# MetaDiffuser: Diffusion Model as Conditional Planner for Offline Meta-RL

**Anonymous Authors**[1]

## Abstract

Recently, diffusion model shines as a promising backbone for the sequence modeling paradigm in offline reinforcement learning (RL). However, these works mostly lack the generalization ability across tasks with reward or dynamics change. To tackle this challenge, in this paper we propose a task-oriented conditioned diffusion planner for offline meta-RL (MetaDiffuser), which considers the generalization problem as conditional trajectory generation task with contextual representation. The key is to learn a context conditioned diffusion model which can generate task-oriented trajectories for planning across diverse tasks. To enhance the dynamics consistency of the generated trajectories while encouraging trajectories to achieve high returns, we further design a dual-guided module in the sampling process of the diffusion model. The proposed framework enjoys the robustness to the quality of collected warm-start data from the testing task and the flexibility to incorporate with different task representation method. The experiment results on MuJoCo benchmarks show that MetaDiffuser outperforms other strong offline meta-RL baselines, demonstrating the outstanding conditional generation ability of diffusion architecture. More visualization results are released on project page.

## 1. Introduction

Offline Reinforcement Learning (Offline RL) (Levine et al., 2020) aims to learn policies from pre-collected data without interacting with the environment and has made many success in the fields of games (Chen et al., 2021), robotic manipulation (Ebert et al., 2018). However, one of the inherent difficulties of offline RL is the challenges to generalize to unseen tasks. Recent work in offline meta-RL (Mitchell et al., 2021b; Li et al., 2020; 2021a;b) aims to solve this problem by training a meta-policy from multi-task offline datasets that can efficiently adapt to unseen tasks with small amounts of warm-start data.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

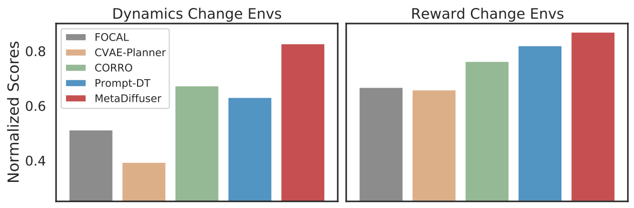


*Figure 1.* Overall few-shot generalization performance comparisons on various environments including 2 domains with dynamics change and 4 domains with reward change. The expert performance in each environment is chosen as normalized baseline.

Conventional offline meta-RL methods (Li et al., 2021a; Yuan & Lu, 2022) learn a context encoder to infer task representation and a meta policy conditioned on the learned context for generalization across tasks. These works extended from the online meta-RL setting, still rely on context-conditioned policy trained by temporal difference (TD) learning, which may potentially cause instability in policy optimization and limited performance (Levine et al., 2020; Ajay et al., 2022). A more recent work Prompt-DT (Xu et al., 2022) turn to tackle the generalization problem from the sequence modeling perspective, which joint models state-action trajectories to avoid TD-learning. This approach uses prompting method to generalize across unseen tasks without the need for explicit extraction of task representation through pre-trained context encoder. However, the key limitation is that the quality of the pre-collected warm-start data must be high enough, which is challenging to collect in unseen tasks, to act as an expert prompt for guiding sequence generation, otherwise performance may suffer with random or medium data. The aforementioned limitations raise a key question: Can we design a offline meta-RL framework to achieve the generalization ability across multiple tasks with robustness for the quality of warm-start data while utilize the promising ability of sequence-modeling paradigm?

Planning with diffusion model (Janner et al., 2022b) provides a promising paradigm for offline RL, which utilizes diffusion model as a trajectory generator by joint diffusing the states and actions from the noise to formulate the sequence decision-making problem as standard generative modeling. The concurrent works (Ajay et al., 2022; Wang et al., 2022b) also showcase the potential of the diffusion model as a highly promising generative model, highlighting its ability to serve as a key backbone for addressing sequence modeling problems in RL, while avoiding the limitations of TD-learning. But these works focus on a single task and lack research on generalization ability across tasks, which leaves the conditioned diffusion unexplored for offline meta-RL. However, conditioned diffusion model has made significant progress in vision and language tasks (Ho & Salimans, 2022), such as DALL-E (Ramesh et al., 2022) and ImageGen (Saharia et al., 2022) for text-to-image generation tasks. These works demonstrate the powerful conditional generation capabilities of conditioned diffusion models with the textual label without the need for expert images as prompts.

Inspired by this, we propose a novel framework for offline meta-RL, named MetaDiffuser that leverages diffusion model to conduct desired trajectories generation for generalization across unseen tasks. During meta-training, to provide accurate conditional labels for subsequent trajectory generation, we first pre-train an accurate context encoder that can capture task-relevant information from offline trajectories mixed with different tasks. Then the compact task representation is injected as a contextual label to the conditional diffusion model to manipulate the task-oriented trajectories generation. In this way, the diffusion model learns to estimate the conditional distribution of multi-task distributions based on the task-oriented context. During meta-testing, with the predicted context from provided warm-start data in the testing task, the conditional diffusion model can denoise out desired trajectories for the testing task. The generated trajectories can guide the subsequent action to step into the next state, similar to the planning in RL. Moreover, to decrease the discrepancy between generated trajectories and real-rollout trajectories, we design an effective dual-guide to enhance the dynamics consistency of generated trajectories while encouraging the high return simultaneously. The contributions of this work are as follows:

- **Generalization Ability**: We propose MetaDiffuser to leverage the diffusion model to conduct conditional trajectory generation to achieve the generalization ability across unseen tasks.
- **Robustness and Flexibility**: MetaDiffuser enjoys the flexibility to incorporate with different task representation method and the robustness to the quality of collected warm-start data at the testing task.
- **Dual-guide Enhanced Planner**: We design the dual-guide of both dynamics and rewards to ensure the feasibility of guided trajectories while encouraging the generated trajectories to achieve high returns.
- **Superior Performance**: The experiments on various benchmarks empirically show that MetaDiffuser much better generalizes to unseen tasks than prior methods.

## 2. Related Work

### 2.1. Offline Meta-RL

Offline meta-RL investigates learning to learn from offline data, with the aim to quickly adapt to unseen tasks. Recent works (Mitchell et al., 2021b; Li et al., 2020; 2021a), including FOCAL (Li et al., 2021b) and CORRO (Yuan & Lu, 2022), trains a context encoder for compact task representation for the conditioned policy to generalize. These methods extended from the traditional online meta-RL setting, still rely on context-conditioned policy trained by TD-learning, which may potentially cause instability in policy optimization and limited performance. Prompt-DT (Xu et al., 2022) turns to solve the generalization problem from the sequence modeling perspective, which joint models state-action trajectories to avoid TD-learning. This approach can utilize the collected prompt as a prefix to generalize across tasks without the need for explicit context encoder. However, the key limitation is the high requirement for the quality of warm-start data as prompt, which is challenging to pre-collect in unseen task. See more discussion in Appendix C. To combine the best of both context-based manner and sequence-modeling fashion, we propose MetaDiffuser, which not only avoiding TD-loss, but also enjoy the robustness to the quality of warm-start data.

### 2.2. Diffusion Model for Sequence Decision Making

Recently, many works have emerged to utilize diffusion models to solve sequence decision-making tasks, showing the great potential of diffusion model as a promising backbone of sequence modeling. Diffuser (Janner et al., 2022b) applies a diffusion model as a trajectory generator, which is trained by diffusing over the full trajectory of state-action pairs from the noises. A separate reward model is trained to predict the cumulative rewards of each trajectory sample, then the gradient guidance from reward model is injected into the reverse sampling stage. Then the first action in the generated trajectories will be applied to execute in the environment to step into the next state, which repeats in a loop until the terminal. The consequent work Decision Diffuser (Ajay et al., 2022) frames offline sequential decision-making as conditional generative modeling based on returns, constraints and skills to eliminate the complexities in traditional offline RL. The concurrent work Diffusion-QL (Wang et al., 2022b), build policy with the reverse chain of a conditional diffusion model, which allows for a highly expressive policy class, as a strong policy-regularization method. However, these works mostly focus on a single task and lack the generalization ability to unseen tasks in the setting of offline meta-RL. Our approach MetaDiffuser leverages the conditioned diffusion model to conduct conditional trajectory generation to achieve the generalization across unseen tasks with different reward functions or dynamics.

### 2.3. Conditional Diffusion Model

Recently, there have been incredible advances in the field of conditional content generation with the strong generation capabilities of conditioned diffusion models. Conditional diffusion model pushes the state-of-the-art on text-to-image generation tasks such as DALL-E (Ramesh et al., 2022) and ImageGen (Saharia et al., 2022). The technique of conditioning can divide into two fashions: classifier-guided (Nichol & Dhariwal, 2021) and classifier-free (Ho & Salimans, 2022). The former improves sample quality while reducing diversity in conditional diffusion models using gradients from a pre-trained classifier $p_\phi(\boldsymbol{y}|\boldsymbol{x}_k)$ during sampling. The latter is an alternate technique that avoids this pre-trained classifier by instead jointly training a single diffusion model on conditional $\epsilon_\theta(\boldsymbol{x}_k, \boldsymbol{y}, k)$ and unconditional $\epsilon_\theta(\boldsymbol{x}_k, k)$ noise model via randomly dropping conditional label $\boldsymbol{y}$.

In fact, the aforementioned Diffuser (Janner et al., 2022b) can also be considered as a classifier-guided conditional diffusion model, where the pre-trained reward model is another form of classifier for evaluating the sample quality. Our designed MetaDiffuser builds upon the Diffuser and additionally incorporates classifier-free manner, by injecting the context as label $\boldsymbol{y}$ into the conditional noise model $\epsilon_\theta(\boldsymbol{x}_k, \boldsymbol{y}, k)$, achieving more precise conditional generation. The details about the relationship between two different conditional fashions can be found in Appendix A.

