PDPP: Projected Diffusion for Procedure Planning in Instructional Videos

Hanlin Wang[1] Yilu Wu[1] Sheng Guo[3] Limin Wang[1, 2, ✉]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, China
[2]Shanghai AI Lab, China [3]MYbank, Ant Group, China

Abstract

In this paper, we study the problem of procedure planning in instructional videos, which aims to make goal-directed plans given the current visual observations in unstructured real-life videos. Previous works cast this problem as a sequence planning problem and leverage either heavy intermediate visual observations or natural language instructions as supervision, resulting in complex learning schemes and expensive annotation costs. In contrast, we treat this problem as a distribution fitting problem. In this sense, we model the whole intermediate action sequence distribution with a diffusion model (PDPP), and thus transform the planning problem to a sampling process from this distribution. In addition, we remove the expensive intermediate supervision, and simply use task labels from instructional videos as supervision instead. Our model is a U-Net based diffusion model, which directly samples action sequences from the learned distribution with the given start and end observations. Furthermore, we apply an efficient projection method to provide accurate conditional guides for our model during the learning and sampling process. Experiments on three datasets with different scales show that our PDPP model can achieve the state-of-the-art performance on multiple metrics, even without the task supervision. Code and trained models are available at https://github.com/MCG-NJU/PDPP.

1. Introduction

Instructional videos [1,31,38] are strong knowledge carriers, which contain rich scene changes and various actions. People watching these videos can learn new skills by figuring out what actions should be performed to achieve the desired goals. Although this seems to be natural for humans, it is quite challenging for AI agents. Training a model that can learn how to make action plans to transform from the start state to goal is crucial for the next-generation AI system as such a model can analyze complex human behaviours and help people with goal-directed problems like cooking or repairing items. Nowadays the computer vision community is paying growing attention to the instructional video understanding [4, 8, 9, 24, 37]. Among them, Chang *et al.* [4] proposed a problem named as procedure planning in instructional videos, which requires a model to produce goal-directed action plans given the current visual observation of the world. Different with traditional procedure planning problem in structured environments [12, 29], this task deals with unstructured environments and thus forces the model to learn structured and plannable representations in real-life videos. We follow this work and tackle the procedure planning problem in instructional videos. Specifically, given the visual observations at start and end time, we need to produce a sequence of actions which transform the environment from start state to the goal state, as shown in Fig. 1.

✉: Corresponding author (lmwang@nju.edu.cn).

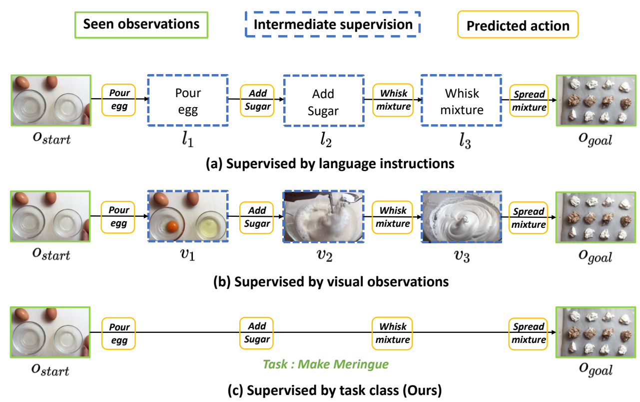

Figure 1. Procedure planning example. Given a start observation o_{start} and a goal state o_{goal}, the model is required to generate a sequence of actions that can transform o_{start} to o_{goal}. Previous approaches rely on heavy intermediate supervision during training, while our model only needs the task class labels (bottom row).

Previous approaches for procedure planning in instructional videos often treat it as a sequence planning problem and focus on predicting each action accurately. Most works rely on a two-branch autoregressive method to predict the intermediate states and actions step by step [2,4,30]. Such models are complex and easy to accumulate errors during the planning process, especially for long sequences.

Recently, Zhao *et al*. [36] proposed a single branch non-autoregressive model based on transformer [33] to predict all intermediate steps in parallel. To obtain a good performance, they used a learnable memory bank in the transformer decoder, augmented their model with an extra generative adversarial framework [13] and applied a Viterbi post-processing method [34]. This method brought multiple learning objectives, complex training schemes and tedious inference process. Instead, we assume procedure planning as a distribution fitting problem and planning is solved with a sampling process. We aim to directly model the joint distribution of the whole action sequence in instructional video rather than every discrete action. In this perspective, we can use a simple MSE loss to optimize our generative model and generate action sequence plans in one shot with a sampling process, which results in less learning objectives and simpler training schemes.

For supervision in training, in addition to the action sequence, previous methods often require heavy intermediate visual [2, 4, 30] or language [36] annotations for their learning process. In contrast, we only use task labels from instructional videos as a condition for our learning (as shown in Fig. 1), which could be easily obtained from the keywords or captions of videos and requires much less labeling cost. Another reason is that task information is closely related to the action sequences in a video. For example, in a video of $jacking\ up\ a\ car$, the possibility for action $add\ sugar$ appears in this process is nearly zero.

Modeling the uncertainty in procedure planning is also an important factor that we need to consider. That is, there might be more than one reasonable plan sequences to transform from the given start state to goal state. For example, change the order of $add\ sugar$ and $add\ butter$ in $making\ cake$ process will not affect the final result. So action sequences can vary even with the same start and goal states. To address this problem, we consider adding randomness to our distribution-fitting process and perform training with a diffusion model [18, 26]. Solving procedure planning problem with a diffusion model has two main benefits. First, a diffusion model changes the goal distribution to a random Gaussian noise by adding noise slowly to the initial data and learns the sampling process at inference time as an iterative denoising procedure starting from a random Gaussian noise. So randomness is involved both for training and sampling in a diffusion model, which is helpful to model the uncertain action sequences for procedure planning. Second, it is convenient to apply conditional diffusion process with the given start and goal observations based on diffusion models, so we can model the procedure planning problem as a conditional sampling process with a simple training scheme. In this work, we concatenate conditions and action sequences together and propose a projected diffusion model to perform conditional diffusion process.

