Prompting Decision Transformer for Few-Shot Policy Generalization

Mengdi Xu [1] **Yikang Shen** [2] **Shun Zhang** [3] **Yuchen Lu** [2] **Ding Zhao** [1] **Joshua B. Tenenbaum** [4] **Chuang Gan** [3 4 5]

Abstract

Human can leverage prior experience and learn novel tasks from a handful of demonstrations. In contrast to offline meta-reinforcement learning, which aims to achieve quick adaptation through better algorithm design, we investigate the effect of architecture inductive bias on the few-shot learning capability. We propose a Prompt-based Decision Transformer (Prompt-DT), which leverages the sequential modeling ability of the Transformer architecture and the prompt framework to achieve few-shot adaptation in offline RL. We design the trajectory prompt, which contains segments of the few-shot demonstrations, and encodes task-specific information to guide policy generation. Our experiments in five MuJoCo control benchmarks show that Prompt-DT is a strong few-shot learner without any extra finetuning on unseen target tasks. Prompt-DT outperforms its variants and strong meta offline RL baselines by a large margin with a trajectory prompt containing only a few timesteps. Prompt-DT is also robust to prompt length changes and can generalize to out-of-distribution (OOD) environments. Project page: https://mxu34.github.io/PromptDT/.

1. Introduction

Offline Reinforcement Learning (offline RL) (Levine et al., 2020) aims to learn an optimal policy from trajectories collected by a set of behavior policies without access to the environments. This data-driven approach is essential in many settings, where online interactions could be expensive (e.g., robotics or educational agents) and dangerous (e.g., autonomous driving or healthcare). A number of recent works have illustrated the power of such approaches in enabling data-driven learning of policies for game environments (Chen et al., 2021), robotic manipulation behaviors (Ebert et al., 2018; Kalashnikov et al., 2018), and robotic navigation skills (Kahn et al., 2021).

However, we identify one of offline RL's inherent difficulties as the failure to generalize to unseen tasks. While the agent might be able to get good state coverage from the training tasks, due to the distribution shift, it would still struggle to find a good policy in the test tasks. As a result, recent work from Mitchell et al. (2021) considers the offline meta-RL setting that aims to solve the generalization problem in offline RL. It proposed the Meta-Actor Critic with Advantage Weighting (MACAW) algorithm that uses advantage-weighted regression (Peng et al., 2019) as a subroutine RL algorithm, and optimizes the agent's adaptive ability with a meta-learning objective (Finn et al., 2017a).

While meta-learning methods address this issue through the algorithmic learning perspective, we aim to investigate the power of architecture inductive bias in this work. It is known that Transformer (Vaswani et al., 2017) models, when pretrained on large-scale datasets, are able to perform few-shot or zero-shot learning. Furthermore, recent works from Natural Language Processing (NLP) (Liu et al., 2021; Brown et al., 2020) suggest the prompt-based framework as an effective paradigm for adaptation to new tasks, such as translation and question-answering. In the prompt-based framework, the prompt contains valuable information about the task, and is pre-pended as a prefix to the input. As a result, it casts the problem of few-shot or zero-shot generalization to a conditional sequence generation, which has been the strength of these large Transformer models. Recently Chen et al. (2021) shows that beyond the natural language, the Transformer architecture can also have a strong sequence modeling capability for trajectory data, achieving state-of-art results on offline RL. In this work, we aim to address the question: *Can we leverage the prompt-based framework from NLP, and adapt it to the context of offline RL to enable few-shot generalization to unseen tasks?*

It is worth noting that adapting the prompt-based method to RL problems is non-trivial. In NLP, Language Models (LMs) are pretrained on massive amounts of raw text, including almost all information on the internet. Moreover, most NLP tasks can be rewritten into standard blank-filling formats as prompts. These prompts serve as queries to extract

[1]Carnegie Mellon University [2]University of Montreal, Mila [3]MIT-IBM Watson AI Lab [4]Massachusetts Institute of Technology [5]UMass Amherst. Correspondence to: Mengdi Xu <mengdixu@andrew.cmu.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

the right information from the pretrained LMs. In RL, due to inherent differences between different tasks, it is questionable whether a pretrained model has enough knowledge to solve an unforeseen task. We propose to use the few-shot demonstrations as prompts, which we call the trajectory prompt. Instead of unsupervised language model pretraining, we focus on supervisedly training an agent that can imitate these demonstrations to generate a new policy without finetuning. In our prompt-based offline RL framework, the agent is first trained with offline trajectories that are collected from different tasks in the same domain/environment. For each task, the agent learns to predict a target trajectory while conditioning on the trajectory prompt sampled from the same task. During the evaluation, the agent is given a new task and a handful of new trajectories (total step length of which is less or equal to 15) to construct the prompt. Without extra finetuning, the agent should leverage the task information shown in these trajectories and generate policies for new tasks. This framework is powerful and attractive for a number of reasons: it allows the agent to exploit offline trajectories that are collected from different tasks, and the agent can perform few-shot learning, adapting to new scenarios without updating the agent.

We call our method Prompt-based Decision Transformer (Prompt-DT), which leverages the sequential modeling ability of the Transformer architecture and the prompt framework to achieve few-shot adaptation in offline RL. Our contributions are as follows.

1. We propose Prompt-DT, a Transformer-based model that learns to adapt to unseen tasks via short trajectory prompts constructed from a handful of trajectories.
2. Our experiments in five MuJoCo control benchmarks show that Prompt-DT is a strong few-shot learner without any extra finetuning on target tasks, beating strong meta offline RL baselines by a large margin.
3. Our analysis suggests the necessity of our prompt-based framework, as well as the robustness to prompt length and sensitivity to the prompt quality.
4. Prompt-DT can generalize to out-of-distribution tasks, while all the prior methods fail.

2. Related Work

Offline Reinforcement Learning. Offline RL (Levine et al., 2020) learns a policy with the pre-collected dataset, which contains trajectories sampled under a behavior policy. As identified in Levine et al. (2020), the offline RL problem has shown to be more challenging than online RL, as the learning agent needs to estimate the value of a policy using only the offline data. Similar to online RL, we can adopt a model-based or a model-free approach. When using a model-based approach, we can estimate the reward and transition functions with offline data. However, we need to modify the RL algorithm to avoid exploiting errors in the estimated model (Yu et al., 2020b; Kidambi et al., 2021; Yu et al., 2021). Alternatively, when choosing a model-free approach, we can adapt Q-learning algorithms or the policy gradient algorithms to the offline setting, but need to explicitly correct the distributional mismatch between the behavior policy in the offline data and the policy we want to optimize (Kumar et al., 2020; Islam et al., 2019).