## 3. Preliminaries

### 3.1. Problem Formulation

The reinforcement learning problem can be generally modeled as a Markov Decision Process (MDP), represented as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \rho, R)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T(s'|s,a)$ is the transition dynamics of the environment, $\rho(s)$ is the initial state distribution, $R(s,a)$ is the reward function. The objective is to find a policy $\pi(a|s)$ that optimizes the expected cumulative rewards, $\mathbb{E}_{s_0\sim\rho,\pi}\sum_t \gamma^t R(s_t)$. In the offline meta-RL setting, aiming to adapt to new tasks via pre-collected data quickly, an agent is given a set of tasks $\mathcal{T}$, where a task $\mathcal{T}_i \in \mathcal{T}$ is defined as $(\mathcal{M}_i, \pi_i)$, containing an MDP $\mathcal{M}_i$ and a behavior policy $\pi_i$. For each task $\mathcal{T}_i$, the agent is provided with a dataset $\mathcal{D}_i$, which contains trajectories sampled using $\pi_i$. The agent is trained with a subset of training tasks denoted as $\mathcal{T}^{train}$ and is expected to find the optimal policies in a set of test tasks $\mathcal{T}^{test}$, which is disjoint with $\mathcal{T}^{train}$.

### 3.2. Diffusion Model

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) are a type of generative model that consists a forward diffusion process and a reverse denoising process to learn the data distribution $q(\boldsymbol{x})$. Here, the data-generating procedure is modelled with a predefined forward noising process $q(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k) := \mathcal{N}(\boldsymbol{x}_{k+1}; \sqrt{\alpha_k}\boldsymbol{x}_k, (1-\alpha_k)\boldsymbol{I})$ and a trainable reverse process $p_\theta(\boldsymbol{x}_{k-1}|\boldsymbol{x}_k) := \mathcal{N}(\boldsymbol{x}_{k-1}|\mu_\theta(\boldsymbol{x}_k, k), \Sigma_k)$, where $\mathcal{N}(\mu, \Sigma)$ denotes a Gaussian distribution with mean $\mu$ and variance $\Sigma$, $\alpha_k \in \mathbb{R}$ determines the variance schedule, $\boldsymbol{x}_0 := \boldsymbol{x}$ is a sample, $\boldsymbol{x}_k$ are the sequentially sampled latent variables for $k = 1, \ldots, K$, and $\boldsymbol{x}_K \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ for carefully chosen $\alpha_k$ and long enough $K$. Starting with Gaussian noise, samples are then iteratively generated through a series of reverse denoising steps by the predicted noise. The predicted noise $\epsilon_\theta(\boldsymbol{x}_k, k)$, parameterized with a deep neural network, estimates the noise $\epsilon \sim \mathcal{N}(0, I)$ added to the dataset sample $\boldsymbol{x}_0$ to produce noisy $\boldsymbol{x}_k$, which can be trained by a simplified surrogate loss (Ho et al., 2020): $\mathcal{L}_{\text{denoise}}(\theta) := \mathbb{E}_{k\sim[1,K], \boldsymbol{x}_0\sim q, \epsilon\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})}[||\epsilon - \epsilon_\theta(\boldsymbol{x}_k, k)||^2]$.

## 4. Methodology

To tackle the generalization challenge from the sequence modeling perspective, we propose MetaDiffuser, a novel offline meta-RL framework, which leverages the conditioned diffusion model to conduct task-oriented trajectories generation for generalization across unseen tasks. As shown in Figure 2, the overall generalization process can explicitly be divided into meta-training and meta-testing. During **meta-training**, in order to provide accurate conditional labels for subsequent trajectory generation, we first need to pre-train an accurate context encoder that can capture both reward changes and dynamics changes from trajectories. Then the compact task representation inferred by the context encoder is injected as a contextual label into the step-wise denoising process from the Gaussian noise for estimating the conditional distribution of multi-task trajectories. During **meta-testing**, with predicted context from provided warm-start data, the conditional diffusion model can denoise out desired trajectories for the testing task. Moreover, to alleviate the discrepancy between generated trajectories and real-rollout trajectories, the previously trained reward model and dynamics model are utilized as a trajectory evaluator to enhance the dynamics consistency and high returns of trajectories.

### 4.1. Task-oriented Context Encoder

To manipulate the conditional trajectory generation with a high correlation with the desired specific task, it is necessary to establish an accurate mapping from trajectories to the contextual label it belongs to. Considering the environments in the meta-RL setting can change in reward functions and transition dynamics, we expect the context can fully distinguish between the two types of environmental changes with a unified learning objective. For this, we propose a simple yet effective context encoder $E_\phi$, jointly with generalized reward model $R_\psi$ and dynamics model $P_\omega$. We augment

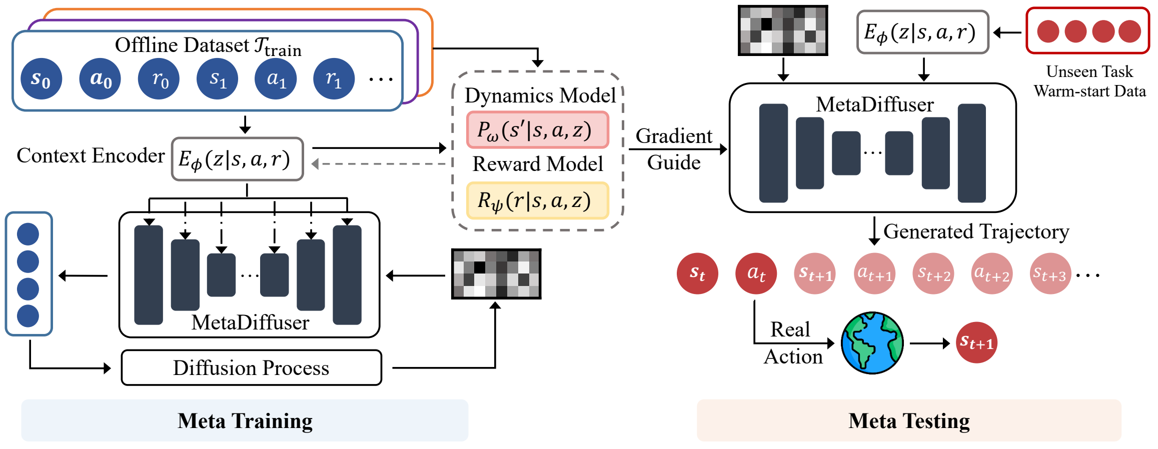


*Figure 2.* The overview of MetaDiffuser. During **meta-training** phase, a task-oriented context encoder is trained jointly with conditioned dynamics model and reward model in a self-supervised manner to infer the current task from the recent historical transitions. Then, the multi-task trajectories can be labeled with the trained context encoder and the inferred context are injected in the conditioned diffusion model to estimating the multi-modal distribution mixed by different training tasks. During **meta-testing** phase, context encoder can capture the task information from provided warm-start data from the test task. Then the conditioned diffusion model can manipulate the noise model to denoise out desired trajectories for the test task with the inferred context. Additionally, the pretrained dynamics model and reward model can serve as classifiers for evaluation, with gradient to guide the conditional generation in a classifier-guide fashion.

context into state and action to minimize the prediction loss of both dynamics and reward simultaneously.

Specifically, given the multi-task offline dataset $\mathcal{D}$, which contains the trajectories $\tau^{\mathcal{M}} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{K}$ with horizon $K$ for training task $\mathcal{M} \sim \mathcal{T}^{train}$. For each trajectories, a trajectories segment $\tau_t^{\mathcal{M}} = \{(s_{t+i}, a_{t+i}, r_{t+i}, s_{t+i+1})\}_{i=0}^{h}$ of size $h$ are sampled started from random selected $t$. With the historical sub-trajectories, the context encoder $E_\phi$ captures the latent representations $z_t = E_\phi(\tau_t^{\mathcal{M}})$ as the contextual information of the task. Then the generalized reward model $R_\psi$ and dynamics model $P_w$ are conditioned on $z$, parameterized with $\psi, \omega$. The context encoder is trained jointly by minimizing the state transition and reward prediction error conditioned on the learned context:

$$\mathcal{L}_{\phi,\psi,\omega} = -\mathbb{E}_{\tau_t^{\mathcal{M}}}\left[\log P_\omega(\hat{s}_{t+1}|s_t, a_t, z_t) + \log R_\psi(\hat{r}_t|s_t, a_t, z_t)\right] \tag{1}$$

Moreover, our method additionally trains the generalized reward model and dynamics model as byproducts, which will play a key role as a useful classifier in the later classifier-guided conditional generation module. It should be noted that our framework is flexible to other representation methods, the further analysis is illustrated in Section 5.5.