Contributions. To sum up, the main contributions of this work are as follows: a) We cast the procedure planning as a conditional distribution-fitting problem and model the joint distribution of the whole intermediate action sequence as our learning objective, which can be learned with a simple training scheme. b) We introduce an efficient approach for training the procedure planner, which removes the supervision of visual or language features and relies on task supervision instead. c) We propose a novel projected diffusion model (PDPP) to learn the distribution of action sequences and produce all intermediate steps at one shot. We evaluate our PDPP on three instructional videos datasets and achieve the state-of-the-art performance across different prediction time horizons. Note that our model can still achieve excellent results even if we remove the task supervision and use the action labels only.

2. Related work

Procedural video understanding. The problem of procedural video understanding has gained more and more attention with an aim to learn the inter-relationship between different events in videos recently. Zhao *et al*. [37] investigated the problem of abductive visual reasoning, which requires vision systems to infer the most plausible visual explanation for the given visual observations. Furthermore, Liang *et al*. [24] proposed a new task: given an incomplete set of visual events, AI agents are asked to generate descriptions not only for the visible events and but also for the lost events by logical inferring. Unlike these works trying to learn the abductive information of intermediate events, Chang *et al*. [4] introduced procedure planning in instructional videos which requires AI systems to plan an action sequence that can transform the given start observation to the goal state. In this paper, we follow this work and study the procedural video understanding problem by learning goal-directed actions planning.

Diffusion probabilistic models. Nowadays, diffusion probabilistic models [28] have achieved great success in many research areas. Ho *et al*. [18] used a reweighted objective to train diffusion model and achieved great synthesis quality for image synthesis problem. Janner *et al*. [21] studied the trajectory planning problem with diffusion model and get remarkable results. Besides, diffusion models are also used in video generation [17, 20], density estimation [22], human motion [32], sound generation [35], text generation [23] and many other domains, all achieved competitive results. In this work, we apply diffusion process to procedure planning in instructional videos and propose our projected diffusion model, which achieves state-of-the-art performance only with a simple learning scheme.

Projected gradient descent. Projected gradient descent is an optimal solution suitable for constrained optimization problems, which is proven to be effective in optimization

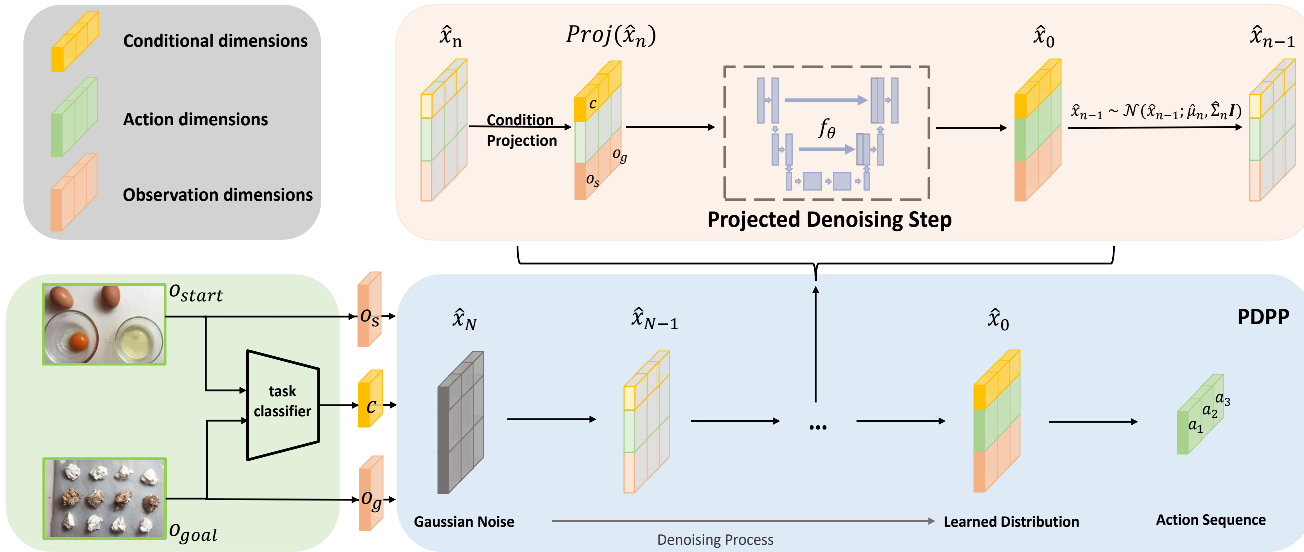

Figure 2. Overview of our projected diffusion model (prediction horizon T = 3). We first train a task classifier to generate conditional information c, which will be used as guidance along with the given observations o_s and o_g. Then we compute the denoising process iteratively. In each step, we first conduct a condition projection to the input, then predict the initial distribution by the learned model f_θ. After that we calculate $\hat{x}_{n-1}$ with the predicted $\hat{x}_0$. We finally select the action dimensions as our result after N denoising steps.

with rank constraints [5], online power system optimization problems [14] and adversarial attack [6]. The core idea of projected gradient descent is to add a projection operation to the normal gradient descent method, so that the result is ensured to be constrained in the feasible region. Inspired by this, we add a similar projection operation to our diffusion process, which keeps the conditional information for diffusion unchangeable and thus provides accurate guides for learning.