Meta-Reinforcement Learning. Meta-reinforcement learning (meta-RL) aims to generalize an agent's knowledge from one task to another. One popular meta-RL algorithm is the Model-Agnostic Meta-Learning (MAML) proposed in Finn et al. (2017a). The objective of MAML is to find a policy parameter such that given a new task within the task distribution, it can achieve a good performance in the new task only after a few updates. MAML involves an inner loop and an outer loop in its optimization process. The inner loop optimizes the policy parameter to adapt to a given task in one step or a few steps. This can be done by following any policy gradient algorithm. The outer loop involves the meta-learning objective, which optimizes the performance of the policy *after adaption* over all possible tasks in the task distribution. MAML has shown successful and effective adaptions in benchmark domains. However, such an optimization algorithm requires backpropagation from the inner loop to the outer loop, which is computationally expensive. More follow-up methods were proposed to mitigate the computational burden (Nichol et al., 2018; Rajeswaran et al., 2019).

Policy Learning as Sequence Modeling. RL algorithms need to handle the challenge of long-term credit assignment, which is typically done by temporal difference (TD) learning (Sutton & Barto, 2018). However, models designed for NLP, like Transformer (Vaswani et al., 2017), can inherently handle the long-term credit assignment problem. Recently, Decision Transformer (Chen et al., 2021) was proposed to model an RL problem as a sequence-prediction problem, using state, action, reward-to-go as tokens in a Transformer model. A concurrent work takes a similar approach that uses Transformer to predict the dynamics of the environment (Janner et al., 2021). These Transformer-based approaches have achieved similar or better performances in benchmark domains compared with classic RL algorithms. Recently, Furuta et al. (2021) demonstrates that Decision Transformer model is doing hindsight information matching.

Few-Shot Learning. Few-Shot Learning (FSL) aims to rapidly generalize to new tasks containing only a few samples with supervised information (Wang et al., 2020). FSL can advance robotics through developing agents that can replicate human actions. Examples include one-shot imitation (Wu & Demiris, 2010), multi-armed bandits (Duan

et al., 2017), visual navigation (Finn et al., 2017a), and continuous control (Yoon et al., 2018). Applications of FSL include image classification (Vinyals et al., 2016), object tracking (Bertinetto et al., 2016), visual question answering (Dong et al., 2018), language modeling (Vinyals et al., 2016), and neural architecture search (Brock et al., 2017). FSL can reduce the data gathering effort for data-intensive applications. Another classic FSL scenario is where examples with supervised information are hard to acquire due to safety or ethical issues (Altae-Tran et al., 2017).

Prompt-based Learning. For NLP, prompt-based learning is based on language models that model the probability of text directly. Unlike traditional supervised learning, which trains a model to take in an input x and predict an output y as $P(y|x)$, a prompt-based method uses a template to modify the original input x into a textual string prompt x' that has some unfilled blanks, and then uses the language model to probabilistically fill answers y into blanks (Liu et al., 2021). In this way, by selecting appropriate prompts, we can manipulate the model to predict desired outputs, sometimes even without any additional task-specific training (Brown et al., 2020; Radford et al., 2019; Gao et al., 2020). The underlying hypothesis is that pretrained language models have learned adequate knowledge from the pretraining corpus and we just need to find the right way to extract the knowledge. However, in the RL setting, we don't have a pretraining corpus that is large and general enough to cover different environments and tasks. Thus, in this work, we propose to use prompts in a different way. Instead of using prompts to extract knowledge from the pretrained model, the RL agent is required to imitate the provided trajectory prompts, such that it can reproduce the policy that generates these trajectories.

3. Preliminaries

3.1. Online and Offline Meta-Reinforcement Learning

A reinforcement learning (RL) problem is a sequential decision-making problem where a learning agent interacts with an environment and optimizes its control policy to obtain the optimal value. Each sequential decision-making task in dynamic environments is in general modeled as a Markov Decision Process (MDP) (Sutton & Barto, 2018) represented by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \mu)$. $\mathcal{S}$ and $\mathcal{A}$ are the state space and the action space. $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition model, where $P(s, a, s')$ is the probability of reaching state s' by taking action a in state s. $R : \mathcal{S} \to \mathbb{R}$ is the reward function. μ is the initial state distribution. At each step, an RL agent interacts with the environment by taking an action a based on the current state s, observing reward r and resulting next state s' from the environment. The objective of a sequential decision-making task is to find a policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that optimizes the expected cumulative rewards, $\mathbb{E}_{s_0 \sim \mu, \pi} \sum_t \gamma^t R(s_t)$.

Generally, RL is performed online, where the agent iteratively takes actions and receives feedback from the environment. However, this may not always be feasible as RL algorithms may require a large number of training data due to their generally low sample efficiency. This makes training in an online environment time-consuming. In some real-world safety-critical environments, deploying the agent online in the training phase may cause catastrophic failures. We consider the *offline RL* setting (Levine et al., 2020), which aims to learn a policy from data that are pre-collected using a (possibly-unknown) behavior policy. In this setting, the agent has access to a dataset $\mathcal{D}$ containing a set of trajectories. A trajectory $\{s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T\}$ is sampled using a behavior policy in the environment. The agent is expected to find the optimal policy using only the dataset $\mathcal{D}$ without interacting with the environment itself.

Both online and offline RL settings are originally proposed to find the optimal policy in one task. The efficiency of RL can be further improved if the designed learning agent is able to adapt to similar tasks with a handful of newly collected data after learning on a few tasks, which is mainly developed under *meta-RL* (Finn et al., 2017a). Recently, *meta-RL* has been extended to offline settings, aiming to adapt to new tasks via pre-collected data quickly. In the *offline meta-RL* setting proposed in Mitchell et al. (2021), 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 π_i. For each task $\mathcal{T}_i$, the agent is provided with a dataset $\mathcal{D}_i$, which contains trajectories sampled using π_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. Decision Transformer

Transformer which has been extensively studied in NLP (Devlin et al., 2018) and computer vision (Carion et al., 2020), has shown to outperform RNN-based architectures. It is recently applied to solve RL problems for its efficiency and scalability when modeling long sequential data. Decision Transformer (Chen et al., 2021) for offline RL treats learning a policy as a sequential modeling problem. It proposes to model trajectories with state s_t, action a_t and reward-to-go $\hat{r}_t$ tuples collected at different time steps t. The reward-to-go is the cumulative rewards from the current time step till the end of the episode. Instead of including the one-step reward r_t, this novel representation helps guide action selection towards optimizing the return. At timestep t, Decision Transformer takes a trajectory sequence τ autoregressively as input which contains the most recent K-step history.

$$\tau = (\hat{r}_{t-K+1}, s_{t-K+1}, a_{t-K+1}, \ldots, \hat{r}_t, s_t, a_t). \quad (1)$$