### 4.2. Conditional Diffusion Architecture

Inspired by the great success of the diffusion model in text-to-image tasks, which generates images based on text labels from noises, we leverage the diffusion model as a trajectory generator conditioned on the task-oriented context. Following Diffuser (Janner et al., 2022b), the states and actions in the trajectory are generated simultaneously per time step $t$ over the planning horizon $H$:

$$\boldsymbol{x}_k(\tau) = (s_t, a_t, s_{t+1}, a_{t+1}..., s_{t+H-1}, a_{t+H-1})_k \tag{2}$$

where $k$ denotes the timestep in the denoising process. Now we have the pre-trained context encoder to infer the task labels for different tasks, we can additionally condition the diffusion process on the contextual information of the tasks. In this way, we formulate the meta-RL problem as the conditional generative modeling problem:

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau\sim\mathcal{D}}[\log p_\theta(\boldsymbol{x}_0(\tau)|\boldsymbol{y} = E_\phi(\tau))] \tag{3}$$

where the conditional label $y$ denotes the task-oriented context inferred from the pre-collected offline data from the current task by context encoder $E_\phi$. The goal is to estimate the conditional data distribution with $p_\theta$ so we can later generate desired trajectory $\boldsymbol{x}_0(\tau)$ according to the context label from unseen tasks. The forward diffusion process $q$ and the reverse denoising process $p_\theta$ can be formulated as:

$$q(\boldsymbol{x}_{k+1}(\tau)|\boldsymbol{x}_k(\tau)), \quad p_\theta(\boldsymbol{x}_{k-1}(\tau)|\boldsymbol{x}_k(\tau), \boldsymbol{y} = E_\phi(\tau))) \tag{4}$$

Specifically, for each trajectory $\tau$ in the training offline dataset, we first sample a Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and a denoising timestep $k \in \{1, \ldots, K\}$. Then we construct a noisy array with the same dimension of $\boldsymbol{x}_k(\tau)$ and finally predict the denoising noise as $\hat{\epsilon}_\theta = \epsilon_\theta(\boldsymbol{x}_k(\tau), \boldsymbol{y}(\tau), k)$ in the denoising step $k$ .

For the classifier-free conditioned diffusion model (Ho & Salimans, 2022), the commonly used technique is to randomly drop out the conditioning for improving the quality of generated samples. Intuitively, we also train the noise model jointly with a single diffusion model on conditional and unconditional objective via randomly dropping the conditioning context label with probability $\beta$. The proper drop probability can balance off the diversity and the relevance of the context label of generated trajectories. The detailed analysis about the effects of different context drop probability can be found in Section 5.6.3.

So far, with the mixed trajectories datasets $\mathcal{D}$ paired with contextual information of the task it belongs to, we can train the reverse denoising process $p_\theta$, parameterized through the conditional noise model $\epsilon_\theta$ with the following loss:

$$\mathcal{L}(\theta)=\mathbb{E}_{k,\tau\in\mathcal{D}}\left[\|\epsilon-\epsilon_\theta\left(\boldsymbol{x}_k\left(\tau\right),(1-\beta)E_\phi\left(\tau\right)+\beta\varnothing,k\right)\|^2\right] \tag{5}$$

After training a conditioned diffusion model for imitating expert trajectories in the offline datasets, we now discuss how to utilize the trained diffusion model to achieve the generalization across unseen tasks. During meta-testing, the context encoder captures the task information context from pre-collected trajectories as warm-start data and infer the task-oriented context as the conditional label $\boldsymbol{y}=E_\phi(\tau)$. Then the context label can be injected into conditioned diffusion model to guide the desired expert trajectory generation for the current task. $\boldsymbol{x}_0(\tau)$ is sampled by starting with Gaussian noise $\boldsymbol{x}_K(\tau)$ and refining $\boldsymbol{x}_k(\tau)$ into $\boldsymbol{x}_{k-1}(\tau)$ at each intermediate timestep with the perturbed noise:

$$\hat{\epsilon}=\omega\epsilon_\theta(\boldsymbol{x}_k(\tau),\boldsymbol{y},k)+(1-\omega)\epsilon_\theta(\boldsymbol{x}_k(\tau),\varnothing,k) \tag{6}$$

where the scalar $\omega$ denotes the guidance weight in the classifier-free conditioned diffusion model. Setting $\omega=1$ disables classifier-free guidance while increasing $\omega>1$ strengthens the effect of guidance. Based on the context-conditioned noise generated iteratively, the desired trajectories containing future states and actions can be denoised from the noise step-wisely. With the generated trajectories, the first action will be applied to execute in the environment to step into the next state. This procedure repeats in a standard receding-horizon control loop, similar to traditional planning in RL, described in Appendix D. For architecture details, please refer to Appendix H.

### 4.3. Dual-guide Enhanced Planner

Previous work (Janner et al., 2022a) trains an extra reward predictor $\mathcal{J}$ to evaluate the accumulative return of generated trajectories and utilizes the gradient of return as a guidance in the sampling process of diffusion model, to encourage the generated trajectories to achieve high return. However, during meta-testing for unseen tasks, as shown in the top part of Figure 3, the conditional generated trajectories may not always obey dynamics constraints due to the aggressive guidance aim for high return, making it difficult for the planner to follow the expected trajectories during the interaction with the environment. Therefore, we propose a dual-guide to enhance the dynamics consistency of generated trajectories while encouraging the high return $\mathcal{J}$ simultaneously.

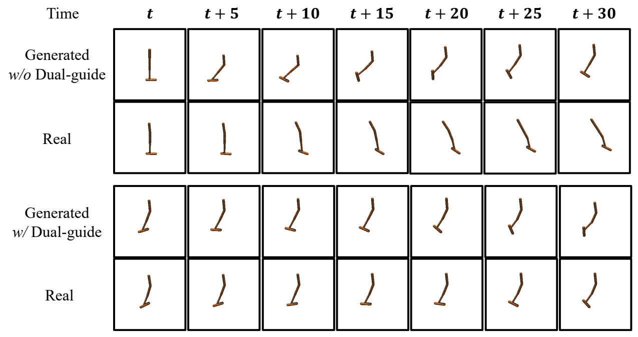


*Figure 3.* The visualization of an extreme case about generated trajectories and real trajectories rollout according to actions within generated trajectories in Hopper-Param, as an environment with dramatic dynamics changes. With dual-guide, the generated trajectories are less aggressive in expected rewards and more dynamics consistent to enhance the reachability between adjacent states.

For this, we utilize the previously pretrained dynamics model, to predict the future state of the generated trajectory based on its planned actions, then compared it to the states in the generated trajectory. The dynamics discrepancy $\zeta$ serves as an important metric to evaluate the consistency and reachability of the generated trajectory. Then the gradient from dual-guide can be formulated as:

$$\begin{aligned}
&g=\nabla\mathcal{J}(\boldsymbol{x}_k(\tau))+\lambda\nabla\zeta(\boldsymbol{x}_k(\tau))\\
&\mathcal{J}\left(\boldsymbol{x}_k(\tau)\right)=\sum_{t=0}^{T}R_\psi\left(s_t,a_t,z_t\right)\\
&\zeta\left(\boldsymbol{x}_k(\tau)\right)=\sum_{t=0}^{T}\|s_{t+1}-P_\omega\left(\hat{s}_{t+1}\mid s_t,a_t,z_t\right)\|^2
\end{aligned} \tag{7}$$

where $\lambda$ denotes the relative scaling coefficient between the dynamics guide and reward guide to balance off the high reward and low discrepancy. The detailed ablation study about the scaling effect can be found in Section 5.6.1. The visualization of an intuitive example is shown in Figure 3.

In this way, incorporate MetaDiffuser not only conducts the classifier-free manner by injecting the context as label $\boldsymbol{y}$ into the conditional noise model $\epsilon_\theta(\boldsymbol{x}_k,\boldsymbol{y},k)$, achieving more precise conditional generation, but also incorporate the classifier-guide fashion in Diffuser where the single reward guide is expanded to designed dual-guide for more complex environment change in meta-RL setting. Formally, the denoising process in Equation (6) can be extended as:

$$\begin{aligned}
\hat{\epsilon}:=&\underbrace{\omega\epsilon_\theta(\boldsymbol{x}_k(\tau),E_\phi(\tau),k)+(1-\omega)\epsilon_\theta(\boldsymbol{x}_k(\tau),\varnothing,k)}_{\text{classifier-free}}\\
&\underbrace{-\sqrt{1-\bar{\alpha}_t}\nabla_{\boldsymbol{x}_k(\tau)}\Big[\mathcal{J}(\boldsymbol{x}_k(\tau))+\lambda\zeta(\boldsymbol{x}_k(\tau))\Big]}_{\text{classifier-guided}}
\end{aligned} \tag{8}$$

The details about the relationship between two different conditional fashions can be found in Appendix A.

*Table 1.* Average test returns of MetaDiffuser against other baselines with a few-shot manner.

| Environment | FOCAL | CVAE-Planner | CORRO | Prompt-DT | MetaDiffuser | Oracle |
|---|---|---|---|---|---|---|
| **Point-Robot** | -5.99±0.26 | -6.11±0.71 | -5.59±0.57 | -5.04 ±0.35 | **-4.48**±0.28 | -3.97±0.12 |
| **Ant-Dir** | 151.3±24.6 | 130.6±41.3 | 193.3±32.1 | 213.2±29.1 | **247.7**±16.8 | 314.4±9.2 |
| **Cheetah-Dir** | 680.9±46.6 | 759.4±47.2 | 823.5±37.0 | 931.7±21.3 | **936.2**±17.9 | 943.4±15.4 |
| **Cheetah-Vel** | -82.5±7.0 | -87.1±13.1 | -56.2±9.4 | -51.3±4.9 | **-45.9**±4.1 | -32.4±1.8 |
| **Walker-Param** | 245.6±37.8 | 187.9±49.7 | 300.5±34.2 | 287.7 ±32.1 | **368.3**±30.6 | 447.2 ±9.7 |
| **Hopper-Param** | 203.6±46.6 | 157.3±54.5 | 289.3±24.7 | 265.2±37.1 | **356.4**±16.9 | 429.6±13.1 |

## 5. Experiments

We conduct experiments on various tasks to evaluate the few-shot generalization performance of the proposed MetaDiffuser. We aim to empirically answer the following questions: 1) Can MetaDiffuser achieve performance gain on few-shot policy **generalization** compared to other strong baselines? 2) Can MetaDiffuser show **robustness** to the quality of warm-start data? 3) Can MetaDiffuser be a **flexible** framework to incorporate with any context representation method?

