ActionDiffusion: An Action-aware Diffusion Model for Procedure Planning in Instructional Videos

Lei Shi[1]   Paul Bürkner[2]   Andreas Bulling[1]
[1]University of Stuttgart, [2]TU Dortmund University
[1]firstname.lastname@vis.uni-stuttgart.de [2]paul.buerkner@gmail.com

Abstract

We present ActionDiffusion – a novel diffusion model for procedure planning in instructional videos that is the first to take temporal inter-dependencies between actions into account. Our approach is in stark contrast to existing methods that fail to exploit the rich information content available in the particular order in which actions are performed. Our method unifies the learning of temporal dependencies between actions and denoising of the action plan in the diffusion process by projecting the action information into the noise space. This is achieved 1) by adding action embeddings in the noise masks in the noise-adding phase and 2) by introducing an attention mechanism in the noise prediction network to learn the correlations between different action steps. We report extensive experiments on three instructional video benchmark datasets (CrossTask, Coin, and NIV) and show that our method outperforms previous state-of-the-art methods on all metrics on CrossTask and NIV and all metrics except accuracy on Coin dataset. We show that by adding action embeddings into the noise mask the diffusion model can better learn action temporal dependencies and increase the performances on procedure planning. Codes are available at https://www.collaborative-ai.org/publications/shi25_wacv/

1. Introduction

To support humans in everyday procedural tasks, such as cooking or cleaning, future autonomous AI agents need to be able to plan actions from visual observations of human actions and their environment – so-called procedure planning [3, 16]. Procedure planning is commonly defined as the task of predicting an action plan, i.e., a sequence of individual actions, from only a start and final observation of the overall procedure. Several previous works have investigated procedure planning from visual observations [8, 9, 27]. But these works have learnt visual representations from artifi-
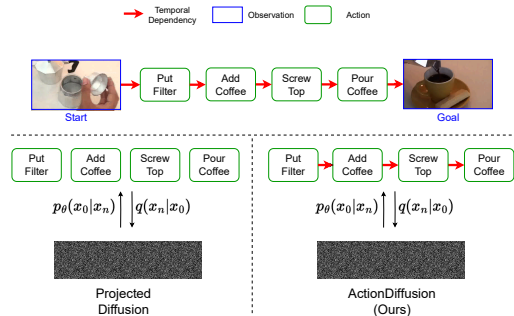

Figure 1. Procedure planning in instructional videos using diffusion models. **Upper section:** Procedure planning task is to generate intermediate actions given the start and goal observation. **Lower left section:** Previous work (Projected Diffusion) [29] does not take the temporal dependencies between actions into account. **Lower right section:** Our method incorporates these dependencies into the diffusion model.

cial and rather simple images, such as a simulated cart pole. Other works have used real-world images to learn action plans [5, 24] but the environment was simplified and constrained by pre-defined object-centred representations [15], for example, coloured cubes as objects on a table. Leveraging more advanced deep learning methods for planning procedures from instructional videos has the potential to address this limitation [2, 3, 34]. Although different approaches have been developed to tackle the procedure planning task, the challenge remains open due to the complex and unstructured video observations.

Diffusion models have achieved outstanding results in many research fields such as image generation [11, 21], text-to-image generation [22, 23, 33], trajectory planning [12, 13], video generation [10, 18], human motion prediction [31], time series imputation and generation [14, 26] and so on. The current state-of-the-art method for procedure planning in videos [29] is also based on a diffusion model. Unlike the diffusion models for images, the input for the diffusion model is a multi-dimensional matrix that consists of the visual observations of the start and goal, the sequence of

actions, and task classes. At inference time, the action plan is taken from the action sequence in the generated full matrix. A key limitation of this work is that the influence of the temporal dependencies of actions, i.e. that actions are more likely to cause particular follow-up actions, has not been considered. Although the input matrix in [29] contains the action labels during the training, the diffusion model still treats the input matrix as a static "image", it does not learn these temporal dependencies.

To overcome this limitation we propose **Action**-aware Noise Mask **Diffusion** (**ActionDiffusion)** – the first method that leverages the temporal dependencies between actions into the diffusion process (see Figure 1). Our method learns the temporal dependencies of actions in the noise space of the diffusion instead of the feature space, thereby unifying the tasks of learning temporal dependencies and generating action plans in the diffusion steps. More specifically, we first propose an action-aware noise mask for the noise-adding stage of the diffusion model. We add the action embeddings in addition to the Gaussian noise in the noise masks so that the model input is transformed into Gaussian noises by iteratively adding the action-aware noise. Second, we introduce an attention mechanism in the denoising neural network (U-Net) to learn the correlations between actions. During the inference, it can then predict action-aware noises to generate action plans. In line with previous work [29, 34], we evaluate the performance of ActionDiffusion through experiments on three popular instructional video benchmarks: CrossTask [35], Coin [25], and NIV [1] with various time horizons. Our results show that Action-Diffusion achieves State-Of-The-Art (SOTA) performances across different metrics for all time horizons on all three datasets. We demonstrate that by adding action embeddings into the noise mask, the diffusion model can effectively learn the temporal dependencies between actions and increase the performance on procedure planning in instructional videos.