3. Method

In this section, we present the details of our projected diffusion model for procedure planning (PDPP). We will first introduce the setup for this problem in Sec. 3.1. Then we present the diffusion model used to model the action sequence distribution in Sec. 3.2. To provide more precise conditional guidance both for the training and sampling process, a simple projection method is applied to our model, which we will discuss in Sec. 3.3. Finally, we show the training scheme (Sec. 3.4) and sampling process (Sec. 3.5) of our PDPP. An overview of PDPP is provided in Fig. 2.

3.1. Problem formulation

We follow the problem set-up of Chang *et al.* [4]: given a start visual observation o_s and a visual goal o_g, a model is required to plan a sequence of actions $a_{1:T}$ so that the environment state can be transformed from o_s to o_g. Here T is the horizon of planning, which denotes the number of action steps for the model to take and $\{o_s, o_g\}$ indicates two different environment states in an instructional video.

We decompose the procedure planning problem into two sub-problems, as shown in Eq. (1). The first problem is to learn the task-related information c with the given $\{o_s, o_g\}$ pair. This can be seen as a preliminary inference for procedure planning. Then the second problem is to generate action sequences with the task-related information and given observations. Note that Jing *et al.* [2] also decompose the procedure planning problem into two sub-problems, but their purpose of the first sub-problem is to provide long-horizon information for the second stage since Jing *et al.* [2] plans actions step by step, while our purpose is to get condition for sampling to achieve an easier learning.

$$p(a_{1:T}|o_s, o_g) = \int p(a_{1:T}|o_s, o_g, c)p(c|o_s, o_g)dc. \quad (1)$$

At training time, we first train a simple model (implemented as multi-layer perceptrons (MLPs)) with the given observations $\{o_s, o_g\}$ to predict which the task category is. We use the task labels in instructional videos $\bar{c}$ to supervise the output c. After that, we evaluate $p(a_{1:T}|o_s, o_g, c)$ in parallel with our model and leverage the ground truth (GT) intermediate action labels as supervision for training. Compared with the visual and language supervision in previous works, task label supervision is easier to get and brings simpler learning schemes. At inference phase, we just use the start and goal observations to predict the task class information c and then samples action sequences $a_{1:T}$ from the learned distribution with the given observations and predicted c, where T is the planning horizon.

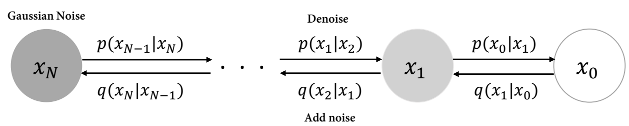

Figure 3. Schematic diagram for the forward and reverse diffusion processes.

3.2. Projected diffusion for procedure planning

Our method consists of two stages: task class prediction and action sequence distribution modeling. The first stage learning is a traditional classification problem which we implement with a simple MLP model. The main part of our model is the second one. That is, how to model $p(a_{1:T}|o_s, o_g, c)$ to solve the procedure planning problem. Jing *et al*. [2] assume this as a Goal-conditioned Markov Decision Process and use a policy $p(a_t|o_t)$ along with a transition model $\tau_\mu(o_t|c, o_{t-1}, a_{t-1})$ to perform the planning step by step, which is complex to train and slow for inference. We instead treat this as a direct distribution fitting problem with a diffusion model.

Diffusion model. A diffusion model [18,26] solves the data generation problem by modeling the data distribution $p(x_0)$ as a denoising Markov chain over variables $\{x_N...x_0\}$ and assume x_N to be a random Gaussian distribution. The forward process of a diffusion model is incrementally adding Gaussian noise to the initial data x_0 and can be represented as $q(x_n|x_{n-1})$, by which we can get all intermediate noisy latent variables $x_{1:N}$ with a diffusion step N. In the sampling stage, the diffusion model conducts iterative denoising procedure $p(x_{n-1}|x_n)$ for N times to approximate samples from the target data distribution. The forward and reverse diffusion processes are shown in Fig. 3.

In a standard diffusion model, the ratio of Gaussian noise added to the data at diffusion step n is pre-defined as $\{\beta_n \in (0,1)\}_{n=1}^N$. Each adding noise step can be parametrized as

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

Since hyper-parameters $\{\beta_n\}_{n=1}^N$ are pre-defined, there is no training in the noise-adding process. As discussed in [18], re-parameterize Eq. (2) we can get:

$$x_n = \sqrt{\overline{\alpha}_n}x_0 + \sqrt{1-\overline{\alpha}_n}\epsilon, \tag{3}$$

where $\overline{\alpha}_n = \prod_{s=1}^n(1-\beta_s)$ and $\epsilon \sim \mathcal{N}(0, I)$.

In the denoising process, each step is parametrized as:

$$p_\theta(x_{n-1}|x_n) = \mathcal{N}(x_{n-1}; \mu_\theta(x_n, n), \Sigma_\theta(x_n, n)), \tag{4}$$

where μ_θ is produced by a learnable model and Σ_θ can be directly calculated with $\{\beta_n\}_{n=1}^N$ [18]. The learning objective for a typical diffusion model in [18] is the noise added to the uncorrupted data x_0 at each step. When training, the diffusion model first selects a diffusion step $n \in [1, N]$ and calculates x_n as shown in Eq. (3). Then the learnable model will compute $\epsilon_\theta(x_n, n)$ and calculate loss with the true noise add to the distribution at step n. After training, the diffusion model can simply generate data like x_0 by iteratively processing the denoising step starting from a random Gaussian noise.

