiplying the loss by a weight matrix to give more emphasis to the predictions of a_1 and a_T. For the step model, we have chosen a weight of 10 for both a_1 and a_T. In the case of the planning model, we have assigned a weight of 5 to a_1 and a_T because the condition dimensions of intermediate steps in the probabilistic procedure knowledge graph path has a more significant impact on generating the full action step sequence. The default value in this weight matrix is 1.

The batch size is 256 for all the experiments. Our training regimen incorporates a linear warm-up strategy that is tailored to suit the varying scales of different datasets. For the results of models utilizing the precomputed features provided in CrossTask, we use 200 for the diffusion step and train the progress through a total of 12,000 training steps. The learning rate is gradually escalated to reach 8×10^{-4} over the initial 4,000 steps. Subsequent to this phase, we implement a learning rate reduction to 4×10^{-4} when reaching the 10,000th step.

When working with the S3D network-extracted features on the CrossTask dataset, the model similarly starts with a diffusion step of 200 but extends the training duration to 24,000 steps. The learning rate here ascends linearly to 5×10^{-4} within the first 4,000 steps, followed by a decay factor of 0.5 applied sequentially at the 10,000th, 16,000th, and 22,000th steps.

The NIV dataset, given its smaller size, necessitates a shorter training cycle of 6,500 steps, starting from a diffusion step of 50. We increase the learning rate linearly to 3×10^{-4} up until step 4,500, then introduce a decay by 0.5 at step 6,000.

For the COIN dataset, which is significantly larger in scale, the model undergoes an extended training sequence of 160,000 steps with the diffusion set at 200. Here, the learning rate experiences a linear surge to 1×10^{-5} within the first 4,000 steps. We implement a decay rate of 0.5 at the 14,000- and 24,000-step marks. After this point, the learning rate is maintained at a constant 2.5×10^{-6} for the remainder of the training process.

When obtaining mIoU, for CrossTask dataset we compute IoU on every single action sequence and calculates the average of these IoUs to obtain mean IoU similar to method used in [14]. For COIN and NIV datasets, we compute mIoU on every mini-batch (batch size = 256) and calculate the average as the result similar to the approach used in [15].

In the scenario where paths from the probabilistic procedure knowledge graph are available between $\hat{a}_1$ and $\hat{a}_T$ but it does not match the desired number of paths (R), then repetition of already available paths are considered to meet the desired path requirement. In scenarios where the combination of $\hat{a}_1$ and $\hat{a}_T$ does not result in the inclusion of a path from the probabilistic procedure knowledge graph, the graph paths (P) are formulated as follows:

$$\text{Path Variations } (L) = [[[\hat{a}_1]^{T-M}.[\hat{a}_T]^{M}], [[\hat{a}_1]^{M}.[\hat{a}_T]^{T-M}]]$$

$$\text{Generated Paths } (P) = [L[i \mod \text{len}(L)] \text{ for } i \text{ in range } (R)]$$

where T is the planning horizon, $M = T//2$, and path variations (L) are the two types of possible combinations of $\hat{a}_1$ and $\hat{a}_T$. For example if horizon (T) equals 5, then the path variations (L) possible are [[$\hat{a}_1$, $\hat{a}_1$, $\hat{a}_1$, $\hat{a}_T$, $\hat{a}_T$], [$\hat{a}_1$, $\hat{a}_1$, $\hat{a}_T$, $\hat{a}_T$, $\hat{a}_T$]]. So if one path ($R = 1$) is required as the condition for the planning model, generated path is [$\hat{a}_1$, $\hat{a}_1$, $\hat{a}_1$, $\hat{a}_T$, $\hat{a}_T$], and if 2 paths are required ($R = 2$) as the condition for the planning model, generated paths are [$\hat{a}_1$, $\hat{a}_1$, $\hat{a}_1$, $\hat{a}_T$, $\hat{a}_T$], and [$\hat{a}_1$, $\hat{a}_1$, $\hat{a}_T$, $\hat{a}_T$, $\hat{a}_T$], and if 3 paths are required ($R = 3$) as the condition for the planning model, generated paths are [$\hat{a}_1$, $\hat{a}_1$, $\hat{a}_1$, $\hat{a}_T$, $\hat{a}_T$], [$\hat{a}_1$, $\hat{a}_1$, $\hat{a}_T$, $\hat{a}_T$, $\hat{a}_T$], and [$\hat{a}_1$, $\hat{a}_1$, $\hat{a}_1$, $\hat{a}_T$, $\hat{a}_T$] likewise. The generated paths (P) are then aggregated through linear weighting into a single path to be given to the planning model as mentioned in Section A.2.

C.3 Implementation of Ablations

Ablation on the probabilistic procedure knowledge graph. When the probabilistic procedure knowledge graph is not included as a condition, the model simplifies only to the planning model, ignoring the step model for generating $\hat{a}_1$ and $\hat{a}_T$. In this scenario, the planning model generates the action sequences using a diffusion process conditioned solely on the start and end visual observations.