The specific contributions of our work are threefold: 1) We propose ActionDiffusion, a novel method incorporating the action temporal dependencies in the diffusion model for procedure planning in instructional videos. We unify the learning of temporal dependencies and action plan generation in the noise space. 2) We add action embeddings into noise masks in the noising-adding stage of diffusion models and use a denoising neural network with self-attention to better learn and predict the action-aware noise to reconstruct the action plan in the denoising phase. 3) We evaluate our methods on CrossTask, Coin, and NIV datasets across various time horizons and achieve SOTA performances in multiple metrics and show the advantage of incorporating action temporal dependencies in the diffusion model, which previous work did not consider.

2. Related Work

2.1. Learning Actions from Videos

There are several lines of research in learning actions from videos. Action recognition [30, 32] is to classify what actions humans are performing in videos. This is a video classification problem. Action anticipation [6, 7] is the task of predicting future actions based on the video given to a model. The task of procedure planning differs from action recognition and action anticipation, it plans the action sequence between the given visual input of the start and goal state. There were works to learn and plan actions from visual input [8, 9, 27]. However, these works learnt visual representations from rather simple simulated images. Other works used real-world images to learn action plans [5, 24], however, the environment was still simple and constrained. Understanding the natural real-life scenario in which humans are present to plan actions is still a challenge and it requires models to have the ability to understand complex visual scenes and human actions.

2.2. Procedure Planning

The task of procedure planning was defined by [3]. The authors used CNNs to extract visual features and modeled the dynamics between observations and actions in the feature space using Multi-Layer Perceptrons (MLPs) and Recurrent Neural Networks (RNNs). In [24], the authors followed the same paradigm of modelling the dynamics between actions and observations but used transformers instead. RL was used in [2] to learn the action policy. All these works used a separate planning algorithm (Beam search or Walk Through) during planning. Later works used generative models to sample action plans during the inference stage. In [34], a generative component together with transformers was trained for sampling action plans during inference. The denoising diffusion probabilistic model was used for procedure planning in [29]. The action plan, task class, o_S and o_g were formed as the input to the diffusion model and the action plan was denoised from random noise.

2.3. Modeling Temporal Dependencies

Previous works used different approaches to model the temporal dependencies between actions in procedure planning. In [3], the authors modeled the temporal dependencies based on Markov Decision Process (MDP), where the action at $t+1$ is based on the current state x_t and the current action a_t. As transformers [28] have shown strong capabilities in sequence modeling, P3IV [34] used the transformer to empirically model the temporal dependencies by putting the start and goal observations as the first and last queries in the transformer and making the intermediate queries (related to action sequences) learnable. PlaTe [24] also used MDP for modeling temporal dependencies, specifically two

transformers, i.e. an action transformer and a state transformer were used. During training, all actions and states were used and during inference all past action-state pairs were used. Although we also use self-attention to model the temporal dependencies, which is a key component in transformers, there are two differences. First, we build the correlation between actions by accumulating action embeddings in noise-adding stage in the diffusion model. Second, we add self-attention in the U-Net to enhance the temporal relations during denoising.

3. Method

3.1. Problem Formulation

We adopt the problem formulation of procedure planning used in previous work [3, 29]: Given the visual observation of the start state o_s and the goal state o_g, the task is to predict the intermediate steps, i.e., the action plan $\pi = a_{1:T}$ for a chosen time horizon T. The action plan π will transform o_s to o_g. More formally, procedure planning can be written as,

$$p(\pi \mid o_s, o_g) = \int p(\pi \mid o_s, o_g, \hat{c}) \, p(\hat{c} \mid o_s, o_g) \, d\hat{c}, \quad (1)$$

where $\hat{c}$ is the predicted task class of the video (e.g., make coffee) and to complete a task, a sequence of actions needs to be performed. The planning is decomposed into two steps [29]: 1) predicting the task class $\hat{c}$ given the start state o_s and the goal state o_g, 2) inferring the action plan π given o_s, o_g, and $\hat{c}$ by sampling from the diffusion model.

3.2. Action-aware Noise Mask Diffusion

Figure 2 provides an overview of our proposed Action-Diffusion method. Taking an instructional video as input, the method first extracts the visual features of the start state o_s, the goal state o_g, and of action embeddings $a_{e_{1:T}}$. It then uses the task class c, one-hot action class $a_{1:T}$, o_s, and o_g to form the input for the diffusion model. Note that during training, we use the ground truth task class c while the predicted task class $\hat{c}$ is used during inference. To obtain $\hat{c}$ we train a separate task classifier that models $p(\hat{c} \mid o_s, o_g)$ in Equation 1. In the noise-adding phase, during training, the noise is added on $a_{1:T}$ in the model input x_0. For each action, we add all previous as well as the current action embedding in addition to the Gaussian noise. In the denoising phase, during inference, we use a U-Net with attention to predict the action-aware noise to denoise the noisy input x_n at step n. The action sequence $\hat{a}_{1:T}$ from the reconstructed input x_0 is the predicted action plan (see below for details).

