P³IV: Probabilistic Procedure Planning from Instructional Videos with Weak Supervision

He Zhao¹,² Isma Hadji¹ Nikita Dvornik¹ Konstantinos G. Derpanis¹,² Richard P. Wildes¹,² Allan Jepson¹
¹Samsung AI Centre Toronto, ²York University
{zhul, kosta, wildes}@eecs.yorku.ca, {isma.hadji, n.dvornik, allan.jepson}@samsung.com

Abstract

In this paper, we study the problem of procedure planning in instructional videos. Here, an agent must produce a plausible sequence of actions that can transform the environment from a given start to a desired goal state. When learning procedure planning from instructional videos, most recent work leverages intermediate visual observations as supervision, which requires expensive annotation efforts to localize precisely all the instructional steps in training videos. In contrast, we remove the need for expensive temporal video annotations and propose a weakly supervised approach by learning from natural language instructions. Our model is based on a transformer equipped with a memory module, which maps the start and goal observations to a sequence of plausible actions. Furthermore, we augment our model with a probabilistic generative module to capture the uncertainty inherent to procedure planning, an aspect largely overlooked by previous work. We evaluate our model on three datasets and show our weakly-supervised approach outperforms previous fully supervised state-of-the-art models on multiple metrics.

1. Introduction

Procedure planning is a natural task for humans – one must plan out a sequence of actions that takes one from the current state to the desired goal. While effortless for humans, procedure planning is notoriously hard for artificial agents. Nevertheless, solving procedure planning is of great importance for building next-level artificial intelligence systems capable of analyzing and mimicking human behaviour, and eventually assisting humans in goal-directed problem solving, e.g., cooking, assembling furniture or tasks that can be represented as a clear set of instructions. Traditionally, procedure planning has been addressed in structured environments, such as object manipulation on a table surface [13, 43]. While restricting the environment helps improve planning, it also limits the range of possible applications. Here, we follow more recent work [8] and tackle procedure planning in the realm of instructional videos [46, 54]. Given visual observations of the start and goal states, the task is to predict a sequence of high-level actions needed to achieve a goal; see Fig 1. This task is particularly challenging as it requires parsing unstructured environments, recognizing human activities and understanding human-object interactions. Yet, the range of applications for such planners is broad, which motivates research efforts on this problem.

Current approaches for procedure planning from instructional videos share a serious limitation – reliance on strong supervision with expensive annotations [6, 8, 47]. Specifically, all such methods require access to (i) a list of action labels used to transition from start to the goal state and (ii)
the visual representation of the intermediate states. Using such intermediate visual representations entails very expensive annotation of the start and end times of each intermediate instructional step; see Fig 1 (bottom). In contrast, our work removes the need for intermediate visual states during training and instead uses their linguistic representation for supervision. Relying on language representations allows us to better leverage instructional videos and significantly reduce the labeling effort; see Fig. 1 (top). For example, language annotations for the intermediate instructional steps could be extracted from general procedure descriptions available in recipes or websites, e.g., WikiHow [21]. In contrast, to obtain the timestamps for intermediate instructional steps one must watch the entire instructional video. In addition, a language representation can be a more stable supervisory signal [40], as the description of a given step (e.g., add seasoning) remains the same, while its visual observation varies across different videos.

Previous work on procedure planning from video relies on a two-branch autoregressive approach while adopting different architectures to model these branches [6,8,45]. In such models, one branch is dedicated to predicting actions based on the previous observation, while the other approximates the observation given the previous action in a step-by-step manner. Such models are cumbersome and compound errors, especially for longer sequences. In contrast, we rely on a single branch non-autoregressive model, implemented as a transformer [47] that generates all intermediate steps in parallel conditioned on the start and goal observations.

Another important factor in procedure planning is to model the uncertainty inherent to the prediction task. For example, given a set of ingredients and the goal of making a pancake, the intermediate steps could be either (i) [add wet ingredients → add dry ingredients → whisk mixture] or (ii) [add dry ingredients → add wet ingredients → whisk mixture]. This example shows that in realistic scenarios some plans can vary even under a shared common goal. This observation is usually handled in physical path planning tasks (e.g., robotic arms are allowed to follow multiple feasible trajectories [43]); yet, effort is lacking on probabilistic modeling of procedure planning from instructional videos. While previous work included a probabilistic component at training time [6], we are the first to use and benefit from multiple plausible plans at inference. We explicitly handle uncertainty in procedure planning with a dedicated generative module that can produce multiple feasible plans.