However, applying this kind of diffusion model to procedure planning directly is not suitable. First, the sampling process in our task is condition-guided while no condition is applied in the standard diffusion model. Second, the distribution we want to fit is the whole action sequence, which has a strong semantic information. Directly predicting the semantically meaningless noise sampled from random Gaussian distribution can be hard. In experiments which take noise as the predicting objective, our model just fails to be trained. To address these problems, we make two modifications to the standard diffusion model: one is about input and the other is the learning objective of model.

Conditional action sequence input. The input of a standard diffusion model is the data distribution it needs to fit and no guided information is required. For the procedure planning problem, the distribution we aim to fit is the intermediate action sequences $[a_1, a_2...a_T]$, which depends on the given observations and task class we get in the first learning stage. Thus we need to find how to add these guided conditions into the diffusion process. Although there are multiple guided diffusion models presented [7,19], they are expensive for learning and need complex training or sampling strategy. Inspired by Janner *et al*. [21], we here apply a simple way to achieve our goal: just treat these conditions as additional information and concatenate them along the action feature dimension. Notably, we concatenate o_s and a_1 together, same for o_g and a_T. In this way, we introduce a prior knowledge that the start/end observations are more related to the first/last actions, which turns out to be useful for our learning (details are provided in the supplementary material). Thus our model input for training now can be represented as a multi-dimension array:

$$\begin{bmatrix} c & c & & c & c \\ a_1 & a_2 & ... & a_{T-1} & a_T \\ o_s & 0 & & 0 & o_g \end{bmatrix}. \tag{5}$$

Each column in our model input represents the condition information, action one-hot code, and the corresponding observation for a certain action. Note that we do not need the intermediate visual observations as supervision, so all observation dimensions are set to zero except for the start and end observations.

Learning objective of our diffusion model. As mentioned above, the learning objective of a standard diffusion model is the random noise added to the distribution at each diffusion step. This learning scheme has demonstrated great success in data synthesis area, partly because predicting noise rather than the initial input x_0 brings more variations for data generation. For procedure planning problem, however, the distribution we need to fit contains high-level features rather than pixels. Since the planning horizon T is relatively short and the one-hot action features require less randomness than in data generation, predicting noise in diffusion process for procedure planning will just increase the difficulty of learning. So we modify the learning objective to the initial input x_0, which will be described in Sec. 3.4.

3.3. Condition projection during learning

Our model transforms a random Gaussian noise to the final result, which has the same structure with our model input (see Eq. (5)) by conducting N denoising steps. Since we combine the conditional information with action sequences as the data distribution, these conditional guides can be changed during the denoising process. However, the change of these conditions will bring wrong guidance for the learning process and make the observations and conditions useless. To address this problem, we add a condition projection operation into the learning process. That is, we force the observation and condition dimensions not to be changed during training and inference by assigning the initial value. The input x of condition projection is either a noise-add data (Alg.1 L5) or the predicted result of model (Alg.1 L7). We use $\{\hat{c}, \hat{a}, \hat{o}\}$ to represent different dimensions in x, then our projection operation $\mathrm{Proj}()$ can be denoted as:

$$\underbrace{\begin{bmatrix} \hat{c}_1 & \hat{c}_2 & & \hat{c}_T \\ \hat{a}_1 & \hat{a}_2 & \dots & \hat{a}_T \\ \hat{o}_1 & \hat{o}_2 & & \hat{o}_T \end{bmatrix}}_{x} \rightarrow \underbrace{\begin{bmatrix} c & c & & c \\ \hat{a}_1 & \hat{a}_2 & \dots & \hat{a}_T \\ o_s & 0 & & o_g \end{bmatrix}}_{\mathrm{Proj}(x)}, \tag{6}$$

where $\hat{c}_i$, $\hat{o}_i$ and $\hat{a}_i$ denote the i^{th} horizon class, observation dimensions and predicted action logits in x, respectively. c, o_s, o_g are the conditions.

3.4. Training scheme

Our training scheme contains two stages: a) training a task-classifier model $\mathcal{T}_\phi(c|o_s, o_g)$ that extracts conditional guidance from the given start and goal observations; b) leveraging the projected diffusion model to fit the target action sequence distribution.

For the first stage, we apply MLP models to predict the task class c with the given o_s, o_g. Ground truth task class labels $\overline{c}$ are used as supervision. In the second learning stage, we follow the basic training scheme for diffusion model, but change the learning objective as the initial input x_0. We use a U-Net model [27] $f_\theta(x_n, n)$ as the learnable model and our training loss is:

$$\mathcal{L}_{\text{diff}} = \sum_{n=1}^{N} (f_\theta(x_n, n) - x_0)^2, \tag{7}$$

Algorithm 1 Training

Input Initial input x_0, total diffusion steps number N, model f_θ, $\{\overline{\alpha}_n\}_{n=1}^N$, weight matrix w

1: **repeat**
2: $\quad n \sim Uniform(\{1, ..., N\})$
3: $\quad \epsilon \sim \mathcal{N}(0, I)$
4: $\quad x_n = \sqrt{\overline{\alpha}_n} x_0 + \sqrt{1 - \overline{\alpha}_n} \epsilon$
5: $\quad \hat{x}_0 = f_\theta(Proj(x_n), n)$
6: $\quad$ Take gradient descent step on
7: $\quad\quad \bigtriangledown_\theta \, ||(x_0 - Proj(\hat{x}_0)) * w||^2$
8: **until** converged

We believe that a_1 and a_T are more important because they are the most related actions for the given observations. Thus we rewrite Eq. (7) as a weighted loss by multiplying a weight matrix to $\mathcal{L}_{\text{diff}}$ as follows:

$$\begin{bmatrix} 1 & 1 & & 1 & 1 \\ w & 1 & \dots & 1 & w \\ 1 & 1 & & 1 & 1 \end{bmatrix}. \tag{8}$$

Besides, we add a condition projection step to our diffusion process. So given the initial input x_0 which contains action sequences, task conditions and observations, we first add noise to the input to get x_n, and then apply condition projection to ensure the guidance not changed. With x_n and the corresponding diffusion step n, we calculate the denosing output $f_\theta(x_n, n)$, followed by condition projection again. Finally, we compute the weighted $\mathcal{L}_{\text{diff}}$ and update model, as shown in Algorithm 1.