### 5.1. Environments Settings

We adopt a 2D navigation environment Point-Robot and multi-task MuJoCo control tasks to make comparisons, as classical benchmarks commonly used in meta-RL (Mitchell et al., 2021b; Li et al., 2020; 2021a). More details about environments are available in Appendix E. For each environment, different tasks are randomly sampled from the task distribution, divided into a training set $\mathcal{T}^{train}$ and testing set $\mathcal{T}^{test}$. On each task, we use SAC (Haarnoja et al., 2018) to train a single-task policy independently. The trajectories of expert policy for each task are collected to be the offline datasets. See more details in Appendix G.

### 5.2. Baselines

**FOCAL** (Li et al., 2021b) proposes a novel negative-power distance metric learning method to train the context encoder for task inference, as an end-to-end offline meta-RL algorithm with high efficiency.

**CORRO** (Yuan & Lu, 2022) proposes a contrastive learning framework for task representations that are robust to the distribution mismatch of behavior policies in training and testing. CORRO demonstrates superior performance than prior context-conditioned policy-based methods.

**Prompt-DT** (Xu et al., 2022) leverages the sequential modeling ability of the Transformer architecture and the prompt framework to achieve few-shot adaptation in offline RL, as a strong meta-RL baseline in sequence modeling fashion.

**CVAE-Planner** To investigate the influence of different generative architectures, we substitute the conditioned diffusion to conditioned VAE, serving the same role as a trajectory generator to guide the planning across tasks.

### 5.3. The Generalization Ability on Task Adaptation

To evaluate the performance on task adaptation, we sample tasks from the test set with warm-start data, which is pre-collected by a random policy or an expert policy. Then we measure the few-shot generalization ability of different methods with the average episode accumulated reward. For fairness, all methods are trained with the same expert dataset in each environment to investigate whether the diffusion model facilitates few-shot generalization and the performance of MetaDiffuser. The testing curves and converged performance are summarized in Figure 4 and Table 1 respectively, which contain six environments varying in dynamics and rewards. In relatively simple environments such as Point-Robot and Cheetah-Dir, MetaDiffuser and Prompt-DT significantly outperforms other baselines to a large extent. In Ant-Dir MetaDiffuser outperforms other baselines by a large margin, which show the strong generalization ability in unseen task with different reward functions. Moreover, in Cheetah-Vel, MetaDiffuser is more data-efficient to achieve better asymptotic performance than others, with the benefit of the strong generative capacity of the diffusion model. In dynamics change environments, such as Hopper-Param and Walker-Param, CORRO, as a context-based method, can have a more stable improvement than Prompt-DT. The potential reason may be that the complex environment varying in dynamics is more challenging for Prompt-DT to implicitly capture the dynamics information within a prompt.

MetaDiffuser can outperform CORRO benefiting from the stability of the sequence-modeling framework instead of TD-learning. The detailed analysis of the context representation method can be found in Section 5.5.The CVAE-Planner struggles to generalize to different tasks, illustrating the strong modeling capability of the diffusion model against CVAE, when meeting with the extreme multi-modal distribution. We will illustrate the detailed analysis in Section 5.6.2.

### 5.4. The Robustness of Warm-start Data Quality

Benefiting from the context encoder and the manner of injecting the explicit context as a label into the diffusion model to conduct the conditional generation, MetaDiffuser is robust to the quality of warm-start data, similar to traditional context-based methods like CORRO. Prompt-DT is sensi-

*Table 2.* The comparisons of the performance of Prompt-DT and MetaDiffuser with different qualities of provided warm-start data during the meta-testing phase. The ↓ denotes the performance drop with other quality of data.

| Environment | MetaDiffuser | | | Prompt-DT | | |
|---|---|---|---|---|---|---|
| | *Expert* | *Medium* | *Random* | *Expert* | *Medium* | *Random* |
| **Point-Robot** | **-4.48**±0.28 | -4.54±0.31 (↓ 1.5%) | -4.61±0.21 (↓ 3.2%) | **-5.04**±0.35 | -5.17±0.29 (↓ 3.8%) | -5.85±0.32 (↓ 23.4%) |
| **Ant-Dir** | **247.7**±16.8 | 238.9±18.1 (↓ 3.6%) | 213.8±26.5 (↓ 13.7%) | **213.2**±29.1 | 154.7±39.5 (↓ 27.4%) | 40.1±16.3 (↓ 81.2%) |
| **Cheetah-Dir** | **936.2**±17.9 | 930.3±18.5 (↓ 0.6%) | 916.7±21.8 (↓ 1.9%) | **931.7**±21.3 | 922.6±28.2 (↓ 1.0%) | 913.9±30.8 (↓ 1.9%) |
| **Cheetah-Vel** | **-45.9**±4.1 | -50.2±5.2 (↓ 1.9%) | -55.8±2.3 (↓ 4.4%) | **-51.3**±4.9 | -125.6±7.5 (↓ 33.2%) | -208.4±1.9 (↓ 76.1%) |
| **Walker-Param** | **368.3**±30.6 | 357.9±33.7 (↓ 2.8%) | 341.6±38.4 (↓ 7.2%) | **287.7**±32.1 | 200.1±26.3 (↓ 30.4%) | 64.7±8.1 (↓ 77.5%) |
| **Hopper-Param** | **356.4**±16.9 | 337.0±21.2 (↓ 5.4%) | 319.6±14.2 (↓ 10.3%) | **265.2**±37.1 | 159.6±35.7 (↓ 39.8%) | 82.6±15.3 (↓ 68.9%) |

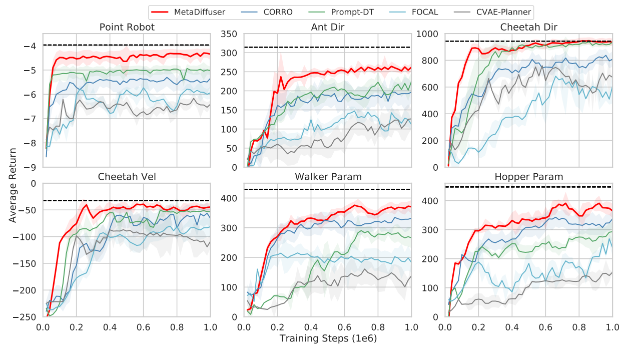


*Figure 4.* **Meta-testing average performance** of MetaDiffuser against baselines run over five random seeds in unseen tasks . The dashed lines denote the oracle performance of expert policy trained separately for each test task.

tive to the quality of prompt and the performance can drop a lot with the middle or random prompt, also mentioned in the original paper (Xu et al., 2022). We conduct a more detailed experiment to investigate the robustness of two algorithms.

The results in Table 2 show that when the quality of prompt data is not high enough, the performance of Prompt-DT will drop by a large extent except for Cheetah-Dir. This environment contains just two tasks forward and backward, both concluded in the training set and testing set, potentially decreasing the reliance on expert warm-start data. The performance of MetaDiffuser may also experience a slight drop, but still superior to Prompt-DT. The slight drop may be caused by the distribution shift exhibited by poor quality warm-start data during meta-testing and expert data during the pre-trained stage of context encoder, resulting in the inferred context being less accurate. For Prompt-DT, the prompt as the prefix to guide the subsequent sequence generation should contain enough valuable knowledge about how to solve the current task, not just information about what the current task is. But MetaDiffuser has no strict demand with the quality of warm-start data and even can be rollout with any arbitrary policy. The role of warm-start data is just to provide the task-oriented information to the context encoder can infer the task context as the label and then be injected in the conditional denoising process to generate the desired trajectories for planning to fast adaption.

## 5.5. The Flexibility in Context Representation Method

The generalization ability of MetaDiffuser arises from capturing task information as context to guide the diffusion model to conditional generation. We argue that our framework can flexibly integrate different task representation algorithm, and the improvement of context accuracy can enhance the generalization performance. We conduct experiments to investigate the effect of different context representations on the few-shot generalization capability of MetaDiffuser.

*Table 3.* The comparisons of the influences of different context representation methods on generalization ability to unseen task.

| Environment | CORRO | Ours | Ours+CORRO | Ours+GT |
|---|---|---|---|---|
| **Point-Robot** | -5.59±0.57 | -4.48±0.28 | -4.43±0.26 | **-4.02**±0.13 |
| **Ant-Dir** | 193.3±32.1 | 247.7±16.8 | 251.3±17.2 | **282.9**±13.6 |
| **Cheetah-Dir** | 823.5±37.0 | 936.2±17.9 | 936.9±18.1 | **939.7**±15.7 |
| **Cheetah-Vel** | -56.2±9.4 | -45.9±4.1 | -44.6±3.9 | **-41.1**±3.2 |
| **Walker-Param** | 300.5±34.2 | 368.3±30.6 | 377.0±29.6 | **394.1**±17.5 |
| **Hopper-Param** | 289.3±24.7 | 356.4±16.9 | 361.3±19.2 | **382.5**±12.0 |

To this end, we borrowed the representation module of CORRO and integrated it into MetaDiffuser, shown as Ours+CORRO in Table 3, resulting in a slight improvement. The performance gain demonstrates that the powerful generalization ability of MetaDiffuser is not achieved by improving context representation capability. The simple representation method we design is not better than the fine-grained representation trained in contrastive learning manner used in CORRO. Considering the combination of CORRO representation with MetaDiffuser can earn a large performance gain than the original conditioned policy manner, conditional sequence modeling shows great potential as a promising paradigm for generalization tasks.

Although we do not seek improvement in generalization performance through a more complicated context representation design in this paper, the incorporation of a more accurate context representation method is always encouraged. The significant improvement in incorporating ground truth parameters of tasks as context into MetaDiffuser demonstrates that there is still rich room for improvement in the integration of context method, shown as Ours+GT.