Contributions. In summary, the main technical contributions of our work are threefold. (i) We introduce a weakly supervised approach for procedure planning, which leverages powerful language representations extracted from pre-trained text-video embeddings. (ii) We tackle the task with a simpler single branch model, which can generate all intermediate steps in parallel, rather than relying on the two-branch auto-regressive approach used in previous work. (iii) We propose a generative adversarial framework, trained with an extra adversarial objective, to capture the stochastic property of planned procedures. We evaluate our approach on three widely used instructional videos datasets and show state-of-the-art performance across different prediction time horizons, even while relying on weaker supervision. We also show the advantage of modeling uncertainty. Our code is available at: https://github.com/SamsungLabs/procedure-planning.

2. Related work

Procedure planning. Traditionally, goal-conditioned planning has been studied mostly in physical environments, e.g., robotic motion planning [14,16,23] and human pedestrian trajectory planning [33]. Recently, the task of procedure planning from instructional videos was introduced [8]. Various approaches have made use of recurrent neural networks (RNNs) [8], transformers [45] and adversarial policy planning [6]; all have used two-branches and strong visual supervision. In contrast, we model the actions directly using a non-autoregressive transformer-based architecture. More importantly, we use low-budget weak supervision in the form of language instructions instead of supervising the model with “costly” visual observations as done by all existing approaches.

Supervision with natural language. A common alternative to training visual models using manually defined label sets is to exploit semantic supervision from natural language. Using natural language as supervision has several advantages: (i) language annotations can be collected automatically [37]; (ii) modeling language and vision jointly can produce stronger representations [10]; (iii) such supervision can achieve better generalization to unseen domains [40]. Such benefits resulted in a growing interest in using language as supervision for variety of tasks, e.g., image classification [10,17,22], representation learning [36,50], video retrieval [15,37], step localization [11,36] and navigation with instruction following [38]. We use advances in joint video and language modeling for procedure planning. We use pre-trained features [36] to map language and video to a common space and replace expensive video supervision with readily available language instructions.

Sequence modeling with transformers. Procedure planning is a task of conditional sequence prediction and thus it directly benefits from recent advances in sequence modeling. One of the strongest recent approaches to sequence modeling is the transformer architecture [47], which has been adopted for a wide variety of tasks, e.g., images [26], videos [4] and multi-modal data [31,32,50] tasks. Recent work adopted the transformer decoder architecture for fixed-size set prediction via learnable input
queries \cite{7, 52}. We build upon similar ideas by setting the first and last queries to correspond to the start and goal observations, while making the intermediate queries learnable. To improve long-range sequence modeling and help overall sequence coherence, recent work augmented transformers with an explicit external memory \cite{29, 49}. For the same reasons, we also integrate a learnable memory module.

**Future prediction.** The task of procedure planning is closely related to future prediction, where only past observations are provided as input. A key consideration in future prediction is modeling prediction uncertainty from partial initial observations. One common approach for modeling uncertainty is a Variational Auto Encoder (VAE) \cite{25} that captures the distribution over future actions. Another approach is to use generative adversarial networks (GANs) to forecast multiple, distinct and high quality future activities \cite{39, 53}. In this work, we adopt the generative modeling framework to model distributions over possible plans.

### 3. Technical approach

Here, we present our approach to procedure planning that relies on three main components. First, we predict all steps in the plan in parallel using a non-autoregressive transformer decoder. To obtain coherent plan predictions, our transformer is augmented with a learned memory shared across all possible tasks in a given dataset (Sec. 3.2). Second, to model the uncertainty inherent to the task, we include a generative component trained with an adversarial loss. As a result, we can infer multiple feasible plans conditioned on start and goal observations (Sec. 3.3). Third, to supervise the transformer’s outputs, we use the cross-entropy loss on action predictions and a contrastive loss to mediate state predictions, as illustrated in Fig. 2.