3.5. Inference

At inference time, only the start observation o_s and goal observation o_g are provided. We first predict the task class by choosing the maximum probability value in the output of task-classifier model $\mathcal{T}_\phi$. Then the predicted task class c is used as the class condition. To sample from the learned action sequence distribution, we start with a Gaussian noise, and iteratively conduct denoise and condition projection for N times. The detailed inference process is shown in Algorithm 2.

Once we get the predicted output $\hat{x}_0$, we take out the action sequence dimensions $[\hat{a}_1, ..., \hat{a}_T]$ and select the index of every maximum value in $\hat{a}_i (i = 1, ..., T)$ as the action sequence plan for procedure planning. Note that for the training stage, the class condition dimensions of x_0 are the ground truth task labels, not the output of our task-classifier as in inference.

4. Experiments

In this section, we evaluate our PDPP model on three real-life datasets and show our competitive results for various planning horizons. We first present the result of our first

Algorithm 2 Inference

Input total diffusion steps number N, model f_θ, $\{\overline{\alpha}_n\}_{n=1}^N$, $\{\beta_n\}_{n=1}^N$

$\hat{x}_N \sim \mathcal{N}(0, I)$
for $n = N, ..., 1$ **do**
  $\hat{x}_0 = f_\theta(Proj(\hat{x}_n), n)$
  if $n > 1$ **then**
    $\hat{\mu}_n = \frac{\sqrt{\overline{\alpha}_{n-1}}\beta_n}{1-\overline{\alpha}_n}\hat{x}_0 + \frac{\sqrt{\alpha_n}(1-\overline{\alpha}_{n-1})}{1-\overline{\alpha}_n}\hat{x}_n$
    $\hat{\Sigma}_n = \frac{1-\overline{\alpha}_{n-1}}{1-\overline{\alpha}_n} \cdot \beta_n$
    $\hat{x}_{n-1} \sim \mathcal{N}(\hat{x}_{n-1}; \hat{\mu}_n, \hat{\Sigma}_n\mathbf{I})$
  end if
end for
return $\hat{x}_0$

training stage, which predicts the task class with the given observations in Sec. 4.3. Then we compare our performance with other alternative approaches on the three datasets and demonstrate the effectiveness of our model in Sec. 4.4. We also study the role of task-supervision for our model in Sec. 4.5. Finally, we show our prediction uncertainty evaluation results in Sec. 4.6.

4.1. Implementation details

We use the basic U-Net [27] as our learnable model for projection diffusion, in which a modification is made by using a convolution operation along the planning horizon dimension for downsampling rather than max-pooling as in [21]. For training, we use the linear warm-up training scheme to optimize our model. We train different steps for the three datasets, corresponding to their scales. More details about training and model architecture are provided in the supplementary material.

4.2. Evaluation protocol

Datasets. We evaluate our PDPP model on three instructional video datasets: **CrossTask** [38], **NIV** [1], and **COIN** [31]. CrossTask contains 2,750 videos from 18 different tasks, with an average of 7.6 actions per video. The NIV dataset consists of 150 videos about 5 daily tasks, which has 9.5 actions in one video on average. COIN is much larger with 11,827 videos, 180 different tasks and 3.6 actions/video. We randomly select 70% data for training and 30% for testing as previous work [2, 4, 36]. Note that we do not select 70%/30% for videos in each task, but in the whole dataset. Following previous work [2, 4, 36], we extract all action sequences $\{[a_i, ..., a_{i+T-1}]\}_{i=1}^{n-T+1}$ with predicting horizon T from the given video which contains n actions by sliding a window of size T. Then for each action sequence $[a_i, ..., a_{i+T-1}]$, we choose the video clip feature at the beginning time of action a_i and clip feature around the end time of a_{i+T-1} as the start observation o_s and goal state o_g, respectively. Both clips are 3 seconds long. For experiments conduct on CrossTask, we use two kinds of pre-extracted video features as the start and goal observations. One are the features provided in CrossTask dataset: each second of video content is encoded into a 3200-dimensional feature vector as a concatenation of the I3D, ResNet-152 and audio VGG features [3, 15, 16], which are also applied in [2, 4]. The other kind of features are generated by the encoder trained with the HowTo100M [25] dataset, as in [36]. For experiments on the other two datasets, we follow [36] to use the HowTo100M features for a fair comparison.

	CrossTask$_{Base}$	CrossTask$_{How}$	COIN	NIV
$T = 3$	94.38	92.43	79.42	100.00
$T = 4$	83.64	92.98	78.89	100.00
$T = 5$	83.37	93.39	-	-
$T = 6$	83.85	93.20	-	-

Table 1. Classification results on all datasets. CrossTask$_{Base}$ uses features provided by the dataset while CrossTask$_{How}$ applies features extracted by HowTo100M trained encoder.