## 5.6. Ablation Study

### 5.6.1. The Effect of Dual-guide

For meta-testing for unseen tasks, the real trajectory rollout with actions in generated trajectory often deviates greatly from the expected trajectory, especially when meeting with a dynamics change environment. Here we conduct a detailed ablation study to demonstrate the importance of dual-guide for meta-RL setting and gain performance improvements of different relative scaling coefficients between reward guide and dynamics guide in all environments. The visualization in Hopper-Param is shown in Figure 3 and the results are illustrated in Table 4. The utilization of dual-guide can greatly enhance the feasibility and also encourage the high value of generated trajectories when the tasks shift dramatically. In relatively simple environments such as Point-Robot or environments with limited task numbers such as Cheetah-Dir, overly large dynamics guides can cause diffusion models to generate trajectories that are too conservative and lack a high value to guide. We also tried to omit the value guide and solely utilize the dynamics guide, and found that it yielded relatively poor performance for the same reason.

*Table 4.* Ablation of dual-guide and relative scaling coefficient.

| Environment | $\lambda = 0$ | $\lambda = 0.5$ | $\lambda = 1$ | $\lambda = 2$ |
|---|---|---|---|---|
| **Point-Robot** | -4.57±0.33 | **-4.48**±0.28 | -4.74±0.26 | -4.89±0.27 |
| **Ant-Dir** | 210.3±10.7 | 214.6±19.8 | **247.7**±16.8 | 238.3±18.1 |
| **Cheetah-Dir** | 924.2±19.6 | **936.2**±17.9 | 929.7±15.1 | 916.0±19.8 |
| **Cheetah-Vel** | -52.9±4.72 | -49.9±2.95 | **-45.9**±4.1 | -48.6±3.75 |
| **Walker-Param** | 326.5±24.9 | 330.6 ±23.4 | 347.2 ±19.3 | **368.3**±30.6 |
| **Hopper-Param** | 293.3±13.8 | 307.2±18.6 | 328.1±16.7 | **356.8**±16.9 |

### 5.6.2. The comparisons of Generative Models

To investigate the importance of the conditional diffusion model in MetaDiffuser, we substitute the conditioned diffusion model to conditioned VAE as the same role of trajectory generator to guide the planning across tasks, named as CVAE-Planner. For fairness, the length of generated trajectories and the planning procedure with samples keep the same with MetaDiffuser. The results of the experiment are shown in Table 5, demonstrating that the fitting capability of CVAE is significantly inferior to the conditional diffusion model, struggling to generate reasonable trajectories for unseen tasks. Moreover, compared to the end-to-end generative paradigm of CVAE, MetaDiffuser can fully utilize the gradient from dual-guide during the step-wise iterative denoising process. Additionally, we also trained an unconditional diffusion model over mixed expert data on all the training tasks, named as UDiffuser. UDiffuser struggles to model such a diverse distribution of data and fails to generate the desired trajectories for specific tasks for the lack of ability to infer what the testing task is.

*Table 5.* The comparisons between different generative models on generalization ability to unseen task.

| Environment | CVAE-Planner | UDiffuser | MetaDiffuser |
|---|---|---|---|
| **Point-Robot** | -6.11±0.71 | -7.48±0.89 | **-4.48**±0.28 |
| **Ant-Dir** | 130.6±41.3 | 53.2±51.1 | **247.7**±16.8 |
| **Cheetah-Dir** | 759.4±47.2 | 614.7±63.5 | **936.2**±17.9 |
| **Cheetah-Vel** | -87.1±13.1 | -121.7±36.2 | **-45.9**±4.1 |
| **Walker-Param** | 187.9±49.7 | 135.2±78.1 | **368.3**±30.6 |
| **Hopper-Param** | 157.3±54.5 | 103.5±64.0 | **356.4**±16.9 |

### 5.6.3. The Effect of Context Drop Probability

The proper context drop probability can balance off the diversity and the relevance of the conditional label of generated samples (Ho & Salimans, 2022). We conduct an ablation study with the aim to investigate the effect of context drop probability in the training of the conditional diffusion model. When $\beta$ reaches 1, MetaDiffuser devolves into the unconditional version previously mentioned as UDiffuser. The results in Table 6 shows that removing conditional context with a proper probability can improve generalization ability, but the best probability differs from environments. One possible explanation for this could be the varying levels of information sharing among tasks in different environments, as an interesting future work.

*Table 6.* Ablation of context drop probability.

| Environment | $\beta = 0$ | $\beta = 0.1$ | $\beta = 0.2$ | $\beta = 0.3$ |
|---|---|---|---|---|
| **Point-Robot** | -4.86±0.22 | -4.61±0.34 | -4.71±0.30 | **-4.48**±0.28 |
| **Ant-Dir** | 234.9±12.8 | 241.3±16.6 | **247.7**±16.8 | 224.8±25.3 |
| **Cheetah-Dir** | **936.2**±17.9 | 915.4±19.8 | 909.6±23.1 | 873.8±28.6 |
| **Cheetah-Vel** | -48.3±2.7 | -49.8±3.5 | -47.4±3.7 | **-45.9**±4.1 |
| **Walker-Param** | 346.5±31.4 | 349.6 ±35.7 | **368.3**±30.6 | 357.8 ±29.0 |
| **Hopper-Param** | 347.1±17.3 | **356.8**±16.9 | 336.9±12.4 | 343.3±18.6 |

## 6. Conclusion

This paper proposes MetaDiffuser, a novel framework for offline meta-RL, which leverages the diffusion model to conduct conditional trajectory generation to achieve the generalization ability across unseen tasks. By combining the context representation module with a task-oriented conditional diffusion model to generate the desired trajectories for unseen tasks, MetaDiffuser demonstrates that the conditional diffusion model can be a promising backbone for offline meta-RL. Moreover, we design the dual-guide to improve the quality of generated trajectories in the sampling process, ensuring dynamics transition consistency with the real world while encouraging the generated trajectories to achieve high returns. The experiments on various benchmarks empirically show that MetaDiffuser much better generalizes to unseen tasks than prior methods, while also enjoying the flexibility to incorporate with other task representation methods and the robustness to the quality of collected warm-start data at the testing task.

## References

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J., Jaakkola, T., and Agrawal, P. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Dorfman, R., Shenfeld, I., and Tamar, A. Offline Meta Learning of Exploration. *arXiv preprint arXiv:2008.02598*, 2020.

Ebert, F., Finn, C., Dasari, S., Xie, A., Lee, A., and Levine, S. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with A Stochastic Actor. In *International Conference on Machine Learning*, 2018.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34: 1273–1286, 2021.

Janner, M., Du, Y., Tenenbaum, J., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022a.

Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022b.

Kenton, J. D. M.-W. C. and Toutanova, L. K. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT*, 2019.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. MOReL: Model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Li, J., Vuong, Q., Liu, S., Liu, M., Ciosek, K., Ross, K., Christensen, H. I., and Su, H. Multi-Task Batch Reinforcement Learning with Metric Learning. In *International Conference on Learning Representations*, 2020.

Li, L., Huang, Y., Chen, M., Luo, S., Luo, D., and Huang, J. Provably Improved Context-Based Offline Meta-RL with Attention and Contrastive Learning. *arXiv preprint arXiv:2102.10774*, 2021a.

Li, L., Yang, R., and Luo, D. FOCAL: Efficient Fully-Offline Meta-Reinforcement Learning Via Distance Metric Learning and Behavior Regularization. In *International Conference on Learning Representations*, 2021b.

Liu, Z. et al. Swin transformer: Hierarchical vision transformer using shifted windows. *Proceedings of the IEEE/CVF ICCV*, 2021.

Misra, D. Mish: A self regularized non-monotonic neural activation function. In *British Machine Vision Conference*, 2019.

Mitchell, E., Rafailov, R., Peng, X. B., Levine, S., and Finn, C. Offline Meta-Reinforcement Learning with Advantage Weighting. In *International Conference on Machine Learning*, 2021a.

Mitchell, E., Rafailov, R., Peng, X. B., Levine, S., and Finn, C. Offline meta-reinforcement learning with advantage weighting. In *International Conference on Machine Learning*, pp. 7780–7791. PMLR, 2021b.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, 2021.

Rakelly, K., Zhou, A., Quillen, D., Finn, C., and Levine, S. Efficient Off-Policy Meta-Reinforcement Learning Via Probabilistic Context Variables. In *International Conference on Machine Learning*, 2019.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine Learning*, 1988.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Wang, K., Zhao, H., Luo, X., Ren, K., Zhang, W., and Li, D. Bootstrapped transformer for offline reinforcement learning. *arXiv preprint arXiv:2206.08569*, 2022a.

Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022b.

Wen, M., Kuba, J. G., Lin, R., Zhang, W., Wen, Y., Wang, J., and Yang, Y. Multi-agent reinforcement learning is a sequence modeling problem. *Advances in Neural Information Processing Systems*, 2022.

Wu, Y. and He, K. Group normalization. In *European Conference on Computer Vision*, 2018.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Xu, M., Shen, Y., Zhang, S., Lu, Y., Zhao, D., Tenenbaum, J., and Gan, C. Prompting decision transformer for few-shot policy generalization. In *International Conference on Machine Learning*, pp. 24631–24645. PMLR, 2022.

Yuan, H. and Lu, Z. Robust task representations for offline meta-reinforcement learning via contrastive learning. In *International Conference on Machine Learning*, pp. 25747–25759. PMLR, 2022.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. Scaling vision transformers. *CoRR*, abs/2106.04560, 2021.