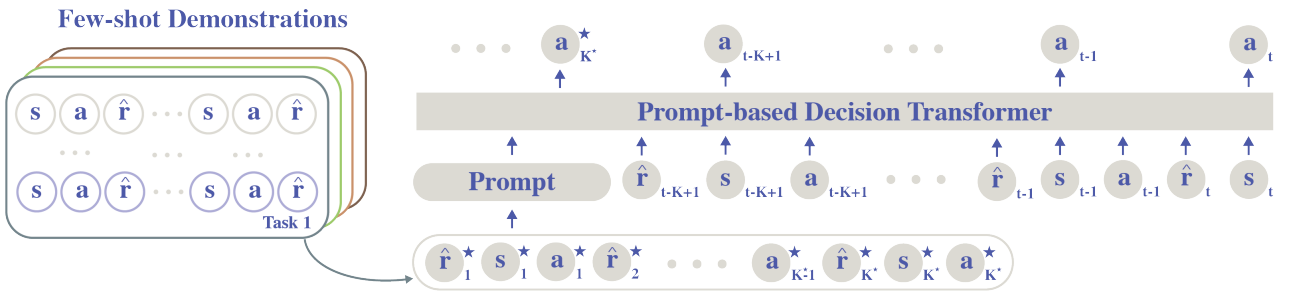

Figure 1. Prompt-DT for few-shot policy generalization. The left shows the few-shot demonstration dataset $\mathcal{P}_i$ for each task $\mathcal{T}_i \in \mathcal{T}^{train} \cup \mathcal{T}^{test}$. The trajectory prompt is defined as a trajectory sequence of length $K^\star$ sampled from various episodes stored in $\mathcal{P}_i$. In both pretraining and few-shot evaluation, Prompt-DT takes both the trajectory prompt augmentation and the most recent K-step history as input, and autoregressively outputs actions corresponding to each state in the input sequence.

When training with offline collected data, $\hat{r}_t = \sum_{i=t}^{T} r_i$. During testing, $\hat{r}_t = G^\star - \sum_{i=0}^{t} r_i$ where $G^\star$ is the targeted total return for an episode. Each trajectory τ corresponds to $3K$ tokens in the standard Transformer model. To encode the sequence timestep information, Decision Transformer concatenate the same timestep embedding to the embeddings of s_t, a_t and $\hat{r}_t$. Each head corresponding to a state token is trained to predict an action by minimizing mean-squared loss when continuous action spaces.

4. Prompt-based Decision Transformer

This section presents Prompt-based Decision Transformer (Prompt-DT), a novel Transformer architecture for few-shot policy generalization as visualized in Figure 1.

4.1. Problem Formulation

We formalize the offline few-shot RL problem as a few-shot policy generalization problem to new tasks after training on a set of tasks with offline data. Each task $\mathcal{T}_i$ in the training set $\mathcal{T}^{train}$ is associated with a dataset $\mathcal{D}_i$, which contains trajectories pre-collected with an unknown behavior policy π_i. In contrast to offline meta-RL, which updates the model weights with task-specific offline data or online interactions, we desire to achieve generalization with no finetuning or gradient updates, which maintains high efficiency and avoids catastrophic forgetting due to parameter shifts.

To achieve few-shot learning in the context of RL, we assume that there exists a dataset $\mathcal{P}_i$ containing few-shot demonstrations for each task $\mathcal{T}_i \in \mathcal{T}^{train} \cup \mathcal{T}^{test}$. For a training task $\mathcal{T}_i \in \mathcal{T}^{train}$, we let $\mathcal{P}_i$ be a subset of the offline data set $\mathcal{D}_i$. $\mathcal{P}_i$ for a test task $\mathcal{T}_i \in \mathcal{T}^{test}$ could be obtained with a human experimenter or a behavior policy. In this work, we hope to design an architecture that can directly extract unique task-specific information stored in the demonstration dataset $\mathcal{P}_i$ and exploit the information to guide policy generation.

4.2. Prompt Representation

Text prompts containing task-specific instructions enable a large Transformer model to generate answers without changing the model parameters in NLP tasks. In the context of RL, text descriptions that could serve as prompts are recently introduced to solve Atari video games and multi-modal household tasks (Shridhar et al., 2020). Such text descriptions usually require predefined language templates and may require large human labor to annotate.

We instead define *trajectory prompt* for RL as a sequence that consists of a few trajectory segments. Each trajectory segment contains multiple state $s^\star$, action $a^\star$ and reward-to-go $\hat{r}^\star$ tuples, $(s^\star, a^\star, \hat{r}^\star)$, following the trajectory representation in Decision Transformer. Each element with superscript $\cdot^\star$ is associated with the trajectory prompt. Since each sequential decision-making task $\mathcal{T}_i$ can be modeled as an MDP $\mathcal{M}_i$, trajectory prompts can store partial to complete information to specify a task by implicitly capturing the transition model and the reward function. Trajectory prompts are relatively easy to obtain compared to text prompts by directly sampling trajectory segments from the few-shot demonstration dataset $\mathcal{P}_i$. Formally, we define a trajectory prompt $\tau_i^\star$ for task $\mathcal{T}_i$ as

$$\tau_i^\star = (\hat{r}_1^\star, s_1^\star, a_1^\star, \hat{r}_2^\star, s_2^\star, a_2^\star, \ldots, \hat{r}_{K^\star}^\star, s_{K^\star}^\star, a_{K^\star}^\star). \quad (2)$$

$K^\star$ is the number of environment steps stored in the prompt. Note that our choice of $K^\star$ is much shorter than the horizon of the task. So a trajectory prompt only contains the information needed to help identify the task but insufficient information for the agent to imitate.

4.3. Prompt-DT Architecture

Our Prompt-DT architecture is built on Decision Transformer (Chen et al., 2021) and solves the offline few-shot RL problem through the lens of a prompt-augmented sequence-modeling problem. The proposed trajectory prompt allows minimal architecture change to the Decision Transformer for

Algorithm 1 Prompt-DT Training

Input: training tasks $\mathcal{T}^{train}$, causal Transformer *Transformer*$_\theta$, training iterations N, offline dataset $\mathcal{D}$, demonstrations $\mathcal{P}$, per-task batch size M, learning rate α
for $n = 1$ **to** N **do**
  for Each task $\mathcal{T}_i \in \mathcal{T}^{train}$ **do**
    for $m = 1$ **to** M **do**
      Sample a trajectory $\tau_{i,m}$ of length K from $\mathcal{D}_i$
      Sample a prompt $\tau^\star_{i,m} =$ *GetPrompt*$(\mathcal{T}_i, \mathcal{P}_i)$
      Get input $\tau^{input}_{i,m} = (\tau^\star_{i,m}, \tau_{i,m})$
    end for
    Get a minibatch $\mathcal{B}^M_i = \{\tau^{input}_{i,m}\}^M_{m=1}$
  end for
  Get a batch $\mathcal{B} = \{\mathcal{B}^M_i\}^{T^{train}}_{i=1}$, $T^{train} = |\mathcal{T}^{train}|$
  $a^{pred} =$ *Transformer*$_\theta(\tau^{input})$, $\forall \tau^{input} \in \mathcal{B}$
  $\mathcal{L}_{MSE} = \frac{1}{|\mathcal{B}|}\sum_{\tau^{input}\in\mathcal{B}}(a - a^{pred})^2$
  $\theta \leftarrow \theta - \alpha\nabla_\theta\mathcal{L}_{MSE}$
  Prompt-DT Few-Shot Evaluation along training
end for