3.3. Diffusion Model

A diffusion model [11] takes input x_0 and performs two steps on the input: The first one is the noise-adding step,

where Gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is added to x_0 incrementally and eventually x_0 approaches a standard Gaussian distribution in x_N. This noise-adding process $q(x_n \mid x_{n-1})$ for $n = N, \dots, 1$ is described by the following equation,

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

where $\beta_n \in (0, 1)$ is pre-defined (see below). β_n decides how much of the noise is added to x_n. A reparameterization is then applied,

$$x_n = \sqrt{\bar{\alpha}_n} x_0 + \sqrt{1 - \bar{\alpha}_n} \epsilon, \quad (3)$$

where $\bar{\alpha}_n = \prod_{s=1}^{n} (1 - \beta_s)$. A cosine noise scheduling technique [21] is used to determine $\{\beta_s\}_{s=1}^{n}$,

$$\bar{\alpha}_n = \frac{f(n)}{f(0)}, f(n) = \cos(\frac{n/N + \tau}{1 + \tau} \times \frac{\pi}{2})^2, \quad (4)$$

where τ is an offset value to prevent β_n from becoming too small when n is close to 0.

The second step is the denoising process, where the diffusion model samples x_N from Gaussian noise $\mathcal{N}(0, \mathbf{I})$ and denoises x_N to obtain x_0 via the denoising process

$$p_\theta(x_{n-1} \mid x_n) = \mathcal{N}(x_{n-1}; \mu_\theta(x_n, n), \Sigma_\theta(x_n, n)), \quad (5)$$

where $\mu_\theta(x_n, n)$ is parameterised by a neural network $\epsilon_\theta(x_n, n)$, and $\Sigma_\theta(x_n, n)$ is calculated by using $\beta_n \mathbf{I}$.

3.4. Action-aware Noise Mask

We follow [29] to construct the input x_0 to the diffusion model as,

$$x_0 = \begin{bmatrix} c & c & \dots & c & c \\ a_1 & a_2 & \dots & a_{T-1} & a_T \\ o_s & 0 & \dots & 0 & o_g \end{bmatrix}, \quad (6)$$

at the noise-adding step, the noise is only added to the action dimension, i.e. $a_{1:T}$ and a_i is a one-hot vector of a number of A action classes, thus $\mathbf{I} \in \mathbb{R}^{T \times A}$ in Equation (2) and $x_0 \in \mathbb{R}^{T \times (O + A + C)}$, O and C are the dimension of o_s and the dimension of task class c. We design an action-aware noise mask M_a with multiple previous actions accumulated (MultiAdd) as

$$M_a = \begin{bmatrix} 0 & 0 & \dots & 0 \\ g(a_{e_1}) & \sum_{i=1}^{2} g(a_{e_i}) & \dots & \sum_{i=1}^{T} g(a_{e_i}) \\ 0 & 0 & \dots & 0 \end{bmatrix}, \quad (7)$$

where $a_{e_{1:T}}$ are the embeddings of $a_{1:T}$, and $g(\cdot)$ normalises the values of action embeddings to the range of [-1, 1]. M_a has the same dimension as $\mathbf{I}$. Since the denoising step clamps the feature to the range of [-1, 1] too, we normalise the action embeddings for more stable training. In addition to the Gaussian noise applied to x_0, we add the normalised
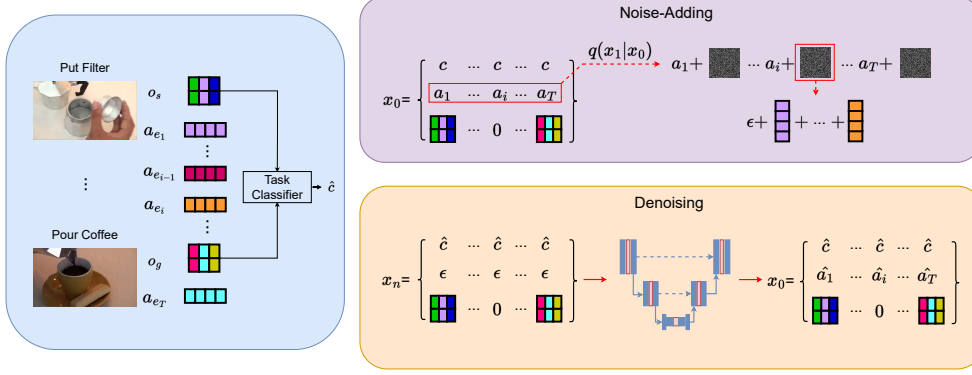
Figure 2. Overview of ActionDiffusion. From an instructional video, we extract the visual feature of the start state o_s and the goal state o_g as well as the features of actions $a_{e_{1:T}}$. We use the task class c, one-hot action class $a_{1:T}$, o_s and o_g as the input of the diffusion model. Note that in the training, we use the ground truth task class c and predicted task class $\hat{c}$ during inference. A separate task classifier is trained to get $\hat{c}$. In the noise-adding phase in training, the noise is added on $a_{1:T}$. For each action, we add all previous action embeddings and the current action embedding in addition to the Gaussian noise. In the denoising phase during inference, we use the U-Net with attention to predict the action-aware noise to denoise x_n. The predicted action plan is the action sequence $\hat{a}_{1:T}$ from the reconstructed input x_0.
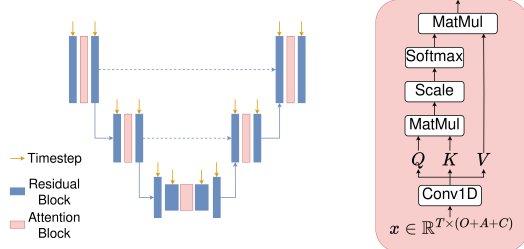