#### 3.1. Problem formulation

Given a start visual observation, \(v_{\text{start}}\), and a desired visual goal, \(v_{\text{goal}}\), our task is to predict a plan defined as the sequence of \(T\) intermediate action steps, \(\hat{\pi} = a_{1:T}\), taken to transition from \(v_{\text{start}}\) to \(v_{\text{goal}}\). We overscore with ~ to indicate our predictions, while the lack of such overscoring indicates ground truth (GT). At training time, given \(v_{\text{start}}\) and \(v_{\text{goal}}\), we predict a plan, \(\hat{\pi}\), and corresponding visual observations, \(\hat{v}_{1:T}\). We use the intermediate action labels, \(a_{1:T}\), to train the plan prediction, \(\hat{\pi}\), and corresponding language descriptions (embedded with a pre-trained text encoder), \(l_{1:T}\), to supervise the intermediate visual observations, \(\hat{v}_{1:T}\). That is, we substitute the visual information about intermediate instruction steps, \(v_{1:T}\), with their language counterparts, \(l_{1:T}\), to train the planner; see Fig. 2. We believe such supervision substitution is meaningful, as we are using a strong pre-trained vision-language encoder \cite{36}, mapping visual activities and their descriptions in a common space, thus making visual, \(v_t\), and language, \(l_t\), features, corresponding to the same activity, interchangeable for training.

In contrast, previous work assumes access to the set of intermediate action-observation pairs (i.e., \(a_{1:T}, v_{1:T}\)) \cite{6, 8, 45}, thereby requiring strong supervision to identify all intermediate visual observations. At inference time, we only use the start and goal observation, to predict a plan, \(\hat{\pi} = a_{1:T}\), for a given time horizon, \(T\).

#### 3.2. Memory augmented transformer decoder

To implement our planner, we use a non-autoregressive transformer decoder architecture \cite{7, 52}. Our transformer decoder takes two input types; namely, learnable queries augmented with the start and goal observations and a learned memory component, and outputs action and intermediate state predictions, as illustrated in Fig. 2.

**Conditioned learned-query input.** The first input is the query set, \(Q = [q_{\text{start}}, q_1, \ldots, q_{T-1}, q_{\text{goal}}]\), where the first and last inputs correspond to the representations of our initial and goal visual observations, \(v_{\text{start}}\) and \(v_{\text{goal}}\), respectively, while \(q_{1:T-1}\) are a set of learned queries. Queries, \([q_1, \ldots, q_{\text{goal}}]\), are associated with the action labels, \(a_{1:T}\), that we wish to predict. To communicate information about the order of elements to the decoder, we add to each query a fixed cosine positional embedding \cite{5}, \(p_t\), as follows

\[
Q = [q_{\text{start}} + p_0, \ldots, q_t + p_t, \ldots, q_{\text{goal}} + p_T],
\]

where \(q_t\) and \(p_t\) all are encoded as \(d\) dimensional embeddings, (i.e., \(q_t, p_t \in \mathbb{R}^d\)) and \(t = 1, \ldots, T - 1\).

**Learned memory input.** The second input to our transformer decoder is a learned memory component that is common across all examples in a given dataset. The memory is defined as a set of \(d\)-dimensional vectors

\[
M = [m_1, m_2, \ldots, m_n] \in \mathbb{R}^{d \times n},
\]

where \(n\) is the number of learnable vectors in the memory bank. Notably, the size of the memory (i.e., number of entries, \(n\), in the memory) is a hyperparameter that is independent from the prediction time horizon. We use read-only memory \cite{34} and share it among all layers for simplicity.

**Memory-augmented transformer decoder.** Our architecture is a stack of standard transformer decoder blocks \cite{47} (see Fig. 3), where each such block has access to the global learnable memory, (2). Specifically, the memory-augmented transformer block consists of two key operations. First, the input is processed with the self-attention operation. Second, the cross-attention module attends to the learnable memory to generate the output. The input to
tional on the input, we augment the entire query input, (1), with a random noise vector, \( z \sim \mathcal{N}(0, 1), z \in \mathbb{R}^d \), through concatenation. The new query input sequence to our transformer, \( \mathcal{T} \), thus becomes

\[
Q^z = \{ q_t; z \mid q_t \in Q \}. \tag{4}
\]

We employ adversarial training wherein the generator, \( G \), is trained to produce realistic action sequences, while the critic, \( C \), provides the supervisory signal for training \( G \) \[3\]. In our case, we treat the memory-augmented transformer, \( \mathcal{T} \), as the generator, \( G \) \( (i.e., \quad G = h_v(\mathcal{T}(Q^z, M)) \), while the critic is modeled by a simple MLP. More precisely, we pass the output of our transformer, \( \tilde{v}_{1:T} \), concatenated along the temporal dimension, to the critic, \( C \), which outputs a value between 0 and 1, indicating its ability to discriminate between the predicted and ground truth sequence, as depicted in Fig. 2. Notably, to avoid the notorious issue of mode collapse associated with training GANs \[44\] \( (i.e., \quad \text{regardless of variations in random latent noise, } z) \), we follow previous work \[30, 51, 53\] and include the normalized distance regularizing loss, \( L_{\text{reg}} \), defined in the supplement.