Algorithm 2 Trajectory Prompt Generation (*GetPrompt*)

Input: task $\mathcal{T}$, task-specific demonstrations $\mathcal{P}$, sample episode J, segmentation length H
Sample J episodes from $\mathcal{P}$
Sample segments $\tau^\star_j$ of length H, $\forall j \in [J]$ (Equation 4)
Return: trajectory prompt $\tau^\star = (\tau^\star_1, \ldots, \tau^\star_J)$

Algorithm 3 Prompt-DT Few-Shot Evaluation

Input: test tasks $\mathcal{T}^{test}$, causal Transformer *Transformer*$_\theta$, demonstrations $\mathcal{P}$, target return $G^\star$, episode length T
for Each task $\mathcal{T}_i \in \mathcal{T}^{test}$ **do**
  Initialize history τ with zeros, desired reward $g = G^\star_i$
  Sample a prompt $\tau^\star =$ *GetPrompt*$(\mathcal{T}_i, \mathcal{P}_i)$
  for $t \le T$ **do**
    Get action $a =$ *Transformer*$_\theta((\tau^\star, \tau))$[-1]
    Step env. and get feedback $s, a, r, g \leftarrow g - r$
    Append $[s, a, g]$ to recent history τ
  end for
end for

generalization. For each task $\mathcal{T}_i$, Prompt-DT takes τ^{input} as input, which contains both the $K^\star$ step trajectory prompt obtained from $\mathcal{P}_i$ and the most recent K step history sampled from $\mathcal{D}_i$. Formally $\tau^{input} = (\tau^\star_i, \tau_i)$. Since the data pair at each timestep is a 3-tuple $(s, a, \hat{r})$, the input sequence corresponds to $3(K^\star + K)$ tokens in Transformer. Prompt-DT autoregressively outputs $K^\star + K$ actions at heads corresponding to state tokens in the input sequence.

In the implementation, we utilize stochastic trajectory prompt $\tau^\star_i$ aiming to increase training stability and avoid overfitting as in Algorithm 2. The stochastic trajectory prompt $\tau^\star_i$ for task $\mathcal{T}_i$ consists of J trajectory segments with length H and $K^\star = JH$.

$$\tau^\star_i = (\tau^\star_{i,1}, \ldots, \tau^\star_{i,J}), \tag{3}$$

$$\tau^\star_{i,j} = (\hat{r}^\star_{i,j,1}, s^\star_{i,j,1}, a^\star_{i,j,1}, \ldots, \hat{r}^\star_{i,j,H}, s^\star_{i,j,H}, a^\star_{i,j,H}), \forall j \in [J]. \tag{4}$$

Following the structure of Decision Transformer, we utilize a GPT model with linear layers to obtain token embeddings and add the same positional embedding to tokens corresponding to the same timestep.

4.4. Algorithms

We summarize the training and testing algorithms for Prompt-DT in Algorithm 1 and Algorithm 3.

During training, Prompt-DT minimizes errors between the predicted actions and the actions in the data for both the prompt and recent history. By learning the target actions stored in trajectory prompts, Prompt-DT is motivated to extract the task-specific information stored in the trajectory prompt and combine it with the recent history for future action predictions. In continuous settings, Prompt-DT minimizes the mean-squared loss with gradient descent. At each training step, we sample a batch $\mathcal{B}$ that contains prompt-history pairs for *each* training task. Instead of iteratively conducting gradient updates with data batch sampled from one training task, our batch $\mathcal{B}$ helps stabilize training by aggregating gradient estimates across all tasks in $\mathcal{T}^{train}$.

In testing time, we assume that the offline pretrained Prompt-DT can interact with a simulator for each evaluation task in $\mathcal{T}^{test}$. This online evaluation assumption resembles the actual deployment of trained RL agents. We sample a stochastic trajectory prompt based on the demonstration set $\mathcal{P}_i$ for task $\mathcal{T}_i \in \mathcal{T}^{test}$ similar to the training procedure. Prompt-DT then generates actions taking both the prompt and recent context as input. At the beginning of each episode, we initialize a recent history τ with all zeros for Prompt-DT prediction and update it with streamingly collected data.

5. Experiments

We conduct experiments to test the few-shot generalization capability of the proposed Prompt-DT with metric as the episode accumulated reward. We aim to empirically answer the following questions: 1) Can Prompt-DT achieve few-shot policy generalization? 2) How does the prompt quantity and quality affect the few-shot generalization ability? 3) Does Prompt-DT generalize to out-of-distribution tasks?

5.1. Environments and Datasets

We evaluate in five meta-RL control tasks described as follows (Finn et al., 2017a; Rothfuss et al., 2018; Mitchell et al., 2021; Yu et al., 2020a; Todorov et al., 2012).