Figure 3. Architecture of the noise prediction neural network ϵ_θ. The network ϵ_θ is based on U-Net and incorporates attention mechanisms.

action embeddings to the noise mask. In the temporal direction, we accumulate all previous action embeddings. Our intuition is that the actions will affect the states and this should be reflected in the noise space. At each $i \in [2 : T]$ the noise mask knows what the previous actions are and μ_θ can learn the temporal dependencies of actions in the denoising stage. With the action-aware noise mask, Equation 2 then becomes,

$$q(x_n \mid x_{n-1}) = \mathcal{N}(x_n; \sqrt{1-\beta_n}x_{n-1}, \beta_n(\mathbf{I}+M_a)), \quad (8)$$

we again apply re-parameterisation and update Equation 3,

$$x_n = \sqrt{\bar{\alpha}_n}x_0 + \sqrt{1-\bar{\alpha}_n}\epsilon_a, \quad (9)$$

where $\epsilon_a \in \mathcal{N}(\mu_a, \sigma_a^2\mathbf{I})$.

3.5. Denoising Neural Network

Following [29], we use a U-Net as the noise prediction neural network ϵ_θ. To better learn the temporal dependencies between actions we further propose to incorporate an

attention mechanism [4, 22]. Figure 3 shows the architecture of the U-Net with attention. As indicated in Equation 5, the U-Net takes x_n and noise schedule time step n as input. n is first passed to a time step block and then is fed to all residual blocks. The attention block takes x as input and extracts Query (Q), Key (K) and Value (V) by a 1D convolutional layer and calculates the standard self-attention.

3.6. Training

In the training phase, the input x_0 to the diffusion model uses ground truth task class c. However, during the inference phase, we need the predicted task class $\hat{c}$ since the procedure planning task has only access to o_s and o_g to infer π. Thus, we train an MLP for predicting $\hat{c}$ using c as supervision. Next, we train the diffusion model using the following squared loss function,

$$\mathcal{L} = \sum_{n=1}^{N}(\mu_\theta(x_n, n) - x_0)^2, \quad (10)$$

where μ_θ is the denoising neural network U-Net, x_n, x_0 and n are the noised input, the input without noise and the timestep for adding noise respectively.

3.7. Inference

During inference, only o_s, o_g and the predicted task class $\hat{c}$ are given and we need to sample the action plan π. We start with constructing x_n. Then, the noise prediction neural network μ_θ predicts the noise iteratively to denoise x_n into x_0. Here, x_n is

$$x_n = \begin{bmatrix} \hat{c} & \hat{c} & \dots & \hat{c} & \hat{c} \\ \epsilon_{a_1} & \epsilon_{a_2} & \dots & \epsilon_{a_{T-1}} & \epsilon_{a_T} \\ o_s & 0 & \dots & 0 & o_g \end{bmatrix}, \quad (11)$$

only the action plan π is initialised with noise ϵ_{a_i} and $\epsilon_{a_i} \in \mathcal{N}(0, \sigma_{a_i}^2 \mathbf{I})$ since the noise is only added to the actions during the noise-adding stage. After N steps of denoising, we take the action sequence $\hat{a}_{1:T}$ as the action plan π. In other words, all actions in π are generated at once. It is worth noting that we use action-aware noise masks in the noise-adding stage, the action embeddings are added to Gaussian noise (Equation 8). Since the action embeddings are approximately normally distributed (see Figure 4), the noised input after N time steps is also approximately normally distributed with a different standard deviation compared to Gaussian noise (see Figure 5). Hence, instead of sampling from $\epsilon \in \mathcal{N}(0, \mathbf{I})$, we sample from $\epsilon_{a_i} \in \mathcal{N}(0, \sigma_{a_i}^2 \mathbf{I})$. The comparison of using ϵ and ϵ_a for inference is shown in supplementary material.
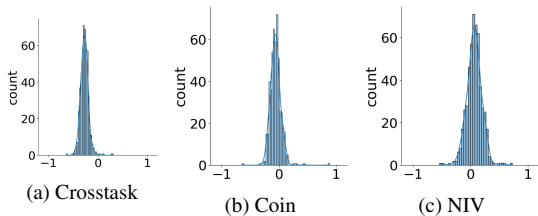

Figure 4. Examples of action embedding distributions from the CrossTask (a), Coin (b), and NIV datasets (c).

4. Experiments

4.1. Datasets

We evaluate ActionDiffusion on three instructional video benchmark datasets: CrossTask [35], Coin [25], and NIV [1]. The CrossTask dataset contains 2,750 videos, 18 tasks, and 105 action classes. The average number of actions per video is 7.6. The Coin dataset has a much larger number of videos, tasks, and action classes: In total, it contains 11,827 videos, 180 tasks, and 778 action classes. The average number of actions per video is 3.6. Finally, the NIV dataset is the smallest among the three datasets. It contains 150 videos, five tasks, and 18 action classes as well as 9.5 actions per video on average. We follow the data curation process used in previous work [3, 29]: For a video containing m number of actions, we extract action sequences with the time horizon T using a sliding window. For an extracted action sequence $[a_i, ..., a_{i+T-1}]$, each action has a corresponding video clip. We extract the video clip feature at the beginning of a_i as the visual observation of the start state o_s and the video clip feature at the end of a_{i+T-1} as the goal state o_g. To facilitate comparability, we use the pre-extracted features from previous work [29, 34]. The features were extracted by using a model [19] which was pre-trained on the HowTo100M dataset [20]. In addition to the video clip features, the model also generated embeddings for actions

corresponding to video clips. We use the action embeddings for our action-aware diffusion. We randomly use 70% of the data for training and 30% for testing.