### 3.4. Training

To supervise our transformer, we rely on two complementary loss functions that enforce our transformer to decode the correct set of action labels in the procedure as well as corresponding visual representations. We also use an adversarial loss to train the stochastic component of our model.

#### Visual step supervision

One of the outputs of our model at training time is the sequence of visual features, corresponding to the procedure steps, \( v_{1:T} \). To supervise the visual features with corresponding language features, \( l_v \), we adopt contrastive learning \[19\]. For each feature, \( v_t \), predicted by the transformer’s head, \( h_v \), we use the corresponding ground truth language embedding, \( l_t \), as the pos-
Overall, our full loss function is defined as Complete loss.

\[ L_l = - \sum_{t=1}^{T} \left[ \log \frac{\exp(l_t \cdot \nu_t)}{\sum_j \exp(l_t \cdot \nu_j)} \right] \tag{5} \]

where (\cdot) denotes the dot-product operator. Note, we use all examples in the language vocabulary as negatives as our vocabulary is typically small (< 1K elements) and doing so allows for better training compared to per-batch negative sampling, e.g., [20].

**Action plan supervision.** We also enforce the action prediction head, \( h_a \), to output sequences of action probabilities, \( \tilde{a}_t \), corresponding to ground truth one-hot labels, \( a_t \). For this purpose, we use the cross-entropy loss

\[ L_a = - \sum_{t=1}^{T} a_t \log \tilde{a}_t. \tag{6} \]

**Adversarial supervision.** To model uncertainty, we use adversarial training on the visual state predictions, \( \tilde{v}_{1:T} \). The goal is to make predicted visual observation sequences indistinguishable from feature sequences composed of the ground truth language step description, \( l_{1:T} \). We optimize the generator, \( G \), (our transformer) and the critic, \( C \), (an MLP) using an adversarial loss [3]

\[ L_{adv} = \min_C \max_G V(G, C, Q^c, M), \tag{7} \]

where \( V \) is the standard GAN objective, defined as \( E_{t \sim P_{data}} [\log C(l_t)] + E_{z \sim P_z} [\log (1 - C(h_a(T(Q^z, M))))] \), with \( l \sim P_{data} \) and \( z \sim P_z \) denoting the data distributions of the language representation and random noise, resp.

**Complete loss.** Overall, our full loss function is defined as

\[ L(\theta) = \lambda_1 L_l + \lambda_2 L_a + \lambda_3 L_{adv} + \lambda_4 L_{reg}, \tag{8} \]

where \( \theta \) refers to the parameters associated with all learnable modules, i.e., queries, memory module, as well as the transformer decoder and discriminator parameters and \( \lambda_{1:4} \) are empirically determined loss weights.

### 3.5. Inference

At inference time, we use our transformer as a generative model to sample multiple procedure plans, \( \tilde{\pi}^k = \tilde{a}_{1:T}^k \), for the same input start and goal observations. This operation is achieved by drawing \( K \) latent noise vectors, \( z^k \), and feeding them through our transformer, \( T \), conditioned on a single start-goal observation, as follows

\[ \tilde{\pi}^k = h_a(T(Q^{z^k}, M)), \quad z^k \sim \mathcal{N}(0, 1), \tag{9} \]

for \( k = 1, \ldots, K \).

To obtain a probability distribution over actions at each timestep, \( t \), of the plan, given by our model, we calculate action frequencies as follows:

\[ \bar{\Pi} = \bar{a}_{1:T} = \frac{1}{K} \sum_{k=1}^{K} [\tilde{a}_1^k, \ldots, \tilde{a}_T^k]. \tag{10} \]

Given that \( a_t^k \) are one-hot vectors, each \( \bar{a}_t \) results in a marginal distribution over actions at a specific timestep, \( t \).

Most standard benchmark metrics for procedure planning, such as Success Rate, Accuracy or Intersection over Union (IoU), require a single action sequence output for evaluation (see Section 4.1). To compute the most probable action sequence, induced by our action distribution, \( \bar{\Pi} \), we use the Viterbi algorithm [48], as commonly seen in sequential labeling work [27,28,41,42]. More specifically, we use \( \bar{\Pi} \) as the emission matrix in the Viterbi formulation, while the transition matrix is estimated from action co-occurrence frequencies in the training set (details in the supplement). Our Viterbi post-processing step can be viewed as biasing sample selections from \( \{ \tilde{\pi}^k \}_{k=1}^{K} \), toward plans that are more likely under a first-order model of action transitions. An alternative approach to select a likely action sequence is simply to select the mode from the set \( \{ \tilde{\pi}^k \}_{k=1}^{K} \). We empirically demonstrate the superiority of the Viterbi approach, which proved especially useful for smaller datasets.