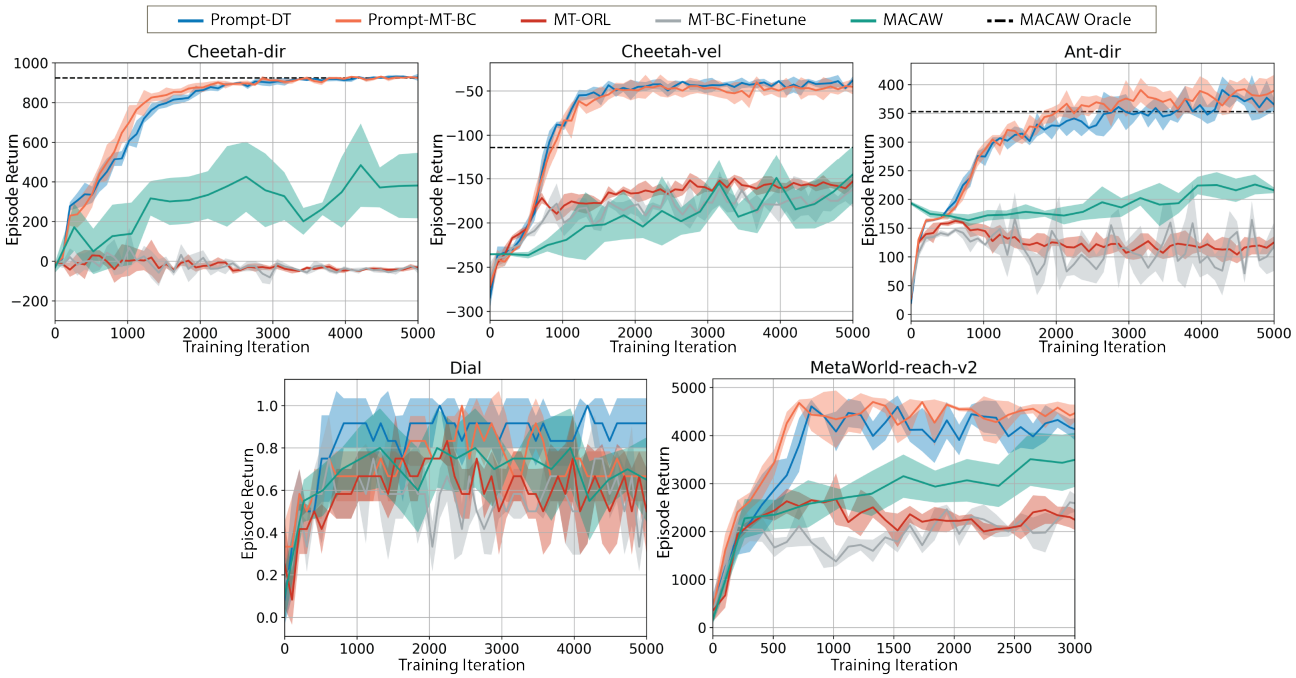

Figure 2. Episodic accumulated returns in never-before-seen tasks of Prompt-DT, Prompt-based Behavior Cloning (Prompt-MT-BC), Multi-task Offline RL (MT-ORL), Multi-task Behavior Cloning (MT-BC-Finetune), and Meta-Actor Critic with Advantage Weighting (MACAW). All methods are trained with the same expert dataset $\mathcal{D}$. Each plot is run with three seeds. Prompt-DT and Prompt-MT-BC have a few-shot dataset $\mathcal{P}$ containing expert demonstrations. Cheetah-dir, Cheetah-vel and Ant-dir have prompts of length $K^\star = 5$. Dial has prompts of length $K^\star = 15$. Meta-World reach-v2 has prompts of length $K^\star = 2$. MT-BC-Finetune and MACAW use the same amount of data which equals the prompt length for finetuning at testing time. The dashed lines show the optimal performance of MACAW reported in Mitchell et al. (2021). Prompt-augmented methods including Prompt-DT and Prompt-MT-BC outperform baselines across environments with a short trajectory prompt.

- **Cheetah-dir.** There are two tasks in Cheetah-dir with goal directions as forward and backward, respectively. The cheetah agent is rewarded with high velocity along the goal direction. The training and testing set are equal, and both contain the two tasks.
- **Cheetah-vel.** There are 40 tasks in Cheetah-vel with different goal velocities. The target velocities are uniformly sampled from the interval [0,3]. The agent is penalized with l_2 errors to the target velocity. We hold out 5 tasks to construct the testing set and train with the remaining 35 tasks.
- **Ant-dir.** There are 50 tasks in Ant-dir with different goal directions uniformly sampled in 2D space. The 8-joints ant is rewarded with high velocity along the goal direction. We sample 5 tasks for testing and leave the rest for training.
- **Dial.** In Dial, the task is to control a 6-DOF Jaco robot to reach a target number located in a number pad. There are 10 numbers in total corresponding to 10 tasks. We train in 6 tasks and test in 4 tasks. Dial is more complex than Meta-World reach-v2 since it directly controls 6 joints.
- **Meta-World reach-v2.** In Meta-World reach-v2, the task is to control a Sawyer robot's end-effector to reach a target position in 3D space. The agent directly controls the XYZ location of the end-effector. Each task has a different goal position. We train in 15 tasks and test in 5 tasks.

Experiments in Cheetah-dir, Cheetah-vel and Ant-dir strictly follow the datasets and settings in Mitchell et al. (2021). The agents in all three tasks are penalized with large control signals. The dataset for Cheetah-dir and Ant-dir contains the full replay buffer for training an RL agent with Soft Actor-Critic (Haarnoja et al., 2018). The dataset for Cheetah-vel contains the replay buffer trained with TD3 (Fujimoto et al., 2018) for its better training stability. In Meta-World reach-v2 (Yu et al., 2020a) and Dial (Shiarlis et al., 2018), we collect expert trajectories with script expert policies provided in both environments.

5.2. Baselines

We compare the few-shot generalization ability of **Prompt-DT** with four baselines, including three variants of Prompt-DT and one state-of-art offline meta-RL algorithm.

- **Multi-task Offline RL (MT-ORL).** We train a Decision Transformer to learn multiple tasks in the training set. We omit the prompt augmentation in Prompt-DT to construct MT-ORL and keep the remaining MT-ORL training process the same as Prompt-DT. In evaluation, the reward-to-go are fed into the Transformer model to provide partial task-specific information. MT-ORL helps ablate the effect of prompt.

- **Prompt-based Behavior Cloning (Prompt-MT-BC).** We omit Prompt-DT's reward-to-go tokens stored in the history input in both training and evaluation. This Prompt-MT-BC baseline only keeps task-specific information in the trajectory prompt. Prompt-MT-BC helps show the effect of reward-to-go tokens.

- **Multi-task Behavior Cloning (MT-BC-Finetune).** We exclude both the prompt augmentation and reward-to-go tokens in the MT-BC-Finetune baseline. To adapt to the target task, we update the Decision Transformer model with finetuning gradient steps with data collected in the target task. MT-BC-Finetune helps show the effect of both prompt and reward-to-go tokens compared with finetuning.

- **Meta-Actor Critic with Advantage Weighting (MACAW).** MACAW is an offline meta-RL algorithm that leverages the strength of both meta-learning and off-policy value-based algorithms. MACAW has high sample efficiency and outperforms multiple finetune adaptation baselines in Cheetah-dir, Cheetah-vel, and Ant-dir (Mitchell et al., 2021).

6. Discussion

6.1. Can Prompt-DT Achieve Few-Shot Policy Generalization?

We compare the few-shot generalization ability of Prompt-DT and the baselines to investigate whether prompts facilitate few-shot generalization, whether prompts encode adequate task-specific information than the rewards, and whether prompt-augmented methods are more data-efficient than finetuning methods. We measure the generalization ability of different methods with the average episode accumulated reward in $\mathcal{T}^{test}$. We show the results in Figure 2, which are the few-shot evaluation performances of different algorithms along the training process.