4.2. Implementation Details

We use AdamW [17] as the optimizer for all datasets. For the training on the Crosstask dataset, we train with batch size 256 and 120 epochs. In each epoch, the train step is 200. The first 20 epochs are the warm-up stage, and the learning rate increases to $5e^{-4}$ linearly. In the last 30 epochs, the learning rate decays by 0.5 for every 5 epochs. For the Coin dataset, we train 800 epochs with batch size 256 with 200 train steps in each epoch. The warm-up state is 20 epochs where the learning rate increases to $1e^{-4}$ linearly. The learning rate decreases in the last 50 epochs with a decay rate 0.5. For the NIV dataset, we train 130 epochs with batch size 256. The train step in each epoch is 50. The learning rate increases to $1e^{-5}$ for $T = 3$ and $3e^{-6}$ for $T = 4$ in the first 90 epochs.

We use the same task predictor and adapt the same training strategy from [29] for predicting the task classes. The task predictor is a 4-layer MLP and it takes the image features of o_s and o_g as input and outputs the predicted task class $\hat{c}$. The accuracy is over 92% on Crosstask, over 78% on Coin and 100% on NIV.

During inference, we sample x_n using ϵ_a instead of ϵ. The noised input approximately follows a normal distribution with a different mean and standard deviation other than the Gaussian noise. The mean shifts and the standard deviation are smaller than one (see Figure 5). We calculate the mean and standard deviations of the noised inputs on the training sets of Crosstask, Coin and NIV and use them for sampling in the inference on the test sets. The standard deviation of each action within the time horizon T for all datasets is shown in Table 1.

The proposed action-aware noise mask M_a will not cost additional computational burden for training since the action embeddings are extracted by using [19]. During training, the action embedding is only added to the Gaussian noise. Adding the attention mechanism will increase the number of parameters of U-Net, the comparison of the numbers of parameters is in the supplementary material.

Table 1. Mean and standard deviations of $\epsilon_{a[1:T]}$ on the training sets of CrossTask, Coin and NIV.

	T_1		T_2		T_3		T_4		T_5		T_6	
	μ	σ_a^2	μ	σ_a^2	μ	σ_a^2	μ	σ_a^2	μ	σ_a^2	μ	σ_a^2
CrossTask	-0.27	0.09	-0.54	0.13	-0.81	0.16	-1.09	0.18	-1.35	0.21	-1.62	0.22
Coin	-0.04	0.59	-0.08	0.68	-0.11	0.72	-0.14	0.72	-	-	-	-
NIV	0.06	0.11	0.12	0.17	0.19	0.20	0.26	0.23	-	-	-	-

4.3. Metrics

We use three metrics for evaluation: The first metric is the **Success Rate (SR)**. An action plan is considered cor-

rect, and thus the procedure planning is a success, only if all actions in the plan are correct and the order of the actions is correct. This is the most strict metric. The second metric is the **mean Accuracy (mAcc)**. The accuracy is calculated based on the individual actions in the action plan. The order of the actions is not considered. The last metric is **mean Single Intersection over Union (mSIoU)**. The calculation of IoU treats the predicted action plans and the ground truth action plans as sets and also does not consider the order of actions. The works in [2,3,34] calculated **mIoU** with all action plans in a mini-batch. However, as pointed out in [29], this calculation is dependent on the batch size. To ensure a fairer comparison, they proposed the **mSIoU** metric instead, which treats each single action plan as a set and is thus agnostic to the batch size. We also opted for the **mSIoU** metric.

4.4. Baselines

We compare our method with several state-of-the-art baseline methods for procedure planning: **DDN** [3], **Ext-GAIL** [2], **P³IV** [34] and **PDPP** [29]. More details about the baseline models are in the supplementary material.

5. Results

5.1. Action Embedding in Noise-Adding

To verify if the noised input follows the normal distribution as mentioned in Section 3.7, we plot the noised actions $\epsilon_{a[1:T]}$ in x_n, i.e. the second row in Equation 11, for time horizon $T = 3$ on all datasets. Additionally, we plot the noised actions $\epsilon_{[1:T]}$ which use $\mathcal{N}(0, \mathbf{I})$ as the noise mask. The time step N is 200, 200 and 50 for CrossTask, Coin and NIV respectively. The results are shown in Figure 5. When using ϵ as the noise mask for the noise-adding, all noised actions approximately follow the same normal distribution for all three datasets. When using ϵ_a as the noise mask, we can observe from the figure that all action distributions in all datasets also follow normal distributions approximately. The mean of the distribution shifts towards the negative direction for CrossTask and Coin, and shifts toward the positive direction for NIV. For actions a_2 and a_3, the means shift further away from zero. The reason is that the mean of action embedding distribution is not zero (Figure 4), a_3 accumulates the action embeddings of a_1 and a_2. Additionally, the standard deviations of ϵ_a are increasing, i.e. $\sigma_{a3}^2 > \sigma_{a2}^2, \sigma_{a2}^2 > \sigma_{a1}^2$. The reason is that at a_3 all previous action embeddings are added to the noise mask.