Metrics. Following previous work [2, 4, 36], we apply three metrics to evaluate the performance. a) **Success Rate (SR)** considers a plan as a success only if every action matches the ground truth sequence. b) **mean Accuracy (mAcc)** calculates the average correctness of actions at each individual time step, which means an predicted action is considered correct if it matches the action in ground truth at the same time step. c) **mean Intersection over Union (mIoU)** measures the overlap between predicted actions and ground truth by computing Iou $\frac{|\{a_t\}\cap\{\hat{a}_t\}|}{|\{a_t\}\cup\{\hat{a}_t\}|}$, where $\{a_t\}$ is the set of ground truth actions and $\{\hat{a}_t\}$ is the set of predicted actions.

Previous approaches [2, 36] compute the mIoU metric on every mini-batch (batch size larger than one) and calculate the average as the result. This brings a problem that the mIoU value can be influenced heavily by batch size. Consider if we set batch size is equal to the size of training data, then all predicted actions can be involved in the ground truth set and thus be correct predictions. However, if batch size is set to one, then any predicted action that not appears in the corresponding ground truth action sequence will be wrong. To address this problem, we standardize the way to get mIoU as computing IoU on every single sequence and calculating the average of these IoUs as the result (equal to setting of batch size $= 1$).

Baselines. Models for procedure planning [2, 4, 36] and other fully supervised planning approaches [10, 11, 29] are all involved in our comparison. Descriptions for these methods are available in the supplementary material.

Models	Supervision	T = 3 SR↑	T = 3 mAcc↑	T = 3 mIoU↑	T = 4 SR↑	T = 4 mAcc↑	T = 4 mIoU↑
Random	-	<0.01	0.94	1.66	<0.01	0.83	1.66
Retrieval-Based	-	8.05	23.30	32.06	3.95	22.22	36.97
WLTDO [10]	-	1.87	21.64	31.70	0.77	17.92	26.43
UAAA [11]	-	2.15	20.21	30.87	0.98	19.86	27.09
UPN [29]	V	2.89	24.39	31.56	1.19	21.59	27.85
DDN [4]	V	12.18	31.29	47.48	5.97	27.10	48.46
Ext-GAILw/o Aug. [2]	V	18.01	43.86	57.16	-	-	-
Ext-GAIL [2]	V	21.27	49.46	61.70	16.41	43.05	60.93
P^3IV [36]	L	23.34	49.96	73.89	13.40	44.16	70.01
Ours$_{Base}$	C	26.47	55.35	58.95	15.40	49.42	56.99
Ours$_{How}$	C	**37.20**	**64.67**	66.57	**21.48**	**57.82**	65.13

Table 2. Evaluation results on CrossTask for procudure planning with prediction horizon $T \in \{3, 4\}$. The *Supervision* column denotes the type of supervision applied in training, where V denotes intermediate visual states, L denotes language feature and C means task class. **Note that we compute mIoU by calculating average of every IoU of a single antion sequence rather than a mini-batch**.

Models	T = 3 SR↑	T = 4 SR↑	T = 5 SR↑	T = 6 SR↑
Retrieval-Based	8.05	3.95	2.40	1.10
DDN [4]	12.18	5.97	3.10	1.20
P^3IV [36]	23.34	13.40	7.21	4.40
Ours$_{Base}$	26.47	15.40	9.37	6.76
Ours$_{How}$	**37.20**	**21.48**	**13.58**	**8.47**

Table 3. Success Rate evaluation results on CrossTask with longer planning horizons.

Horizon	Models	Sup.	NIV SR↑	NIV mAcc↑	NIV mIoU↑	COIN SR↑	COIN mAcc↑	COIN mIoU↑
T = 3	Random	-	2.21	4.07	6.09	<0.01	<0.01	2.47
	Retrieval	-	-	-	-	4.38	17.40	32.06
	DDN [4]	V	18.41	32.54	56.56	13.9	20.19	64.78
	Ext-GAIL [2]	V	22.11	42.20	65.93	-	-	-
	P^3IV [36]	L	24.68	49.01	74.29	15.4	21.67	76.31
	Ours	C	**31.25**	**49.26**	57.92	**21.33**	**45.62**	51.82
T = 4	Random	-	1.12	2.73	5.84	<0.01	<0.01	2.32
	Retrieval	-	-	-	-	2.71	14.29	36.97
	DDN [4]	V	15.97	27.09	53.84	11.13	17.71	68.06
	Ext-GAIL [2]	V	19.91	36.31	53.84	-	-	-
	P^3IV [36]	L	20.14	38.36	67.29	11.32	18.85	70.53
	Ours	C	**26.72**	**48.92**	59.04	**14.41**	**44.10**	51.39

Table 4. Evaluation results on NIV and COIN with prediction horizon $T \in \{3, 4\}$. Sup. denotes the type of supervision in training. Note that we compute IoU on every action sequence and take the mean as mIoU.

4.3. Results for task-classifier

The first stage of our learning is to predict the task class with the given start and goal observations. We implement this with MLP models and the detailed first-stage training process is described in the supplementary material. The classification results for different planning horizons on three datasets are shown in Tab. 1. We can see that our classifier can perfectly figure out the task class in the NIV dataset since only 5 tasks are involved. For larger datasets CrossTask and COIN, our model can make right predictions most of the time.

4.4. Comparison with other approaches

We follow previous work [36] and compare our approach with other alternative methods on three datasets, across multiple prediction horizons.

CrossTask (short horizon). We first evaluate on CrossTask with two prediction horizons typically used in previous work. We use Ours$_{Base}$ to denote our model with features provided by CrossTask and Ours$_{How}$ as model with features extracted by HowTo100M trained encoder. Note that we compute mIoU by calculating the mean of every IoU for a single antion sequence rather than a mini-batch as explained in Sec. 4.2, though the latter can achieve a higher mIoU value. Results in Tab. 2 show that Ours$_{Base}$ beats all methods for most metrics except for the success rate (SR) when $T = 4$, where our model is the second best, and Ours$_{How}$ just significantly outperforms all previous methods. Specifically, for using HowTo100M-extracted video features, we outperform [36] by around 14% and more than 8% for SR when $T = 3, 4$, respectively. As for features provided by CrossTask, Ours$_{Base}$ outperforms the previous best method [2] by more than 5% for SR and 6% for mAcc when $T = 3$.