All the methods in Figure 2 are trained with the same expert dataset in each environment. For Cheetah-dir, Cheetah-vel, and Ant-dir, we select the expert dataset for each task as the last $20,000$ steps in the complete replay buffer. The expert dataset for Dial and Meta-World reach-v2 contains 200 episodes for each task. Prompt-DT and Prompt-MT-BC make use of a few-shot dataset $\mathcal{P}$ containing expert demonstrations sampled from $\mathcal{D}$, and have a trajectory prompt consisting of a segment sampled from a single episode. To have a fair comparison, finetuning methods, including MT-BC-Finetune and MACAW, use offline expert trajectories for estimating finetune gradients and are evaluated in an online manner. We later discuss the effect of demonstration dataset $\mathcal{P}$'s quantity in Section 6.2, and dataset $\mathcal{D}$ and demonstration dataset $\mathcal{P}$'s quality in Section 6.3.

Comparison of MT-ORL and Prompt-MT-BC. With expert datasets $\mathcal{P}$ and $\mathcal{D}$, Prompt-DT achieves high performance with few-shot demonstrations in unseen tasks as shown in Figure 2. Prompt-DT consistently outperforms MT-ORL across environments with a large margin. Prompt-DT and Prompt-MT-BC perform similarly in Cheetah-dir, Cheetah-vel, Ant-dir, and Meta-World reach-v2. This observation shows that the trajectory prompts already embed sufficient information to fully specify the task. However, Prompt-DT outperforms Prompt-MT-BC in Dial, which shows that there exist environments where the prompt itself is insufficient, and the rewards help in policy generalizations.

Prompt-MT-BC performs similarly to MT-ORL in Dial and is better than MT-ORL in the other four environments by a large margin. We can see that the expert trajectory prompt augmentation indeed provides more task-specific information than the reward-to-go tokens stored in the historical context in most situations. In Cheetah-dir, Cheetah-vel and Ant-dir, although we construct different tasks by only modifying reward functions (e.g., changing goal directions and velocities), it is still hard to directly generalize to novel tasks with contexts containing recent reward records according to the poor performances of MT-ORL. In contrast, the trajectory prompt augmentation could provide strong task-specific signals to guide the action generation.

Comparison of MACAW and MT-BC-Finetune. During training, we use the same batch sizes for all the algorithms. Thus, methods at the same number of training iterations (the x-axis in Figure 2) use the same amount of training data. During testing, we provide all methods with the same amount of data collected in the target test task, which means the amount of data used for finetuning equals the prompt length for each environment.

In all the environments except for Dial, Prompt-based methods converge to the optimal performances much faster than MACAW. Note that the optimal performance of MACAW marked as the dashed line in Figure 2 requires a finetune batch of size 256. Even with a much smaller amount of adaptation data (5 vs. 256), prompt-based methods have better asymptotic performance than MACAW in relatively complex environments, including Cheetah-vel and Ant-dir, and similar performance in the simple Cheetah-dir envi-

Table 1. Ablation: The effect of prompt quantity on Prompt-DT's few-shot generalization ability. We vary the number of episodes J and the trajectory segment length H in three environments. We use a recent history containing $K = 20$ timesteps for all experiments. Each number is run with 3 seeds. Prompt-DT only requires an expert prompt with a small number of timesteps to achieve a few-shot generalization.

$K^\star$	J	H	Cheetah-dir	Cheetah-vel	Ant-dir
2	1	2	926.46 ± 2.87	-45.26 ± 2.87	$\mathbf{409.81 \pm 9.69}$
5	1	5	$\mathbf{927.20 \pm 18.02}$	-37.92 ± 4.56	367.12 ± 10.50
10	1	10	925.00 ± 1.41	-38.43 ± 2.14	382.94 ± 25.21
40	2	20	926.87 ± 9.30	$\mathbf{-34.43 \pm 2.33}$	323.83 ± 9.33

ronment. In all the environments, prompt-based methods consistently outperform MT-BC-Finetune. Moreover, with the same amount of finetuning data and finetuning steps, MT-BC-Finetune underperforms MACAW. We conjecture that this is due to the data-hungry nature of Transformer, which has significantly more model parameters than the actor and critic net of MACAW. We provide further ablation studies to show the performance of finetune-based algorithms with various amounts of finetune data and finetune steps in Section C.2 and Section C.3.

6.2. Does the Prompt Quantity Affect the Few-Shot Generalization Ability?

In practice, there may exist a limited amount of high-quality demonstrations for each test task, or the demonstrations may contain trajectories with heterogeneous quality. We provide an ablation study to reveal the effect of the trajectory prompt quantity on the few-shot generalization ability. Table 1 summarizes the ablation results with expert dataset $\mathcal{D}$ and expert demostrations $\mathcal{P}$ in all the three environments. We vary the prompt quantity $K^\star$ by changing the number of trajectory segments J and the segment length H.

In Cheetah-dir, Prompt-DT achieves similar high performances with different prompt lengths, even when the prompt only contains two timesteps. It shows that by training in both tasks in Cheetah-dir, Prompt-DT learns to run fast with history context τ, which is a common skill shared across tasks, and extracts goal direction information stores in the $(s, a, \hat{r})$ pair to generate actions. In Cheetah-vel, the trajectory prompt with parameters $K^\star = 40, J = 2, H = 20$ achieves the highest episode return in unseen task sets. Increasing the prompt length $K^\star$ does not greatly increase the generalization performance. With a short two-timestep trajectory prompt, Prompt-DT could still achieve higher performances than other baselines. In Ant-dir, Prompt-DT achieves the highest performance with a two-timestep prompt. We conjecture that the existence of nonexpert episodes (with negative returns) in Ant-dir's dataset decreases the effectiveness of prompts constructed from multiple trajectories.

Our experiments show that, with trajectory prompt sampled from expert demonstrations, Prompt-DT is not sensitive to the prompt quantity and can successfully extract task-specific information even with prompts containing only a few timesteps. It shows that the sequential information in the trajectory prompt is not crucial to revealing the unique task-specific reward information in the current MuJoCo control settings. However, we conjecture that if tested with a more complex task set, such as Meta-world containing tasks that require sequential tool manipulation, the sequential information stored in the prompt will become more critical.

6.3. Does the Prompt Quality Affect the Few-Shot Generalization Ability?

There are situations where expert demonstrations for target tasks are unavailable, especially when the target test task $\mathcal{T}$ itself is not fully characterized due to model uncertainty (e.g., uncertain dynamic parameters in control tasks). Thus it is vital to investigate how the prompt quality affects the generalization ability. We conduct an ablation study in Cheetah-vel and construct expert, medium, and random datasets $\mathcal{D}'$ corresponding to the last, middle, and first 500 trajectories in the full replay buffer collected along training an RL agent. We also sample expert, medium, and random few-shot demonstrations $\mathcal{P}'$ from $\mathcal{D}'$ accordingly. We summarize the ablation results in Figure 3.