## A. Classifier-Free and Classifier-Guided Diffusion Model

In this section, we introduce the details of the theoretical analysis of the conditional diffusion model. The equivalence between diffusion models and score-matching (Song et al., 2021), which shows $\epsilon_\theta(\boldsymbol{x}_k, k) \propto \nabla_{\boldsymbol{x}_k} \log p(\boldsymbol{x}_k)$, naturally leads to two kinds of methods for conditioning: classifier-guided (Nichol & Dhariwal, 2021) and classifier-free (Ho & Salimans, 2022).The classifier-guided improves sample quality while reducing diversity in conditional diffusion models using gradients from a pre-trained classifier $p_\phi(\boldsymbol{y}|\boldsymbol{x}_k)$ during sampling. The classifier-free is an alternate technique that avoids this pre-trained classifier by instead jointly training a single diffusion model on conditional $\epsilon_\theta(\boldsymbol{x}_k, \boldsymbol{y}, k)$ and unconditional $\epsilon_\theta(\boldsymbol{x}_k, k)$ noise model via randomly dropping conditional label $\boldsymbol{y}$.

### A.1. Classifier-Guided Fashion

First, let us start with classifier-guided fashion. The initial conditional distribution that conditions on the respective label $\boldsymbol{y}$ can be formulated by Bayes rule as:

$$p\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{y}\right)=\frac{p\left(\boldsymbol{x}_{k-1}\right) p\left(\boldsymbol{y} \mid \boldsymbol{x}_{k-1}\right)}{p(\boldsymbol{y})} \tag{9}$$

where $k$ denotes the timestep of the denoising process.The most important advantage of the classifier guide is that it can reuse previously trained unconditional generation models $p\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k\right)$. By training an additional classifier $p\left(\boldsymbol{y} \mid \boldsymbol{x}_{k-1}\right)$ on the generated samples, its evaluation about the generated samples can be used as a gradient to guide the noise model during denoising process. Now we consider how to incorporate the unconditional diffusion model into the conditional diffusion model. With the additional condition of the current noisy sample $\boldsymbol{x}_k$, Eq.(9) can be rewritten as:

$$\begin{aligned} p\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k, \boldsymbol{y}\right) &= \frac{p\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k\right) p\left(\boldsymbol{y} \mid \boldsymbol{x}_{k-1}, \boldsymbol{x}_k\right)}{p\left(\boldsymbol{y} \mid \boldsymbol{x}_k\right)} \\ &= \frac{p\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k\right) p\left(\boldsymbol{y} \mid \boldsymbol{x}_{k-1}\right)}{p\left(\boldsymbol{y} \mid \boldsymbol{x}_k\right)} \\ &= p\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_k\right) e^{\log p\left(\boldsymbol{y} \mid \boldsymbol{x}_{k-1}\right)-\log p\left(\boldsymbol{y} \mid \boldsymbol{x}_k\right)} \end{aligned} \tag{10}$$

It is worth noting that $\boldsymbol{x}_k$ are obtained by diffusing over $\boldsymbol{x}_{k-1}$ with noise, which is not helpful for classifier evaluation. So we can assume $p\left(\boldsymbol{y} \mid \boldsymbol{x}_{k-1}, \boldsymbol{x}_k\right)=p\left(\boldsymbol{y} \mid \boldsymbol{x}_{k-1}\right)$. When the diffusing steps are large enough, the difference between $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k-1}$ is tiny. So we apply Taylor's Formula to the exponent term in Eq.(10) and can get:

$$\log p\left(\boldsymbol{y} \mid \boldsymbol{x}_{k-1}\right)-\log p\left(\boldsymbol{y} \mid \boldsymbol{x}_k\right) \approx\left(\boldsymbol{x}_{k-1}-\boldsymbol{x}_k\right) \cdot \nabla_{\boldsymbol{x}_k} \log p\left(\boldsymbol{y} \mid \boldsymbol{x}_k\right) \tag{11}$$

For another term, $p\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k\right)$, also as the unconditional diffusion model, can be rewritten in form of distribution:

$$p\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k\right)=\mathcal{N}\left(\boldsymbol{x}_{k-1} ; \boldsymbol{\mu}\left(\boldsymbol{x}_k\right), \sigma_k^2 \boldsymbol{I}\right) \propto e^{-\left\|\boldsymbol{x}_{k-1}-\boldsymbol{\mu}\left(\boldsymbol{x}_k\right)\right\|^2 / 2 \sigma_k^2} \tag{12}$$

where the mean $\mu$ and variance $\sigma$ denotes the mean and variance of the Gaussian distribution respectively. With the above Eq.(11) and Eq.(12), we have:

$$\begin{aligned} p\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k, \boldsymbol{y}\right) &\propto e^{-\left\|\boldsymbol{x}_{k-1}-\boldsymbol{\mu}\left(\boldsymbol{x}_k\right)\right\|^2 / 2 \sigma_k^2+\left(\boldsymbol{x}_{k-1}-\boldsymbol{x}_k\right) \cdot \nabla_{\boldsymbol{x}_k} \log p\left(\boldsymbol{y} \mid \boldsymbol{x}_k\right)} \\ &\propto e^{-\left\|\boldsymbol{x}_{k-1}-\boldsymbol{\mu}\left(\boldsymbol{x}_k\right)-\sigma_k^2 \nabla_{\boldsymbol{x}_k} \log p\left(\boldsymbol{y} \mid \boldsymbol{x}_k\right)\right)\|^2 / 2 \sigma_k^2} \end{aligned} \tag{13}$$

Based on this proportional property, now we can obtain the classifier-guided conditioned diffusion model $p\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k, \boldsymbol{y}\right)$:

$$\begin{aligned} & p\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k, \boldsymbol{y}\right) \approx \mathcal{N}\left(\boldsymbol{x}_{k-1} ; \boldsymbol{\mu}\left(\boldsymbol{x}_k\right)+\sigma_k^2 \nabla_{\boldsymbol{x}_k} \log p\left(\boldsymbol{y} \mid \boldsymbol{x}_k\right)\right), \sigma_k^2 \boldsymbol{I}) \\ & \Rightarrow \boldsymbol{x}_{k-1}=\boldsymbol{\mu}\left(\boldsymbol{x}_k\right)+\sigma_k^2 \nabla_{x_k} \log p\left(\boldsymbol{y} \mid x_k\right)+\sigma_k \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \end{aligned} \tag{14}$$

This can also be formulated as the denoising version with noise model $\hat{\epsilon}\left(\boldsymbol{x}_k\right)$:

$$\hat{\epsilon}\left(\boldsymbol{x}_k\right):=\epsilon_\theta\left(\boldsymbol{x}_k\right)-\sqrt{1-\bar{\alpha}_k} \nabla_{\boldsymbol{x}_k} \log p_\phi\left(\boldsymbol{y} \mid \boldsymbol{x}_k\right) \tag{15}$$

### A.2. Classifier-Free Fashion

Classifier-free fashion is relatively easy to understand compared with classifier-guided fashion. This manner does not separately train a classifier but modifies the original training setup to learn both a conditional $\epsilon_\theta(\boldsymbol{x}_k, \boldsymbol{y}, k)$ and an unconditional $\epsilon_\theta(\boldsymbol{x}_k, k)$ model for the noise.

Without extra classifier $p\left(\boldsymbol{y} \mid \boldsymbol{x}_{k-1}\right)$ to reuse, we can directly define the conditional data distribution $p\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k, \boldsymbol{y}\right)$ with conditional label $\boldsymbol{y}$:

$$\begin{gathered} p\left(\boldsymbol{x}_{k-1} \mid \boldsymbol{x}_k, \boldsymbol{y}\right) = \mathcal{N}\left(\boldsymbol{x}_{k-1}; \boldsymbol{\mu}\left(\boldsymbol{x}_k, \boldsymbol{y}\right), \sigma_k^2 \boldsymbol{I}\right) \\ \boldsymbol{\mu}\left(\boldsymbol{x}_k, \boldsymbol{y}\right) = \frac{1}{\alpha_k}\left(\boldsymbol{x}_k - \frac{\beta_k^2}{\bar{\beta}_k} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_k, \boldsymbol{y}, k\right)\right) \end{gathered} \tag{16}$$

where $\alpha_k, \bar{\beta}_k \in \mathbb{R}$ are carefully chosen for the variance schedule during the diffuse process. The noise model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ in the denoising process can be trained by minimizing the reconstruction error about noise, following as:

$$\mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{y} \sim \tilde{p}(\boldsymbol{x}_0, \boldsymbol{y}), \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})}\left[\left\|\boldsymbol{\varepsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\bar{\alpha}_k \boldsymbol{x}_0 + \bar{\beta}_k \boldsymbol{\varepsilon}, \boldsymbol{y}, k\right)\right\|^2\right] \tag{17}$$

It is worth noting that the conditional label is randomly dropped with probability $\beta$, following the classifier-free conditioned diffusion model (Ho & Salimans, 2022). The potential reason is the proper context drop probability can balance off the diversity and the relevance of the conditional label of generated samples. At testing-time, the perturbed noise is generated as:

$$\hat{\epsilon}_\theta\left(\boldsymbol{x}_k \mid \boldsymbol{y}\right) = \epsilon_\theta(\boldsymbol{x}_k, k) + \omega(\epsilon_\theta(\boldsymbol{x}_k, \boldsymbol{y}, k) - \epsilon_\theta(\boldsymbol{x}_k, k)) \tag{18}$$

where $\omega$ is referred to as the guidance scaling, similar to classifier-guided fashion. Setting $\omega = 1$ enables classifier-free guidance while increasing $\omega > 1$ strengthens the effect of guidance.

## B. Further Introduce of Sequence Modeling Fashion

Recently, much attention has been focused on the use of large, pre-trained big models on unsupervised datasets to improve results on downstream tasks through fine-tuning. In the field of natural language processing, transformer-based models such as BERT(Kenton & Toutanova, 2019) and GPT-3(Brown et al., 2020) have overcome the limitations of RNNs and improved the ability to use long sequence information, resulting in state-of-the-art performance on tasks such as translation and question answering systems. The field of computer vision has also been inspired by these developments, with models like the Swin Transformer(Liu et al., 2021) and Scaling ViT(Zhai et al., 2021) being proposed to address problems as sequence modeling problems.