### 3.6. Implementation details

Our planner operates on video and language features, pre-extracted by a model trained for joint video-text embedding [36] using the HowTo100M [37] dataset and self-supervision. We use a memory-augmented transformer with two layers and eight heads, and optimize it for 200 epochs with ADAM [24] on a single V100 GPU. Additional training and architecture details are provided in the supplement.

### 4. Experiments

In this section, we evaluate the role of each module in our approach (Sec. 4.2) and demonstrate its performance across three different datasets. We include evaluation on the largest labeled instructional video dataset, which has not been used previously for the task of procedure planning due to the need of strong supervision in previous work (Sec. 4.3). Finally, we provide prediction uncertainty evaluation in procedure planning for the first time, which sheds light on our approach and the task of planning itself (Sec. 4.4).

#### 4.1. Evaluation protocol

**Datasets.** For evaluation, we use three different instructional video datasets, namely, CrossTask [54], the Narrated Instructional Videos datasets [2] (NIV) and COIN [46]. CrossTask contains 2750 videos, depicting 18 different procedure and an average of 7.6 actions/video; the NIV dataset
Table 1. Performance of our model trained with different loss functions on three datasets. The last row of each block represents the results of the total loss but without the Viterbi algorithm.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Loss Objective</th>
<th>SR †</th>
<th>mAcc †</th>
<th>mIoU †</th>
</tr>
</thead>
<tbody>
<tr>
<td>CrossTask</td>
<td>(L_a)</td>
<td>16.90</td>
<td>44.20</td>
<td>57.56</td>
</tr>
<tr>
<td></td>
<td>(L_a + L_l)</td>
<td>22.12</td>
<td>45.57</td>
<td>67.40</td>
</tr>
<tr>
<td></td>
<td>(L_a + L_s + L_{adv})</td>
<td>23.34</td>
<td>49.96</td>
<td>73.89</td>
</tr>
<tr>
<td></td>
<td>w/o Viterbi</td>
<td>22.66</td>
<td>45.95</td>
<td>67.52</td>
</tr>
<tr>
<td>COIN</td>
<td>(L_a)</td>
<td>8.48</td>
<td>12.19</td>
<td>68.15</td>
</tr>
<tr>
<td></td>
<td>(L_a + L_l)</td>
<td>14.41</td>
<td>20.25</td>
<td>73.49</td>
</tr>
<tr>
<td></td>
<td>(L_a + L_s + L_{adv})</td>
<td>15.40</td>
<td>21.67</td>
<td>76.31</td>
</tr>
<tr>
<td></td>
<td>w/o Viterbi</td>
<td>14.18</td>
<td>21.01</td>
<td>75.62</td>
</tr>
<tr>
<td>NIV</td>
<td>(L_a)</td>
<td>17.81</td>
<td>42.35</td>
<td>69.42</td>
</tr>
<tr>
<td></td>
<td>(L_a + L_l)</td>
<td>24.05</td>
<td>46.67</td>
<td>73.89</td>
</tr>
<tr>
<td></td>
<td>(L_a + L_s + L_{adv})</td>
<td>24.68</td>
<td>49.01</td>
<td>74.29</td>
</tr>
<tr>
<td></td>
<td>w/o Viterbi</td>
<td>20.18</td>
<td>47.73</td>
<td>73.09</td>
</tr>
</tbody>
</table>

Table 2. Ablation study on the impact of external memory sizes for prediction horizon, \(T = 3\), with CrossTask. All results are obtained using our transformer, with two layers and eight heads.