Prompt-DT could adjust its generated actions according to the given trajectory prompt when training with expert data. For example, Prompt-DT achieves low episode return with random trajectory prompt and high performances with expert or medium prompts. Similarly, when training with medium data, Prompt-DT's return heavily decreases when random prompts and slightly increases with expert prompts, validating the effect of prompt quality. However, when training with random data, only feeding Prompt-DT expert or medium trajectory prompts does not help improve the generalization ability. This observation may result from the largely overlapped task state distributions in the random training dataset, reducing the prompt's effects and encouraging Prompt-DT to match the commonly shared random state distribution.

6.4. Can Prompt-DT Generalize to Out-of-distribution Tasks via Few-Shot Demonstrations?

In previous discussions, we follow the experimental setting in MACAW (Mitchell et al., 2021) by training in a large batch of training tasks and evaluating in a few held-out tasks. The held-out tasks have goals (target velocity or target direction) within the goal range calibrated by training tasks. We desire to test whether trajectory prompts enable

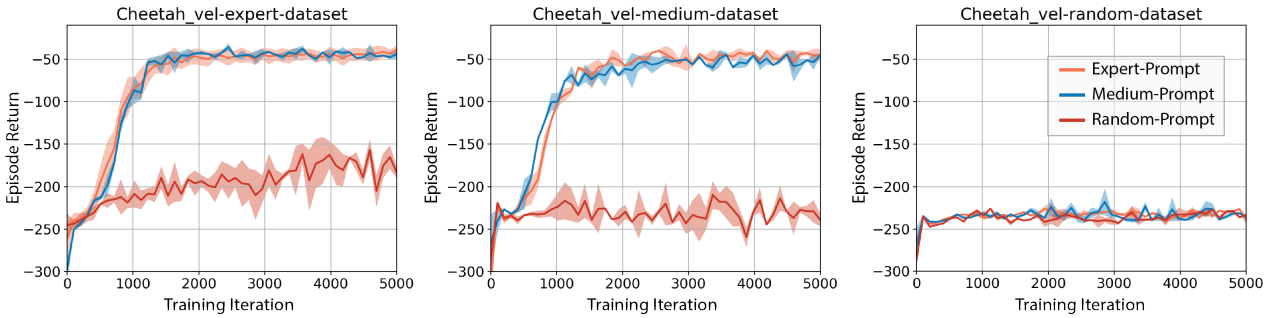

Figure 3. Ablation: The effect of prompt quality to Prompt-DT's few-shot generalization ability. We train Prompt-DT with datasets and demonstrations with the same quality in each plot. The left, middle and right figure corresponds to expert, medium, and random dataset collected in Cheetah-vel. Each plot is run with 2 seeds. We feed Prompt-DT trajectory prompts of different qualities when testing Prompt-DT's few-shot generalization ability. The results show that Prompt-DT tries to generate policies that match the prompt quality, and the quality of training datasets also affects the few-shot generalization ability of Prompt-DT.

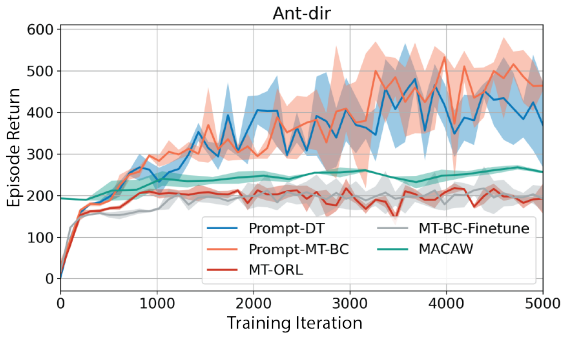

Figure 4. Episodic accumulated returns in novel tasks with goals out of training tasks' goal range in Ant-dir. Each plot is run with 3 seeds. Prompt-MT-BC scores the highest reward and outperforms baselines without trajectory prompts by a large margin.

the extrapolation ability when handling tasks with goals out of the training range. In other words, we hope to test the generalization ability in out-of-distribution tasks.

We sample eight training tasks in Ant-dir and three testing tasks, two of which have indexes smaller than the minimum task index and one larger than the maximum. The task index is proportional to the desired direction angle. We show that Prompt-DT still performs better than baselines with no prompt augmentation in Figure 4. The large variance of Prompt-DT may come from the large variance in episode returns across different testing tasks and the increased sparsity of training tasks.

7. Conclusion

In this work, we proposed Prompt-based Decision Transformer to solve offline few-shot RL problems. We empirically evaluated our algorithm and found it outperform the state-of-the-art offline meta-RL algorithm MACAW (Mitchell et al., 2021) in multiple benchmark domains. We also showed that Prompt-DT is robust to the prompt length changes when trained with an expert dataest, while it is sensitive to the quality of the data provided in the prompt.

To the best of our knowledge, this is the first application of sequence-prediction models in the offline few-shot RL setting, developed based on Decision Transformer (Chen et al., 2021). Our algorithm is simple to implement, which only involves training a prompt-based Transformer, as opposed to training policy and value networks separately using an actor-critic algorithm in MACAW (Mitchell et al., 2021).

We hope this work will inspire more investigation of applications of sequence-prediction models in RL. In future work, we consider designing objective functions that balance the weight of the trajectory prompt and the history context as we currently deem the length of the prompt as a hyperparameter, and use prompt-based Transformer for other RL tasks like meta-imitation learning (Duan et al., 2017; Finn et al., 2017b). We also notice that when using prompts subsampled from expert trajectories, Prompt-DT and Prompt-MT-BC fail to generalize in Meta-World's ML10 benchmark. This motivates future works to design better prompts and prompt-based algorithms to solve complex compositional tasks.

Acknowledgements. This work was supported by MIT-IBM Watson AI Lab and its member company Nexplore, MERL, and DARPA Machine Common Sense program. The information, data, or work presented herein was also funded by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0001210. MDX and ZD gratefully acknowledge support from the National Science Foundation under grant CAREER CNS-2047454. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

Altae-Tran, H., Ramsundar, B., Pappu, A. S., and Pande, V. Low data drug discovery with one-shot learning. *ACS central science*, 3(4):283–293, 2017.

Bertinetto, L., Henriques, J. F., Valmadre, J., Torr, P., and Vedaldi, A. Learning feed-forward one-shot learners. In *Advances in neural information processing systems*, pp. 523–531, 2016.

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*, 2017.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pp. 213–229. Springer, 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.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dong, X., Zhu, L., Zhang, D., Yang, Y., and Wu, F. Fast parameter adaptation for few-shot image captioning and visual question answering. In *Proceedings of the 26th ACM international conference on Multimedia*, pp. 54–62, 2018.

Duan, Y., Andrychowicz, M., Stadie, B. C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. *arXiv preprint arXiv:1703.07326*, 2017.

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.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017a.

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pp. 357–368. PMLR, 2017b.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596. PMLR, 2018.

Furuta, H., Matsuo, Y., and Gu, S. S. Generalized decision transformer for offline hindsight information matching. *arXiv preprint arXiv:2111.10364*, 2021.