Traditional offline RL approaches require fitting value functions or computing policy gradients, which are challenging due to limited offline data (Kumar et al., 2020; Wu et al., 2019; Kidambi et al., 2020). Inspired by the exciting progress of large generative models in vision and language tasks, researchers turn to model the trajectories in offline RL datasets through transformer-like structures. Recent advances in generative sequence modeling (Chen et al., 2021; Janner et al., 2021; 2022b) provide effective alternatives to conventional RL problems by modeling the joint distribution of sequences of states, actions, rewards, and values. For example, Decision Transformer (Chen et al., 2021) casts offline RL as a form of conditional sequence modeling, which allows more efficient and stable learning without the need to train policies via traditional RL algorithms like temporal difference learning (Sutton, 1988). Trajectory Transformer (Janner et al., 2021) is proposed to utilize transformer architecture to model distributions over trajectories, repurposes beam search as a planning algorithm, and shows great flexibility across long-horizon dynamics prediction, imitation learning, goal-conditioned RL, and offline RL. Bootstrapped Transformer (Wang et al., 2022a) further incorporates the idea of bootstrapping and uses the learned model to self-generate more offline data to further improve sequence model training. MAT(Wen et al., 2022) introduces sequence modeling into the online MARL setting and demonstrates high data efficiency in transfer. By treating RL as a sequence modeling problem, it bypasses the need for bootstrapping for long-term credit assignment, avoiding one of the "deadly triad" (Sutton & Barto, 2018) challenges in reinforcement learning.

The above methods mostly focus on Transformer-like architecture, and recent works begin to adopt the diffusion model as the backbone of sequence modeling. Diffuser (Janner et al., 2022b) applies a diffusion model as a trajectory generator, which is trained by diffusing over the full trajectory of state-action pairs from the noises. A separate reward model is trained to predict the cumulative rewards of each trajectory sample, then the gradient guidance from the reward model is injected into the reverse sampling stage. Then the first action in the generated trajectories will be applied to execute in the environment to step into the next state, which repeats in a loop until the terminal. The consequent work Decision Diffuser (Ajay et al., 2022) frames offline sequential decision-making as conditional generative modeling based on returns, constraints, and skills to eliminate the complexities in traditional offline RL.

## C. Additional Discussion of Offline Meta-RL

One of the inherent difficulties of offline RL is the challenge to generalize to unseen tasks. Recent work in offline meta-RL (Mitchell et al., 2021b; Li et al., 2020; 2021a;b) aims to solve this problem by training a meta-policy from multi-task offline datasets that can efficiently adapt to unseen tasks with small amounts of warm-start data.

The context-based offline meta RL methods pre-train a context encoder to learn task representation from the collected offline data and augment the state-action pair with latent representation to generalize across tasks. MACAW (Mitchell et al., 2021a) adopts the advantage weighting loss based on an optimization-based method and learns the initialization of both the value function and the policy. FOCAL (Li et al., 2021b) proposes a novel negative-power distance metric learning method to train the context encoder for task inference, as an end-to-end offline meta-RL algorithm with high efficiency. CORRO (Yuan & Lu, 2022) proposes a contrastive learning framework for task representations that are robust to the distribution mismatch of behavior policies in training and testing. CORRO demonstrates superior performance than prior context-conditioned policy-based methods.

The context-based methods can infer the current task from the pre-collected offline data as long as it contains enough information about the current task, regardless of the quality of data. But one of the limitations of these methods is the high requirements for the representation ability of the context. If the context is not sufficiently accurate, it will negatively impact the generalization performance of policy across tasks, which severely relies on the representation ability of context. Although many recent works (Li et al., 2020; Dorfman et al., 2020; Li et al., 2021a;b) focused on improving the representation accuracy of context, the methods still rely on TD-learning, which may potentially cause instability in context learning.

*Table 7.* A comparison on algorithmic properties of existing methods for offline meta-RL.

| Algorithm | Robustness | Flexibility | Sequence Modeling |
|---|---|---|---|
| MACAW (Mitchell et al., 2021a) | ✓ | ✓ | ✗ |
| FOCAL (Li et al., 2021b) | ✓ | ✓ | ✗ |
| CORRO (Yuan & Lu, 2022) | ✓ | ✓ | ✗ |
| Prompt-DT (Xu et al., 2022) | ✗ | ✗ | ✓ |
| MetaDiffuser (Ours) | ✓ | ✓ | ✓ |

A more recent work Prompt-DT (Xu et al., 2022) turns to solve the generalization problem from the sequence modeling perspective, which joint models state-action trajectories to avoid TD-learning. This approach does not require a pre-trained context encoder to extract the task representation from the prompt and the contained task-oriented information of the prompt is implicitly learned inside the generative model architecture based on powerful ability transformer-style models. The prompt as the prefix to guide the subsequent sequence generation should not only contain rich task information to allow the model to implicitly infer the current task but also has to conclude the expert trajectories as valuable knowledge to utilize. Therefore, the core limitation of the method is the quality of the pre-collected demonstration or prompt has to be high enough to guide the sequence generation or the performance will drop a lot with random or medium prompts. It is worth noting that we can not easily obtain expert trajectories on all tasks, which is challenging and computationally high in many scenarios. A more clear comparison of algorithmic properties of existing methods for offline meta-RL is shown in Table 7.

## D. Pesudocodes of Framework

**Algorithm 1** Task-Oriented Conditioned Diffusion Planner for Offline Meta-RL (MetaDiffuser)

**Input**: Training tasks set $\mathcal{T}^{train}$ and corresponding offline datasets $\mathcal{D}^{train}$, testing tasks set $\mathcal{T}^{test}$ and corresponding warm-start datasets $\mathcal{D}^{test}$, noise model $\epsilon_\theta$, guidance scale $s$, historical trajectory length $h$, planning horizon $H$, context drop probability $\beta$, context encoder $E_\phi$, reward predictor $R_\psi$, dynamics model $P_w$.

**Pre-training Context Encoder**

```
for each iteration do
    Sample a task $M \sim \mathcal{T}^{train}$ and corresponding trajectories $\tau^M$ from $\mathcal{D}^{train}$.
    Sample several historical transitions $\tau_t^{\mathcal{M}} = \{(s_{t+i}, a_{t+i}, r_{t+i}, s_{t+i+1})\}_{i=0}^{h}$ started from random selected $t$
    Predict the context from the historical transitions: $z_t = E_\phi(\tau_t^{\mathcal{M}})$
    Update $\phi$, $\psi$, $\omega$ on $\tau^M$ according the following loss:
    $\mathcal{L}_{\phi,\psi,\omega} = -\mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\tau_t^{\mathcal{M}};\mathcal{M}\sim\mathcal{T}^{train}}\left[\mathbb{E}_{z_t=E_\phi(z_t|\tau_t^{\mathcal{M}})}\left[\log P_\omega(\hat{s}_{t+1}|s_t,a_t,z_t) + \log R_\psi(\hat{r}_t|s_t,a_t,z_t)\right]\right]$
end for
```

**Training Task-Oriented Diffusion Model**

```
for each iteration do
    Sample trajectories with planning horizon $H$ as mini-batch $\mathcal{B} = \{\tau^{\mathcal{M}}; \mathcal{M} \sim \mathcal{T}^{train}\}$ from mixed $\mathcal{D}^{train}$.
    Predict the context $z_\tau$ from context encoder for each $\tau$ in $\mathcal{B}$
    Random sample a denosing timestep $k$ and omit the context with probability $\beta$
    Update the task-oriented conditioned diffusion model according the following loss:
    $\mathcal{L}(\theta) := \mathbb{E}_{k,\tau\in\mathcal{D},p\sim\text{Bern}(\beta)}\left[\left\|\epsilon - \epsilon_\theta\left(\boldsymbol{x}_k(\tau), (1-p)E_\phi(\tau) + p\varnothing, k\right)\right\|^2\right]$
end for
```

**Testing on Unseen Tasks**

```
for each iteration do
    Sample a task $M \sim \mathcal{T}^{train}$ and corresponding warm-start data $\tau^M$ from $\mathcal{D}^{train}$.
    Random sample a sub-segment of historical trajectories from $\tau^M$
    Infer the context for the current test task from context encoder capturing task-oriented information
    while not done do
        Observe state $s$ ; Initialize $\boldsymbol{x}_T(\tau) \sim \mathcal{N}(0, \alpha I)$
        for $t = T \dots 1$ do
            $\hat{\epsilon} := \underbrace{\omega\epsilon_\theta(\boldsymbol{x}_k(\tau), E_\phi(\tau), k) + (1-\omega)\epsilon_\theta(\boldsymbol{x}_k(\tau), \varnothing, k)}_{\text{classifier-free}} - \underbrace{\sqrt{1-\bar{\alpha}_t}\nabla_{\boldsymbol{x}_k(\tau)}\left[\mathcal{J}(\boldsymbol{x}_k(\tau)) + \lambda\zeta(\boldsymbol{x}_k(\tau))\right]}_{\text{classifier-guided}}$
            $(\mu_{t-1}, \Sigma_{t-1}) \leftarrow \text{Denoise}(\boldsymbol{x}_t(\tau), \hat{\epsilon})$
            $\boldsymbol{x}_{t-1} \sim \mathcal{N}(\mu_{t-1}, \alpha\Sigma_{t-1})$
        end for
        Execute first action from $\boldsymbol{x}_0$ as the current action to interact with environments.
    end while
end for
```