<table>
<thead>
<tr>
<th>Memory Size</th>
<th>SR †</th>
<th>mAcc †</th>
<th>mIoU †</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7.49</td>
<td>22.76</td>
<td>31.33</td>
</tr>
<tr>
<td>64</td>
<td>16.30</td>
<td>43.62</td>
<td>55.66</td>
</tr>
<tr>
<td>128</td>
<td>23.34</td>
<td>49.96</td>
<td>73.89</td>
</tr>
<tr>
<td>256</td>
<td>20.81</td>
<td>44.61</td>
<td>59.70</td>
</tr>
</tbody>
</table>

4.2. Ablation study

Impact of different loss functions. We evaluate the role of each loss component by gradually introducing each objective. The results in Table 1 show the pivotal role of language-based supervision, as evidenced by increased performance across all metrics, and the complementarity of the three objectives. Notably, improvements from the adversarial loss may seem marginal (e.g., ∼1% in SR), as the metrics only compare a single prediction to a single ground truth plan. We show its strict superiority to deterministic models for modelling distributions of procedures in Sec. 4.4.

Impact of Viterbi post-processing. Throughout the empirical results, we use the Viterbi algorithm on top of the predicted action probabilities, \(\hat{\Pi} (10)\), to produce the optimal plan at inference time. Notably, the Viterbi post-processing is optional, and one may directly use the set \(\hat{\pi}^k\) to produce the final plan by simply selecting the mode across the set. Comparing the last two rows in each block of Table 1 shows the added advantage of using Viterbi to model the optimal action order in procedure plans explicitly, for all datasets. Notably, Viterbi post-processing is especially helpful on the NIV dataset. We hypothesize that the data scarcity in NIV results in a weaker predictive model at training; therefore, explicitly modeling the optimal transition between the actions with Viterbi plays a more important role in this case.

Impact of model configuration. We also include an ablation on the adoption memory augmented transformer decoder. Table 2 shows that the size of our memory plays a key role in our architecture. Indeed, excluding the memory component yields the worst results, while too large a memory degrades performance. These results suggest that the memory component helps capture dataset content, where it yields stronger results when the number of memory entries is large enough to properly span the actions present in the entire dataset. Notably, while tuning the memory size for each dataset might yield better results, for simplicity, we elect to use the best setting of CrossTask for all datasets.

4.3. Comparison to alternative approaches

CrossTask (short-horizon). Table 3 compares our weakly-supervised approach to a number of alternatives, including the fully supervised state of the art, across the two prediction horizons typically reported in this task. Our results are
CrossTask that extends to longer prediction horizon,

Table 4. Success Rate evaluation of procedure planning results on CrossTask [54] that extends to longer prediction horizon $T \in \{3, 4\}$. The column name Supervision denotes the type of state supervision applied in training, with V and L denoting visual and language state representation, resp.

<table>
<thead>
<tr>
<th>Models</th>
<th>Supervision</th>
<th>$T = 3$</th>
<th>$T = 4$</th>
<th>$T = 5$</th>
<th>$T = 6$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SR↑</td>
<td>SR↑</td>
<td>SR↑</td>
<td>SR↑</td>
</tr>
<tr>
<td>Retrieval-Based</td>
<td>-</td>
<td>&lt;0.01</td>
<td>0.94</td>
<td>1.66</td>
<td>&lt;0.01</td>
</tr>
<tr>
<td>DDN [8]</td>
<td>-</td>
<td>8.05</td>
<td>20.21</td>
<td>30.87</td>
<td>0.98</td>
</tr>
<tr>
<td>Ours (Protocol 1)</td>
<td>V</td>
<td>12.18</td>
<td>21.64</td>
<td>31.70</td>
<td>0.77</td>
</tr>
<tr>
<td>Ours (Protocol 2)</td>
<td>L</td>
<td>24.34</td>
<td>49.96</td>
<td>73.89</td>
<td>13.40</td>
</tr>
</tbody>
</table>

Table 3. Evaluation of procedure planning results on CrossTask for prediction horizon $T \in \{3, 4\}$. The column name Supervision denotes the type of state supervision applied in training, with V and L denoting visual and language state representation, resp.

Consistently better, except for the the success rate (SR) at $T = 4$, where we are the second best approach. The performance improvement in SR is especially striking at shorter-term horizon, $T = 3$, where we outperform the previous best (i.e., Ext-GAIL [6]) by more than 2%, while using weaker supervision. Notably, Ext-GAIL achieves its level of performance via data augmentation, which allows it to have 30% more training data. In a more similar setup (i.e., when Ext-GAIL does not use data augmentation) the performance gain of our method over “Ext-GAIL w/o Aug” is 5.3%. Importantly, our results are obtained with weaker supervision, which speaks decisively in favor of our approach.