Plan recommendations provided by the probabilistic procedure knowledge graph and LLM. To enhance the action prediction, we could incorporate LLM generated action sequence as a condition to our planning diffusion model

as shown in Table 6 in the main paper. To generate the LLM action sequence we use the 'llama-2-13b-chat' or the 'llama-2-70b-chat' model and query the model with the prompt:

"In the multi-step task that I am going to perform, I know {total_steps} steps are required, the first step is {start_action}, and the last step is {end_action} and I need to go from {start_action} to {end_action}. Can you tell me the {total_steps} steps in the form "Step 1, ..., Step {total_steps}" so that I can follow in order to complete this {total_steps}-step task? Note that repetitive steps are possible".

In the above text prompt, 'total_steps' indicates the planning horizon, while 'start_action' denotes the step model-predicted start action $\hat{a}_1$ and 'end_action' denotes the step model-predicted end action $\hat{a}_T$. For a single $[\hat{a}_1, \hat{a}_T]$ pair occurring in the training or testing phase, we query the LLM three times and obtain the optimal action for each step in the sequence by finding the action with the highest occurrence. Then, the optimal sequence is queried back into the LLM for post-processing to check the realistic nature of the generated action sequence. The text prompt used for post-processing verification is as follows:

"Is the following action sequence possible? Note that repetitive steps are possible. Please answer only "Yes" or "No": {steps}".

Here, '{steps}' denotes the the generated steps from the first prompt. We then map the action steps generated from the LLM to the step vocabulary space of the given dataset. Specifically, the generated action steps from the LLM, as well as all of the possible action steps from the dataset, are embedded into a vector space utilizing the 'bert-base-uncased' model. For each action in the LLM-generated sequence, its semantically closest action step from the dataset is found in the vector space using the K-nearest neighbor (K=1) approach. The resulting generated action sequence is given as an additional input condition to the planning model similar to the P^{2}KG condition.

In the situation where both the P^{2}KG and LLM conditions are used for the planning model, the conditional visual states, LLM recommendation and the procedure plan recommendation from the P^{2}KG are concatenated with the actions along the action feature dimension, forming a multi-dimensional array:

(5) where $\tilde{a}_i$ denotes the action from the P^{2}KG at i^{th} step and a_i^* denotes the action generated from LLM at i^{th} step.

Probabilistic procedure knowledge graph (P^{2}KG) and Frequency-based procedure knowledge graph (PKG). The frequency-based procedure knowledge graph (PKG) performs min-max normailization for the entire graph, and the optimal path between two nodes is the maximum weighted path. The path weight is calculated by summing all the individual edge weights between the two nodes. The probabilistic procedure knowledge graph (P^{2}KG) has undergone output edge normalization, where the summation of probabilities of all the output paths from a node is 1. For example, if a node has four output edges (self-loop edges are also considered as output edges), and the weights along the edges are w_1, w_2, w_3, w_4, then the probability of each edge is: $\frac{w_1}{w_1+w_2+w_3+w_4}, \frac{w_2}{w_1+w_2+w_3+w_4}, \frac{w_3}{w_1+w_2+w_3+w_4}, \frac{w_4}{w_1+w_2+w_3+w_4}$ respectively. The process of obtaining the highest probable path between two nodes is explained in Section 3.2.3 of the main paper.

C.4 Datasets and Evaluation Metrics

In our evaluation, we employed datasets from three sources: CrossTask [17], COIN [12], and the Narrated Instructional Videos (NIV) [2]. The CrossTask dataset comprises 2,750 video clips, each representing one of 18 distinct tasks, and features an average of 7.6 ± 4.4 (mean$\pm$std) actions per clip. The COIN dataset is more extensive, including 11,827 videos across 180 tasks, with an average of 3.9 ± 2.4 actions per video. Lastly, the NIV dataset, though smaller in scale, includes 150 videos that capture 5 everyday tasks, with a higher density of actions, averaging 8.8 ± 2.8 actions per video. We randomly select 70% data for training and 30% for testing as previous work [3, 14, 15].

Our study employs mean intersection over union (mIoU), mean accuracy (mAcc), and success rate (SR) as our evaluation metrics. mIoU evaluates the overlap between the predicted actions with the ground-truth actions using IoU by regarding them as two action sets. mAcc is quantified as the average of the precise correspondences between the predicted actions and their respective ground truth counterparts at corresponding time steps. Thus, mAcc counts the order of actions, and is stricter than mIoU. **SR is the most stringent metric**, considering a sequence to be successful solely if it exhibits a perfect match with the ground truth action sequence throughout.

C.5 Baselines

In the current study, we benchmark our proposed approach against several established state-of-the-art methods to con-

duct a comprehensive evaluation. Below, we describe the methods that serve as our baseline comparisons.

Random Strategy: This naive approach involves the stochastic selection of actions from the set of possibilities within the dataset to generate procedure plans.