5.2. Comparison to SOTA Methods

5.2.1 CrossTask

Table 2 shows the results on the CrossTask dataset with time horizon $T = 3$ and $T = 4$. ActionDiffusion achieves SOTA
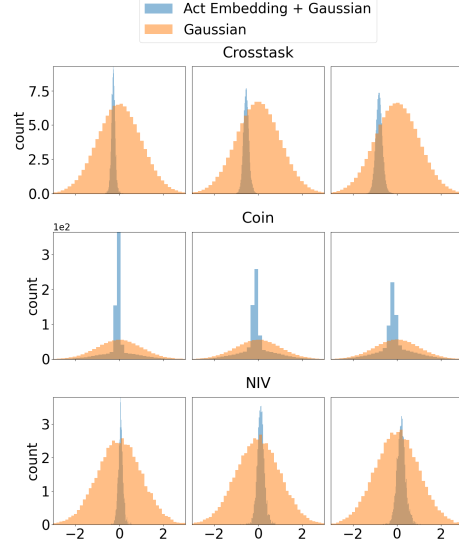

Figure 5. The distributions of diffusion model input after N steps of noise-adding for time horizon $T = 3$. Each column shows the distributions at $a_i, i \in T$. The distribution in blue uses action embedding with Gaussian noise for noise-adding stage. The distribution in orange uses Gaussian noise only. The first row shows the distributions from CrossTask dataset. The second row shows the distributions from Coin dataset. The third row shows the distributions from NIV dataset.

performances for all metrics in both time horizons. Note that DNN, Ext-GAIL, and P³IV only reported mIoU, and only PDPP reported mSIoU. We also evaluate our method with the longer time horizon $T \in \{3, 4, 5, 6\}$. Table 3 shows the SR for all the time horizons. We again have SOTA performances in all time horizons.

5.2.2 Coin

Table 4 shows the results on Coin dataset with time horizon $T = 3$ and $T = 4$. For $T = 3$, we achieved SOTA performance on SR and mSIOU. Although the mAcc is slightly lower than PDPP, our SR is still 2.67% higher than PDPP. This means that ActionDiffusion can learn the temporal dependencies better than PDPP even though the percentage of correctly predicted actions (ignoring the order) is slightly lower. When $T = 4$, we have SOTA results on all metrics. Our mAcc is slightly better than PDPP, SR and mSIoU are 4.63% and 4.84% higher. This also shows our method is better at capturing the temporal dependencies.

5.2.3 NIV

The results on the NIV dataset are shown in Table 4. We obtain SOTA performance on all metrics in both time hori-

Table 2. Results on CrossTask dataset with time horizon $T = 3$ and $T = 4$. Numbers in **Bold** indicate the best results. The arrow ↑ means higher numbers are better.

	T=3			T=4		
	SR(%)↑	mAcc(%)↑	mSIoU(%)↑	SR(%)↑	mAcc(%)↑	mSIoU(%)↑
DDN [3]	12.18	31.29	-	5.97	27.10	-
Ext-GAIL [2]	21.27	49.46	-	16.41	43.05	-
P³IV [34]	23.34	49.96	-	13.40	44.16	-
PDPP [29]	37.20	64.67	66.57	21.48	57.82	65.13
ActionDiffusion-MultiAdd (Ours)	**37.79**	**65.38**	**67.45**	**22.43**	**59.42**	**66.04**

Table 3. Results on CrossTask dataset with time horizon $T \in \{3, 4, 5, 6\}$. Numbers in **Bold** indicate the best results. The arrow ↑ means higher numbers are better.

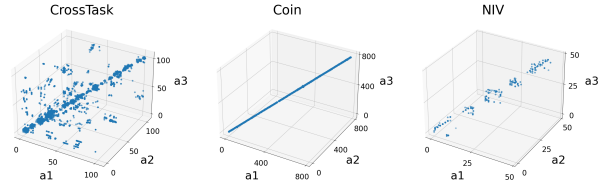|	T=3	T=4	T=5	T=6
	SR(%)↑	SR(%)↑	SR(%)↑	SR(%)↑
DDN [3]	12.18	5,97	3.10	1.20
Ext-GAIL [2]	21.27	16.41	-	-
P³IV [34]	23.34	13.40	7.21	4.40
PDPP [29]	37.20	21.48	13.45	8.41
ActionDiffusion-MultiAdd (Ours)	**37.79**	**22.43**	**13.89**	**9.66**

Figure 6. Distributions of action labels for $T = 3$. Each point in the 3D coordinate represents one action sequence. The coordinates in x, y and z directions is the action labels for a_1, a_2 and a_3 respectively.

zons. The SR is 2.76% higher than PDPP when $T = 3$ and 2.59% higher when $T = 4$. The mACC is 1.18% and 1.39% higher when $T = 3$ and $T = 4$. The mSIoU is also slightly higher when $T = 3$ and $T = 4$.