We also notice a larger gain in mIoU compared to previous work, i.e., 73.89% vs. 61.70% for $T = 3$ and 70.01% vs. 60.93% for $T = 4$. This result suggests that our approach is better at capturing feasible action steps than other approaches (e.g., never producing pour water when input observations are related to making a salad). We hypothesize that this performance is enabled by the language-based contrastive learning, which is more effective at clustering latent representations than its vision counterpart. For example, while some visual observations can look similar (e.g., pour water and add oil), the distinction between the two is clearer in natural language. Notably, the level of improvements in mIoU and mAcc is typically higher than the gain in SR. We attribute this result to the uncertainty inherent to the task (i.e., multiple feasible plans for the same start and goal observations). We explore this aspect in greater detail in Sec. 4.4. Finally, thanks to the non-autoregressive nature, we are 4x faster at inference, e.g., 6.75ms (ours) vs.

27.34ms (DDN [8]) on CrossTask for $T = 3$.

CrossTask (long-horizon). We now evaluate our model’s ability to predict plans for longer time horizons (i.e., $T \in \{3, \ldots, 6\}$). We compare to previous approaches that reported results on such horizons. There are two different protocols for these settings. (i) Protocol 1 [8] and (ii) Protocol 2 [45]; see details in supplement. For a fair comparison, we present our results using both protocols in Table 4 and show that our approach is the most effective on both.

NIV. Following previous work [6], we also evaluate our model on the smaller NIV dataset. The results in Table 5 are consistent with our results on CrossTask. Once again, our approach is the top performer across all metrics. This result suggests our language supervision works for smaller datasets as well.

COIN. To show the capability of our method to scale, we evaluate our method on the largest labeled instructional video dataset (i.e., COIN). As we are the first ones to perform procedure planning on such a large-scale dataset, there is a lack of comparison methods. Therefore, we follow existing work [8] and include three baselines: (i) Random selection; (ii) Retrieval-based; (iii) and a re-implementation of the DDN model [8], using our video features, described in Sec. 3.6. Table 5 shows that our model equipped with language supervision consistently outperforms the baselines, even strongly-supervised ones.
Our approach is probabilistic by design as described in Secs. 3.3 and 3.5. To establish a deterministic baseline, we train our model without the adversarial loss and fix the latent noise vector, $z = 0$, both during training and testing. To build the ground truth distribution over goal-conditioned plans, we retrieve all action sequences of length $T$ in the test set that share the given start and goal state. The plan distribution of our probabilistic model (conditioned on the start and goal observations) is obtained by sampling $K = 1500$ different action sequences, as discussed in Sec. 3.5. For the deterministic baseline, the model produces only one plan (i.e., $K = 1$). In all cases, the probability of a plan is defined as its frequency in the obtained sample set. To evaluate the quality of the predicted plans, we measure the (dis-)similarity between plan distributions produced by each model (i.e., ours vs. the deterministic variant) and that of the ground truth using the KL divergence and NLL. Table 6 shows that our probabilistic approach better matches the ground truth plan distribution (i.e., it has lower KL and NLL). These results come about because our model is able to sample multiple valid plans with respect to the test set distribution, as opposed to the deterministic model that considers a single feasible plan.

**Sample diversity and mode coverage.** For given start and goal observations, our approach produces multiple plan hypotheses, using probabilistic sampling described in Sec. 3.5 (as visualized in Figure 4). In this section, we measure the diversity of our samples and their relation to the ground truth plans distribution. To characterize the ground truth distribution, we define ground truth modes as the set of unique action sequences in the test set that share the same start and goal state. To calculate the relation of our samples to the ground truth modes we define two metrics, Mode Recall (ModeRec) and Mode Precision (ModePrec). ModeRec reflects how well the GT modes are covered by our model and is calculated as the average number of GT modes captured by at least one sample from our model. In comparison, ModePrec measures how often a sampled plan is captured by at least one sample from our model. In all cases, the probability of a plan is defined as its frequency in the obtained sample set. To evaluate the quality of the predicted plans, we measure the (dis-)similarity between plan distributions produced by each model (i.e., ours vs. the deterministic variant) and that of the ground truth using the KL divergence and NLL. Table 6 shows that our probabilistic approach better matches the ground truth plan distribution (i.e., it has lower KL and NLL). These results come about because our model is able to sample multiple valid plans with respect to the test set distribution, as opposed to the deterministic model that considers a single feasible plan.