## E. The Details of Environments

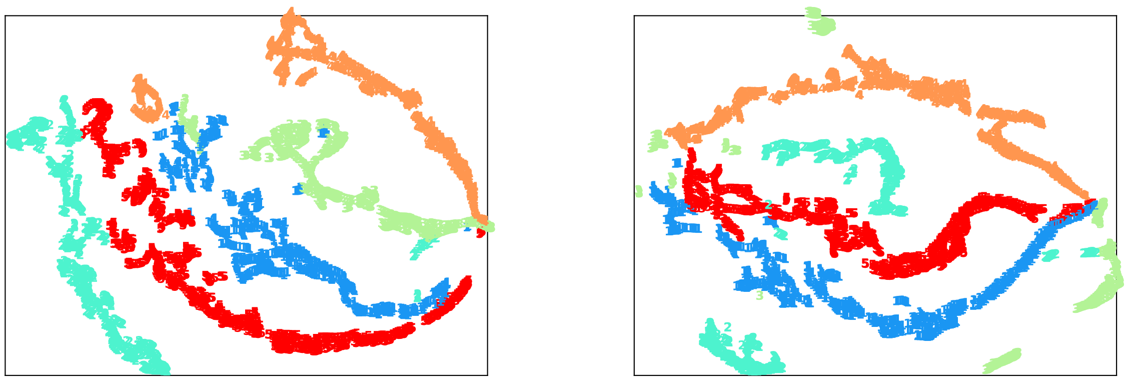
We adopt a 2D navigation environment Point-Robot and multi-task MuJoCo control tasks to make comparisons, as classical benchmarks commonly used in meta-RL (Mitchell et al., 2021b; Li et al., 2020; 2021a). The environments conclude 4 tasks with reward function changes and 2 tasks with transition dynamics changes.

- **Point-Robot** is a 2D navigation environment introduced in Rakelly et al. (2019). Starting from the initial point, the agent should navigate to the goal location. Tasks differ in reward functions, which describe the goal position. The goal positions are uniformly distributed in a square. The maximal episode step is set to 20.

- **Cheetah-Vel, Cheetah-Dir, Ant-Dir** are multi-task MuJoCo benchmarks where tasks differ in reward functions. In Cheetah-Vel, the task is specified by the target velocity of the agent. The distribution of target velocity is $U[0, vmax]$. In Cheetah-Dir, the goal directions are limited to forward and backward. In Ant-Dir, the task is specified by the goal direction of the agent's motion. The distribution of goal direction is $U[0, 2\pi]$.The maximal episode step is set to 200.
- **Walker-Param, Hopper-Param** are multi-task MuJoCo benchmarks where tasks differ in transition dynamics. For each task, the physical parameters of body mass, inertia, damping, and friction are randomized. The agent should adapt to the varying environment dynamics to accomplish the task. The maximal episode step is set to 200.

For all environments except for Cheetah-Dir, 40 tasks are randomly sampled with different goal velocities, locomotion, directions, or different physical parameters. we sample 10 tasks for meta-testing and leave the rest for meta-training. For Cheetah-Dir, with only two tasks limited in forward and backward, we keep the training set the same as the testing set, both containing the two tasks.

## F. The Visualization Example of Context



(a) MetaDiffuser (b) CORRO

*Figure 5.* The visualization of the learned task representation in latent space in Hopper-Param. Points are uniformly sampled from test tasks.

## G. Details of Implementations

For pre-training expert policy for each task, we borrow the provided scripts in the official code repositories from CORRO[1]. All baselines are trained with the same offline dataset collected with the pre-trained expert policy.

For FOCAL and CORRO, we run with the open-source implementations[2] for fair comparisons. For Prompt-DT, note that the open-source implementation of Prompt-DT[3] specializes in different hyper-parameter configurations for different environments, such as target rewards and prompt length. In this paper, we conduct three environments, Point-Robot, Walker-Param, and Hopper-Param, without official hyper-parameter configuration since Prompt-DT does not evaluate them. We select the best target rewards and prompt length for each task from the set of hyper-parameters used in the original paper. Moreover, the prompts with medium and random quality are not provided in the official code repositories, so we collect demonstrations during the policy pre-training process as prompts. The slight difference is that we strictly use the accumulated rewards of the trajectories as the metric to distinguish the quality of the prompt. Prompt-DT divides the prompt into three qualities based on the training epochs, which may result in the quality of medium prompts approaching that of expert prompts when training converged fast for some tasks, which could bring a confusing conclusion.

For CVAE-Planner, we substitute the conditioned diffusion to conditioned VAE, which serves the same role as a trajectory generator to guide the planning across tasks. The generated trajectories conclude the state and actions, similar to MetaDiffuser.

[1] https://github.com/PKU-AI-Edge/CORRO
[2] https://github.com/LanqingLi1993/FOCAL-ICLR
[3] https://github.com/mxu34/prompt-dt

The planning horizon stays the same with MetaDiffuser for fairness to investigate the difference between CVAE and conditional diffusion model.

## H. Hyperparameter and Architectural details

In this section, we describe various architectural and hyperparameter details:

- We represent the noise model $\epsilon_\theta$ with a temporal U-Net (Janner et al., 2022a), consisting of a U-Net structure with 6 repeated residual blocks. Each block consisted of two temporal convolutions, each followed by group norm (Wu & He, 2018), and a final Mish nonlinearity (Misra, 2019).
- We choose the historical trajectories length $h$ of $4$ in Point-Robot tasks, $10$ in Ant-Dir, Cheetah-vel, and Cheetah-Dir tasks with reward change, $20$ in Hopper-Param and Walker-Param tasks with dynamics change.
- The context encoder is modeled as multi-layer perceptrons(MLPs) with 3 hidden layers with the final liner layer to produce a context embedding in $64$ dimensions. The generalized reward model $R_\psi$ and dynamics model $P_\omega$ both shared with the structure of the first half of the U-Net used for the diffusion model, with a final linear layer to produce a scalar output.
- We jointly train the context encoder and corresponding reward model and dynamics model using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of $2 \times 10^{-4}$ and batch size of $64$ for 1000 epochs.
- Timestep and context embeddings, both 64-dimensional vectors, are produced by separate 2-layered MLP (with $256$ hidden units and Mish nonlinearity) and are concatenated together before getting added to the activations of the first temporal convolution within each block. We borrow the code for temporal U-Net from the official implementation of Diffuser[4].
- We train noise model $\epsilon_\theta$ using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of $2 \times 10^{-4}$ and batch size of $32$ for $1 \times 10^6$ train steps.
- We choose the proper probability $\beta$ of removing the conditioning information for each task, the detailed choice, and the results can be found in Section 5.6.3.
- We use $K = 100$ diffusion steps.
- We use a planning horizon $H$ of $4$ in Point-Robot task, $16$ in Cheetah-Vel and Cheetah-Dir tasks, $32$ in Ant-Dir, Hopper-Param and Walker-Param tasks.
- We use a guidance scale $\omega \in \{1.2, 1.4, 1.6, 1.8, 2.0\}$ but the exact choice varies by task.

[4] https://github.com/jannerm/diffuser

## I. Details about Pseudocode for Implementation

Here, we provide pseudocode for our implementation.

**Algorithm 2** MetaDiffuser Training Pytorch-like Pseudocode

```
def training(trajectory, context_drop_prob):
  """
  trajectory: expert trajectory from offline data in training set
  context_drop_prob: the probability to drop conditional context
  """

  # Capture task information
  sub_trajs = random_sample(trajectory, len=h)
  # extract sub-segement from whole trajectory to infer context, h denotes historical length
  latent_embeds = []
  for sub_traj in sub_trajs:
    latent_embed = context_encoder(sub_traj)
    latent_embeds.append(latent_embed)
  context_embed = aggreate(latent_embeds) # arbitrary aggreation technique

  mask_dist = Bernoulli(probs=context_drop_prob)
  mask = mask_dist.sample()
  masked_context_embed = context_embed * mask

  # sample sub-traj for diffuison model to estimate
  traj = random_sample(trajectory, len=H) # H denotes planning horizon

  # Diffuse process on traj
  t = randint(0, T) # time step
  eps = normal(mean=0, std=1)
  nosiy_traj = sqrt(alpha_cumprod(t)) * traj + sqrt(1 - alpha_cumprod(t)) * eps

  # Denoise process to reconstruct traj
  pred_traj = denoise_model(noisy_traj, masked_context_embed, t)

  # Set prediction loss
  loss = set_prediction_loss(pred_traj, noisy_traj)
  return loss
```

**Algorithm 3** MetaDiffuser Sampling Pytorch-like Pseudocode

```
def Sampling(warm_start_data, steps, T):
  """
  warm_start_data: the provided trajectories collected in the test task
  # steps: number of sample steps
  # T: number of time steps
  """

  # Capture task information
  sub_trajs = random_sample(warm_start_data, len=h)
  # extract sub-segement from warm_start_data to infer context, h denotes historical length
  latent_embeds = []
  for sub_traj in sub_trajs:
    latent_embed = context_encoder(sub_traj)
    latent_embeds.append(latent_embed)
  context_embed = aggreate(latent_embeds)

  # noisy traj to be denoised, length set as planning horizon H
  traj_t = normal(mean=0, std=1)

  # uniform sample step size
  times = reversed(linespace(-1, T, steps))

  # [(T-1, T-2), (T-2, T-3), ..., (1, 0), (0, -1)]
  time_pairs = list(zip(times[:-1], times[1:])

  for t_now, t_next in zip(time_pairs):
    # Predict traj from pb_t
    pred_traj = denoise_model(traj_t, context_embd, t_now)

    # Estimate pb_t at t_next
    traj_t = denoise_step(traj_t, pred_traj, t_now, t_next)

  return pred_traj
```