CrossTask (long horizon). We further study the ability of predicting with longer horizons for our model. Following [36], we here evaluate the SR value with planning horizon $T = \{3, 4, 5, 6\}$. We present the result of our model along with other approaches that reported results for longer horizons in Tab. 3. This result shows our model can get a stable and great improvement with all planning horizons compared with the previous best model.

NIV and COIN. Tab. 4 shows our evaluation results on the other two datasets NIV and COIN, from which we can see that our approach remains to be the best performer for both datasets. Specifically, in the NIV dataset where mAcc is

Dataset		w. task sup.			w.o. task sup.		
		SR↑	mAcc↑	mIoU↑	SR↑	mAcc↑	mIoU↑
$T = 3$	CrossTask$_{Base}$	**26.47**	55.35	58.95	22.82	51.56	54.36
	CrossTask$_{How}$	**37.20**	64.67	66.57	35.69	63.91	66.04
	NIV	**31.25**	49.26	57.92	29.41	46.20	56.42
	COIN	**21.33**	45.62	51.82	16.46	36.43	43.50
$T = 4$	CrossTask$_{Base}$	**15.40**	49.42	56.99	14.91	49.55	56.28
	CrossTask$_{How}$	**21.48**	57.82	65.13	20.52	57.47	64.39
	NIV	**26.72**	48.92	59.04	26.72	46.55	59.50
	COIN	**14.41**	44.10	51.39	12.32	35.48	42.75
$T = 5$	CrossTask$_{Base}$	**9.37**	45.93	56.32	8.95	45.77	56.34
	CrossTask$_{How}$	**13.58**	54.05	65.32	12.80	53.44	64.01
$T = 6$	CrossTask$_{Base}$	**6.76**	43.61	57.51	6.06	44.15	57.07
	CrossTask$_{How}$	**8.47**	50.14	65.38	8.15	50.45	64.13

Table 5. Ablation study on the role of task supervision. The $w.\ task\ sup.$ denotes learning with task supervision and $w.o.\ task\ sup.$ means training with the basic action labels only.

relatively high, our model raises the SR value by more than 6.5% both for the two horizons and outperforms the previous best by more than 10% on mAcc metric when $T = 4$. As for the large COIN dataset where mAcc is low, our model significantly improves mAcc by more than 20%.

All the results suggest that our model performs well across datasets with different scales.

4.5. Study on task supervision

In this section, we study the role of task supervision for our model. Tab. 5 shows the results of learning with and without task supervision, which suggest that the task supervision is quite helpful to our learning. Besides, we find that task supervision helps more for learning in the COIN dataset. We assume the reason is that fitting the COIN dataset is hard to our model since the number of tasks in COIN is large. Thus the guidance of task class information is more important to COIN compared with the other two datasets. Notably, training our model without task supervision also achieves state-of-the-art performance on multiple metrics, which suggests the effective of our approach.

4.6. Evaluating probabilistic modeling

As discussed in Sec. 1, we introduce diffusion model to procedure planning to model the uncertainty in this problem. Here we follow [36] to evaluate our probabilistic modeling. We focus on CrossTask$_{How}$ as it has the most uncertainty for planning. Results on other datasets and further details are available in the supplement.

Our model is probabilistic by starting from random noises and denoising step by step. We here introduce two baselines to compare with our diffusion based approach. We first remove the diffusion process in our method to establish the *Noise* baseline, which just samples from a random noise with the given observations and task class condition in one shot. Then we further establish the *Deterministic* baseline by setting the start distribution $\hat{x}_N = 0$, thus the model

Metric↓	Model	T = 3	T=4	T=5	T=6
NLL	Deterministic	**3.57**	4.29	4.70	5.12
	Noise	3.58	4.04	4.45	4.79
	Ours	3.61	**3.85**	**3.77**	**4.06**
KL-Div	Deterministic	**2.99**	3.40	3.54	3.82
	Noise	3.00	3.15	3.30	3.49
	Ours	3.03	**2.96**	**2.62**	**2.76**

Table 6. Evaluation results of the plan distributions metrics.

Metric↑	Model	T = 3	T=4	T=5	T=6
SR	Deterministic	**39.03**	21.17	12.59	7.47
	Noise	34.92	18.99	12.04	7.82
	Ours	37.20	**21.48**	**13.58**	**8.47**
ModePrec	Deterministic	**55.60**	**45.65**	35.47	25.24
	Noise	51.04	43.90	34.35	24.51
	Ours	53.14	44.55	**36.30**	**25.61**
ModeRec	Deterministic	34.13	18.35	11.20	6.75
	Noise	**39.42**	25.56	15.67	11.04
	Ours	36.49	**31.10**	**29.45**	**22.68**

Table 7. Evaluation results of diversity and accuracy metrics.

directly predicts a certain result with the given conditions. We reproduce the *KL divergence, NLL, ModeRec* and *ModePrec* in [36] and use these metrics along with *SR* to evaluate our probabilistic model. The results in Tab. 6 and Tab. 7 suggest our approach has an excellent ability to model the uncertainty in procedure planning and can produce both diverse and reasonable plans(visualizations available in the supplement). Specifically, our approach improves *ModeRec* greatly for longer horizons. There is less uncertainty when $T = 3$, thus the diffusion based models performs worse than the deterministic one.