**Plan distribution modeling.** Our approach is probabilistic by design as described in Secs. 3.3 and 3.5. To establish a deterministic baseline, we train our model without the adversarial loss and fix the latent noise vector, $z = 0$, both during training and testing. To build the ground truth distribution over goal-conditioned plans, we retrieve all action sequences of length $T$ in the test set that share the given start and goal state. The plan distribution of our probabilistic model (conditioned on the start and goal observations) is obtained by sampling $K = 1500$ different action sequences, as discussed in Sec. 3.5. For the deterministic baseline, the model produces only one plan (i.e., $K = 1$). In all cases, the probability of a plan is defined as its frequency in the obtained sample set. To evaluate the quality of the predicted plans, we measure the (dis-)similarity between plan distributions produced by each model (i.e., ours vs. the deterministic variant) and that of the ground truth using the KL divergence and NLL. Table 6 shows that our probabilistic approach better matches the ground truth plan distribution (i.e., it has lower KL and NLL). These results come about because our model is able to sample multiple valid plans with respect to the test set distribution, as opposed to the deterministic model that considers a single feasible plan.

**4.4. Evaluating probabilistic modeling**

To evaluate our probabilistic modeling, we compare plan distributions produced by our model, with the ground truth distribution over feasible plans. We focus our evaluation of probabilistic modeling on CrossTask as it is the most suitable dataset in terms of variations in the set of feasible plans, as we demonstrate in the supplement.

**Plan distribution modeling.** Our approach is probabilistic by design as described in Secs. 3.3 and 3.5. To establish a deterministic baseline, we train our model without the adversarial loss and fix the latent noise vector, $z = 0$, both during training and testing. To build the ground truth distribution over goal-conditioned plans, we retrieve all action sequences of length $T$ in the test set that share the given start and goal state. The plan distribution of our probabilistic model (conditioned on the start and goal observations) is obtained by sampling $K = 1500$ different action sequences, as discussed in Sec. 3.5. For the deterministic baseline, the model produces only one plan (i.e., $K = 1$). In all cases, the probability of a plan is defined as its frequency in the obtained sample set. To evaluate the quality of the predicted plans, we measure the (dis-)similarity between plan distributions produced by each model (i.e., ours vs. the deterministic variant) and that of the ground truth using the KL divergence and NLL. Table 6 shows that our probabilistic approach better matches the ground truth plan distribution (i.e., it has lower KL and NLL). These results come about because our model is able to sample multiple valid plans with respect to the test set distribution, as opposed to the deterministic model that considers a single feasible plan.

**Sample diversity and mode coverage.** For given start and goal observations, our approach produces multiple plan hypotheses, using probabilistic sampling described in Sec. 3.5 (as visualized in Figure 4). In this section, we measure the diversity of our samples and their relation to the ground truth plans distribution. To characterize the ground truth distribution, we define ground truth modes as the set of unique action sequences in the test set that share the same start and goal state. To calculate the relation of our samples to the ground truth modes we define two metrics, Mode Recall (ModeRec) and Mode Precision (ModePrec). ModeRec reflects how well the GT modes are covered by our model and is calculated as the average number of GT modes captured by at least one sample from our model. In comparison, ModePrec measures how often a sampled plan is captured by at least one sample from our model. In all cases, the probability of a plan is defined as its frequency in the obtained sample set. To evaluate the quality of the predicted plans, we measure the (dis-)similarity between plan distributions produced by each model (i.e., ours vs. the deterministic variant) and that of the ground truth using the KL divergence and NLL. Table 6 shows that our probabilistic approach better matches the ground truth plan distribution (i.e., it has lower KL and NLL). These results come about because our model is able to sample multiple valid plans with respect to the test set distribution, as opposed to the deterministic model that considers a single feasible plan.

**5. Conclusion**

We have introduced a weakly-supervised method for probabilistic procedure planning using instructional videos. Different from previous work, we eschew the need for expensive visual supervision in favor of cheaper language supervision by capitalizing on pre-trained text-video embeddings, which, remarkably, leads to superior planning performance. We showed that modeling the interplay between intermediate visual states and actions step-by-step is not a necessity for procedure planning. Instead, we efficiently solve the problem with a “one-shot” transformer decoder architecture. In addition, we demonstrated the crucial role of modeling uncertainty in obtained plans to yield a principled approach to planning from videos. We introduced a way to evaluate such uncertainty on the test set and show that it is a powerful metric to better understand the model and the task of planning itself. Hopefully, future work will adopt the probabilistic view on procedure planning from instructional videos, not only in training but also in evaluation, such that the next generation of planners can confidently predict multiple feasible plans to achieve the desired goal.
References


2946