5.3. Ablation Study

5.3.1 Noise Mask

In Equation 7, we describe the **MultiAdd** action-aware noise mask. At each time step within the time horizon T, we accumulate the embeddings of all previous actions. We want to compare it with **SingleAdd** noise mask described as follows,

$$\begin{bmatrix} 0 & 0 & ... & 0 \\ g(a_{e_1}) & g(a_{e_2}) & ... & g(a_{e_T}) \\ 0 & 0 & ... & 0 \end{bmatrix}, \quad (12)$$

where at each time step only one action embedding is added to the noise mask.

Table 5 shows the results of the MultiAdd mask, SingleAdd mask and without mask (NoMask) on all three datasets with time horizon $T = 3$ and $T = 4$. MultiAdd outperforms SingleAdd and NoMask on all metrics on the Coin dataset and NIV dataset for both time horizons. On the CrossTask dataset, SingleAdd performs the best on SR when $T = 3$. NoMask performs the best on mAcc and MSIoU for $T = 3$ and all metrics for $T = 4$. Although NoMask has the best overall performance on CrossTask, the results of MultiAdd, SingleAdd and NoMask are comparable. Additionally, the performances of NoMask are worse on Coin and NIV, especially on Coin. We interpret

the reason for the performance differences is that the differences in action label compositions in action sequences from datasets. Figure 6 shows the distributions of ground truth action labels in action sequences. The action labels in action sequences in CrossTask are more scattered than in Coin and NIV. For instance, lots of action sequences look like $a_{1:T} = [56, 0, 57]$, where 56, 0 and 57 are the action labels. And most of the action sequences in Coin and NIV look like $a_{1:T} = [48, 49, 50]$. Overall the distribution of action sequences is more linear in Coin and NIV. We think this is the reason that the action-aware mask works better since it is easier for the action-aware mask to build temporal dependencies. Overall, MultiAdd can perform better when the distribution of action labels is more linear (Coin and NIV). For the more scattered distributions, MultiAdd and SingleAdd are not as effective. Nevertheless, MultiAdd still achieves the SOTA performances.

5.3.2 Self-Attention in U-Net

We study the effect of the self-attention mechanism in the U-Net in this section. We use the Multi-Add noise mask and test the U-Net with and without self-attention on all three datasets. The results are shown in Table 6. ActionDiffusion with and without self-attention achieve comparable results on the CrossTask dataset. On the Coin dataset, ActionDiffusion with self-attention performs better on all metrics when $T = 3$. ActionDiffusion without self-attention performs better on all metrics when $T = 4$, although the results of ActionDiffusion with and without self-attention

Table 4. Results on Coin and NIV datasets with time horizon $T = 3$ and $T = 4$. Numbers in **Bold** indicate the best results. The arrow $\uparrow$ means higher numbers are better. ActionDiffusion means our method using MultiAdd mask.

Horizon	Models	COIN SR(%)↑	mAcc(%)↑	mSIoU(%)↑	NIV SR(%)↑	mAcc(%)↑	mSIoU(%)↑
T=3	DDN [3]	13.90	20.19	-	18.41	32.54	-
	Ext-GAIL [2]	-	-	-	22.11	42.20	-
	P^{3}IV [34]	15.40	21.67	-	24.68	49.01	-
	PDPP [29]	21.33	**45.62**	51.82	30.20	48.45	57.28
	ActionDiffusion-MultiAdd (Ours)	**24.00**	45.42	**54.29**	**32.96**	**49.26**	**57.84**
T=4	DDN [3]	11.13	17.71	-	15.97	2.73	-
	Ext-GAIL [2]	-	-	-	19.91	36.31	-
	P^{3}IV [34]	11.32	18.85	-	20.14	28.36	-
	PDPP [29]	14.41	44.10	51.39	26.67	46.89	59.45
	ActionDiffusion-MultiAdd (Ours)	**18.04**	**44.54**	**56.23**	**29.26**	**48.14**	**60.71**

Table 5. Comparison between MultiAdd noise mask, SingleAdd noise mask and without mask on Crosstask, Coin and NIV datasets. Numbers in **Bold** indicate the best results. The arrow $\uparrow$ means higher numbers are better.

Dataset	Models	T=3 SR↑	mAcc↑	mSIoU↑	T=4 SR↑	mAcc↑	mSIoU↑
CrossTask	ActionDiffusion-Multi	37.79	65.38	67.45	22.43	59.42	66.04
	ActionDiffusion-Single	**38.21**	65.34	67.25	22.32	58.92	65.26
	ActionDiffusion-NoMask	37.92	**65.53**	**67.65**	**22.99**	**59.48**	**66.23**
Coin	ActionDiffusion-Multi	**24.00**	**45.42**	**54.29**	**18.04**	**44.54**	**56.23**
	ActionDiffusion-Single	21.52	43.13	52.98	14.97	42.56	55.04
	ActionDiffusion-NoMask	12.46	36.08	43.44	2.47	26.96	34.70
NIV	ActionDiffusion-Multi	**32.96**	**49.26**	**57.84**	**29.26**	**48.14**	**60.71**
	ActionDiffusion-Single	30.74	47.03	56.00	25.76	44.98	57.92
	ActionDiffusion-NoMask	29.26	44.81	54.49	21.40	35.59	51.38

Table 6. Comparison between the U-Net with self-attention (w attention) and the U-Net without self-attention (w/o attention). Numbers in **Bold** indicate the best results. The arrow $\uparrow$ means higher numbers are better.