5. Conclusion

In this paper, we have casted procedure planning in instructional videos as a distribution fitting problem and addressed it with a projected diffusion model. Compared with previous work, our model requires less supervision and can be trained with a simple learning objective. We evaluate our approach on three datasets with different scales and find our model achieves the state-of-the-art performance among multiple planning horizons. Our work demonstrates that modeling action sequence as a whole distribution is an effective solution to procedure planning in instructional videos, even without intermediate supervision.

Acknowledgements. This work is supported by the National Key R&D Program of China (No. 2022ZD0160900), the National Natural Science Foundation of China (No. 62076119, No. 61921006), the Fundamental Research Funds for the Central Universities (No. 020214380091), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *CVPR*, pages 4575–4583. IEEE Computer Society, 2016. 1, 6

[2] Jing Bi, Jiebo Luo, and Chenliang Xu. Procedure planning in instructional videos via contextual modeling and model-based policy learning. In *ICCV*, pages 15591–15600. IEEE, 2021. 1, 2, 3, 4, 6, 7

[3] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *CVPR*, pages 4724–4733. IEEE Computer Society, 2017. 6

[4] Chien-Yi Chang, De-An Huang, Danfei Xu, Ehsan Adeli, Li Fei-Fei, and Juan Carlos Niebles. Procedure planning in instructional videos. In *ECCV (11)*, volume 12356 of *Lecture Notes in Computer Science*, pages 334–350. Springer, 2020. 1, 2, 3, 6, 7

[5] Yudong Chen and Martin J. Wainwright. Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. *CoRR*, abs/1509.03025, 2015. 3

[6] Yingpeng Deng and Lina J. Karam. Universal adversarial attack via enhanced projected gradient descent. In *ICIP*, pages 1241–1245. IEEE, 2020. 3

[7] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, pages 8780–8794, 2021. 4

[8] Nikita Dvornik, Isma Hadji, Konstantinos G. Derpanis, Animesh Garg, and Allan D. Jepson. Drop-dtw: Aligning common signal between sequences while dropping outliers. In *NeurIPS*, pages 13782–13793, 2021. 1

[9] Nikita Dvornik, Isma Hadji, Hai Pham, Dhaivat Bhatt, Brais Martínez, Afsaneh Fazly, and Allan D. Jepson. Flow graph to video grounding for weakly-supervised multi-step localization. In *ECCV (35)*, volume 13695 of *Lecture Notes in Computer Science*, pages 319–335. Springer, 2022. 1

[10] Kiana Ehsani, Hessam Bagherinezhad, Joseph Redmon, Roozbeh Mottaghi, and Ali Farhadi. Who let the dogs out? modeling dog behavior from visual data. In *CVPR*, pages 4051–4060. Computer Vision Foundation / IEEE Computer Society, 2018. 6, 7

[11] Yazan Abu Farha and Juergen Gall. Uncertainty-aware anticipation of activities. In *ICCV Workshops*, pages 1197–1204. IEEE, 2019. 6, 7

[12] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *ICRA*, pages 2786–2793. IEEE, 2017. 1

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 2

[14] Adrian Hauswirth, Saverio Bolognani, Gabriela Hug, and Florian Dörfler. Projected gradient descent on riemannian manifolds with applications to online power system optimization. In *Allerton*, pages 225–232. IEEE, 2016. 3

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016. 6

[16] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin W. Wilson. CNN architectures for large-scale audio classification. In *ICASSP*, pages 131–135. IEEE, 2017. 6

[17] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey A. Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models. *CoRR*, abs/2210.02303, 2022. 2

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 2, 4

[19] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *CoRR*, abs/2207.12598, 2022. 4

[20] Jonathan Ho, Tim Salimans, Alexey A. Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. *CoRR*, abs/2204.03458, 2022. 2

[21] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 9902–9915. PMLR, 2022. 2, 4, 6

[22] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *CoRR*, abs/2107.00630, 2021. 2

[23] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation. *CoRR*, abs/2205.14217, 2022. 2

[24] Chen Liang, Wenguan Wang, Tianfei Zhou, and Yi Yang. Visual abductive reasoning. In *CVPR*, pages 15544–15554. IEEE, 2022. 1, 2

[25] 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 *ICCV*, pages 2630–2640. IEEE, 2019. 6

[26] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 2021. 2, 4

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI (3)*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015. 5, 6

[28] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2256–2265. JMLR.org, 2015. 2

[29] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 4739–4748. PMLR, 2018. 1, 6, 7

[30] 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 Autom. Lett.*, 7(2):4924–4930, 2022. 1, 2

[31] 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 *CVPR*, pages 1207–1216. Computer Vision Foundation / IEEE, 2019. 1, 6

[32] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H. Bermano. Human motion diffusion model. *CoRR*, abs/2209.14916, 2022. 2

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017. 2

[34] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory*, 13(2):260–269, 1967. 2

[35] Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. Diffsound: Discrete diffusion model for text-to-sound generation. *CoRR*, abs/2207.09983, 2022. 2

[36] He Zhao, Isma Hadji, Nikita Dvornik, Konstantinos G. Derpanis, Richard P. Wildes, and Allan D. Jepson. P^3iv: Probabilistic procedure planning from instructional videos with weak supervision. In *CVPR*, pages 2928–2938. IEEE, 2022. 2, 6, 7, 8

[37] Wenliang Zhao, Yongming Rao, Yansong Tang, Jie Zhou, and Jiwen Lu. Videoabc: A real-world video dataset for abductive visual reasoning. *IEEE Trans. Image Process.*, 31:6048–6061, 2022. 1, 2

[38] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David F. Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In *CVPR*, pages 3537–3545. Computer Vision Foundation / IEEE, 2019. 1, 6