Retrieval-based Technique: Upon receiving the visual observational inputs, this method employs a nearest-neighbour algorithm, leveraging the least visual feature space distance within the training corpus to extract a corresponding sequence of actions as the procedural blueprint.

WLTDO [5] ['CVPR 2018']: Implemented via a recurrent neural network (RNN), this method sequentially forecasts the necessary actions based on the supplied pair of observations.

UAAA [1] ['ICCV workshop 2019']: Standing for a two-phased strategy, UAAA integrates an RNN with a Hidden Markov Model (HMM) to autoregressively estimate the sequence of actions.

UPN [10] ['ICML 2018']: This path-planning algorithm is tailored for the physical realm and acquires a plannable representational form for making informed predictions.

DDN [4] ['ECCV 2020']: The DDN framework operates as a dual-branch autoregressive model, honing in on an abstracted representation of action sequences to anticipate the transitions between states and actions in the feature space.

PlaTe [11] ['RA-L 2022']: An advancement of DDN, the PlaTe model incorporates transformer modules within its dual-branch architecture to facilitate prediction.

Ext-GAIL [3] ['ICCV 2021']: Ext-GAIL addresses procedural planning through reinforcement learning and decomposes the problem into a dual-stage challenge. The initial stage in Ext-GAIL is designed to yield time-invariant contextual information for the subsequent planning phase.

P^{3}IV [15] ['CVPR 2022']: A transformative single-branch model, P^{3}IV enriches its structure with a mutable memory repository and an additional generative adversarial network. It is capable of concurrently predicting the entire action sequence during the inference stage.

PDPP [14] ['CVPR 2023']: PDPP conceptualizes the task of procedure planning in instructional videos as analogous to fitting a probabilistic distribution. Echoing the P^{3}IV, PDPP employs a holistic approach during the inference stage by forecasting the entire sequence of action steps concurrently.

SkipPlan [7] ['ICCV 2023']: Consider procedure planning as a mathematical chain problem. This model decomposes relatively long chains into multiple short sub-chains by bypassing unreliable intermediate actions. This transfers long and complex sequence functions into short but reliable ones.

E3P [13] ['ICCV 2023']: This uses procedural task (referred as 'event' in [13]) information for procedure planning by encoding task information into the sequential modeling process. E3P infers the task information from the observed states and then plans out actions based on both the states and the predicted task/event.

References

[1] Yazan Abu Farha and Juergen Gall. Uncertainty-aware anticipation of activities. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 9

[2] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4575–4583, 2016. 8

[3] Jing Bi, Jiebo Luo, and Chenliang Xu. Procedure planning in instructional videos via contextual modeling and model-based policy learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15611–15620, 2021. 6, 8, 9

[4] Chien-Yi Chang, De-An Huang, Danfei Xu, Ehsan Adeli, Li Fei-Fei, and Juan Carlos Niebles. Procedure planning in instructional videos. In *European Conference on Computer Vision*, pages 334–350. Springer, 2020. 2, 6, 9

[5] Kiana Ehsani, Hessam Bagherinezhad, Joseph Redmon, Roozbeh Mottaghi, and Ali Farhadi. Who let the dogs out? modeling dog behavior from visual data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4051–4060, 2018. 9

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 6

[7] Zhiheng Li, Wenjia Geng, Muheng Li, Lei Chen, Yansong Tang, Jiwen Lu, and Jie Zhou. Skip-plan: Procedure planning in instructional videos via condensed action space learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10297–10306, 2023. 1, 2, 6, 9

[8] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2630–2640, 2019. 1

[9] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9879–9889, 2020. 1

[10] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *International Conference on Machine Learning*, pages 4732–4741. PMLR, 2018. 9

[11] Jiankai Sun, De-An Huang, Bo Lu, Yun-Hui Liu, Bolei Zhou, and Animesh Garg. Plate: Visually-grounded plan-

ning with transformers in procedural tasks. *IEEE Robotics and Automation Letters*, 7(2):4924–4930, 2022. 6, 9

[12] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1207–1216, 2019. 8

[13] An-Lan Wang, Kun-Yu Lin, Jia-Run Du, Jingke Meng, and Wei-Shi Zheng. Event-guided procedure planning from instructional videos with text supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13565–13575, 2023. 2, 6, 9

[14] Hanlin Wang, Yilu Wu, Sheng Guo, and Limin Wang. Pdpp: Projected diffusion for procedure planning in instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14836–14845, 2023. 1, 2, 3, 4, 6, 7, 8, 9

[15] 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*, pages 2938–2948, 2022. 2, 6, 7, 8, 9

[16] Honglu Zhou, Roberto Martín-Martín, Mubbasir Kapadia, Silvio Savarese, and Juan Carlos Niebles. Procedure-aware pretraining for instructional video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10727–10738, 2023. 5

[17] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3537–3545, 2019. 8