Dataset	Models	T=3 SR↑	mAcc↑	mSIoU↑	T=4 SR↑	mAcc↑	mSIoU↑
CrossTask	w attention	**37.79**	65.38	**67.45**	22.43	**59.42**	66.04
	w/o attention	37.75	**65.47**	**67.45**	**22.56**	59.17	**66.15**
Coin	w attention	**24.00**	**45.42**	**54.29**	18.04	44.54	56.23
	w/o attention	22.88	44.52	53.56	**18.36**	**44.88**	**56.49**
NIV	w attention	**32.96**	**49.26**	**57.84**	**29.26**	**48.14**	**60.71**
	w/o attention	32.22	48.52	57.58	28.82	46.07	59.12

are comparable. On the NIV dataset, ActionDiffusion with self-attention outperforms the one without self-attention on all metrics with both time horizons. Overall, ActionDiffusion with self-attention performs better than without self-attention on the NIV dataset, while they have comparable performance on the other two datasets. We interpret the reason for this as the size of NIV is much smaller than the other two. ActionDiffusion with attention can better learn the temporal dependencies integrated into the noise mask when the data is limited.

6. Conclusion

In this work, we propose ActionDiffusion, an action-aware diffusion model, to tackle the challenge of procedure planning in instructional videos. We integrate temporal dependencies between actions by adding action embeddings to the noise mask during the diffusion process. We achieve SOTA performances on three procedure planning datasets across multiple metrics, showing the novelty of adding action embedding in the noise mask as the modelling of temporal dependencies.

7. Acknowledgement

References

[1] 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. 2, 5

[2] 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. 1, 2, 6, 7, 8

[3] 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. 1, 2, 3, 5, 6, 7, 8

[4] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 4

[5] Kuan Fang, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. Dynamics learning with cascaded variational inference for multi-step manipulation. *arXiv preprint arXiv:1910.13395*, 2019. 1, 2

[6] Antonino Furnari and Giovanni Maria Farinella. Rolling-unrolling lstms for action anticipation from first-person video. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4021–4036, 2020. 2

[7] Dayoung Gong, Joonseok Lee, Manjin Kim, Seong Jong Ha, and Minsu Cho. Future transformer for long-term action anticipation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3052–3061, 2022. 2

[8] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019. 1, 2

[9] Yiqiang Han, Wenjian Hao, and Umesh Vaidya. Deep learning of koopman representation for control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1890–1895. IEEE, 2020. 1, 2

[10] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 1

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1, 3

[12] Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16750–16761, 2023. 1

[13] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022. 1

[14] Chuhan Jiao, Yao Wang, Guanhua Zhang, Mihai Bâce, Zhiming Hu, and Andreas Bulling. Diffgaze: A diffusion model for continuous gaze sequence generation on 360 {\deg} images. *arXiv preprint arXiv:2403.17477*, 2024. 1

[15] Nishanth Kumar, Willie McClinton, Rohan Chitnis, Tom Silver, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning efficient abstract planning models that choose what to predict. In *Conference on Robot Learning*, pages 2070–2095. PMLR, 2023. 1

[16] Martina Lippi, Petra Poklukar, Michael C Welle, Anastasiia Varava, Hang Yin, Alessandro Marino, and Danica Kragic. Latent space roadmap for visual action planning of deformable and rigid object manipulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5619–5626. IEEE, 2020. 1

[17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[18] Zhengxiong Luo, Dayou Chen, Yingya Zhang, Yan Huang, Liang Wang, Yujun Shen, Deli Zhao, Jingren Zhou, and Tieniu Tan. Videofusion: Decomposed diffusion models for high-quality video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10209–10218, 2023. 1

[19] 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. 5

[20] 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. 5

[21] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 1, 3

[22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1, 4

[23] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023. 1

[24] Jiankai Sun, De-An Huang, Bo Lu, Yun-Hui Liu, Bolei Zhou, and Animesh Garg. Plate: Visually-grounded planning with transformers in procedural tasks. *IEEE Robotics and Automation Letters*, 7(2):4924–4930, 2022. 1, 2

[25] 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. 2, 5

[26] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816, 2021. 1

[27] Bas van der Heijden, Laura Ferranti, Jens Kober, and Robert Babuška. Deepkoco: Efficient latent planning with a task-relevant koopman representation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 183–189. IEEE, 2021. 1, 2

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

[29] 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, 5, 6, 7, 8

[30] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks for action recognition in videos. *IEEE transactions on pattern analysis and machine intelligence*, 41(11):2740–2755, 2018. 2

[31] Haodong Yan, Zhiming Hu, Syn Schmitt, and Andreas Bulling. Gazemodiff: Gaze-guided diffusion model for stochastic human motion prediction. *arXiv preprint arXiv:2312.12090*, 2023. 1

[32] Ceyuan Yang, Yinghao Xu, Jianping Shi, Bo Dai, and Bolei Zhou. Temporal pyramid network for action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 591–600, 2020. 2

[33] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 1

[34] 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. 1, 2, 5, 6, 7, 8

[35] 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. 2, 5