Gao, T., Fisch, A., and Chen, D. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

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*, pp. 1861–1870. PMLR, 2018.

Islam, R., Teru, K. K., Sharma, D., and Pineau, J. Off-Policy Policy Gradient Algorithms by Constraining the State Distribution Shift. *arXiv:1911.06970 [cs, stat]*, 2019.

Janner, M., Li, Q., and Levine, S. Reinforcement learning as one big sequence modeling problem. *arXiv preprint arXiv:2106.02039*, 2021.

Kahn, G., Abbeel, P., and Levine, S. Badgr: An autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters*, 6(2):1312–1319, 2021.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pp. 651–673. PMLR, 2018.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. MOReL : Model-Based Offline Reinforcement Learning. In *arXiv:2005.05951 [Cs, Stat]*, 2021.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-Learning for Offline Reinforcement Learning. *Neural Information Processing Systems*, 2020.

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.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021.

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, 2021.

Nichol, A., Achiam, J., and Schulman, J. On First-Order Meta-Learning Algorithms. *arXiv:1803.02999 [cs]*, 2018.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-Learning with Implicit Gradients. In Wallach, H., Larochelle, H., Beygelzimer, A., d\textquotesingle Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 113–124. Curran Associates, Inc., 2019.

Rothfuss, J., Lee, D., Clavera, I., Asfour, T., and Abbeel, P. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

Shiarlis, K., Wulfmeier, M., Salter, S., Whiteson, S., and Posner, I. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning*, pp. 4654–4663. PMLR, 2018.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10740–10749, 2020.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 2018. ISBN 978-0-262-35270-3.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

Wu, Y. and Demiris, Y. Towards one shot learning by imitation for humanoid robots. In *2010 IEEE International Conference on Robotics and Automation*, pp. 2889–2894. IEEE, 2010.

Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., and Ahn, S. Bayesian model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 7343–7353, 2018.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pp. 1094–1100. PMLR, 2020a.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., and Ma, T. MOPO: Model-based Offline Policy Optimization. *Neural Information Processing Systems*, 2020b.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. COMBO: Conservative Offline Model-Based Policy Optimization. *arXiv:2102.08363 [cs]*, 2021.

Appendix: Prompting Decision Transformer for Few-Shot Policy Generalization

We provide the hyperparameters for Prompt-DT and baselines in Section A, more experiment details in Section B and ablation studies in Section C.

A. Hyperparameters

We show the hyperparameter of Prompt-DT and its variants in Table 2 and Table 3, and MACAW in Table 4.

Table 2. Common Hyperparameters of Prompt-DT, Prompt-MT-BC, MT-ORL and MT-BC-Finetune

Hyperparameters	Value
K (length of context τ)	20
training batch size for each task	8
number of evaluation episodes for each task	20
learning rate	1e-4
learning rate decay weight	1e-4
number of layers	3
number of attention heads	1
embedding dimension	128
activation	ReLU

Table 3. Environment-specific Hyperparameters of Prompt-DT and Prompt-MT-BC

Environments	Target RewardS $G^\star$	Prompt Length $K^\star$
Cheetah-dir	1000	5
Cheetah-vel	0	5
Ant-dir	500	5
Dial	10	15
Meta-World reach-v2	1500	2

Table 4. Hyperparameters of MACAW

Hyperparameters	Value
training inner batch size	256
evaluation batch size	prompt length $K^\star$
inner policy learning rate	1e-3
inner value learning rate	1e-3
outer policy learning rate	1e-4
outer value learning rate	1e-5
replay buffer size	20k
inner buffer size	20k
MLP net width	300
MLP net depth	3
activation	tanh
adaptation step	10

B. Experiment Details

We show the task index of the training and testing set for when evaluating the in-distribution generalization capability in Table 5. In other words, the experiments in Section 6.1, Section 6.2 and Section 6.3 follows the training and testing division in Table 5.

Table 5. Training and testing task indexes when testing the generalization ability in in-distribution tasks

Cheetah-dir	
Training set of size 2	[0,1]
Testing set of size 2	[0.1]
Cheetah-vel	
Training set of size 35	[0-1,3-6,8-14,16-22,24-25,27-39]
Testing set of size 5	[2,7,15,23,26]
ant-dir	
Training set of size 45	[0-5,7-16,18-22,24-29,31-40,42-49]
Testing set of size 5	[6,17,23,30,41]
Meta-World reach-v2	
Training set of size 15	[1-5,7,8,10-14,17-19]
Testing set of size 5	[6,9,15,16,20]
Dial	
Training set of size 6	Target pin number: [1,2,3,4,5,8]
Testing set of size 4	Target pin number: [6,7,9,0]

We also show and the task indexes when evaluating the out-of-distribution generalization capability in Table 6 which accounts for the experiments in Section 6.4.

Table 6. Training and testing task indexes when testing the generalization ability in out-of-distribution tasks

ant-dir	
Training set of size 8	[8,13,16,20,22,26,32,37]
Testing set of size 3	[1,4,41]

C. Ablation Study

In this section, we provide addition ablation studies on the effect of prompt length in Section C.1

C.1. The Effect of Prompt Quantity

We show evaluation curves along the training process with various prompt quantity. The training curves for Prompt-DT are shown in Figure 5 and curves for Prompt-MT-BC in Figure 6. Both figures show that in Cheetah-dir, Cheetah-vel and Ant-dir, prompt-based methods are not sensitive to the number of episodes stored in the trajectory prompt or the prompt length.

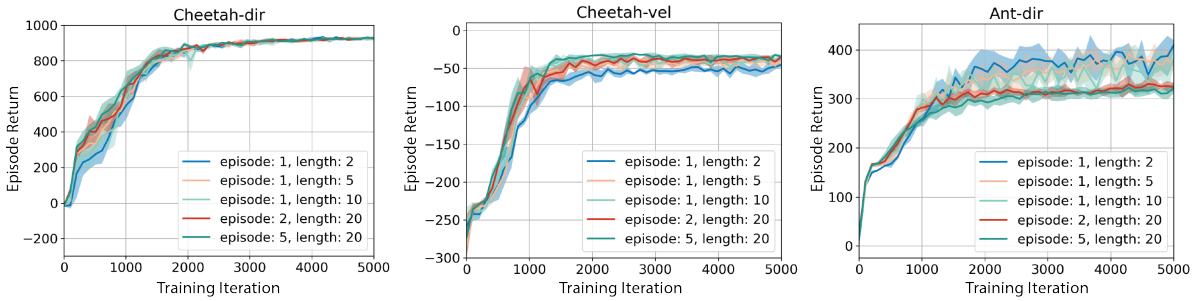

Figure 5. Ablation: The effect of trajectory prompt length to Prompt-DT's performance. Each plot is run with 3 seeds.

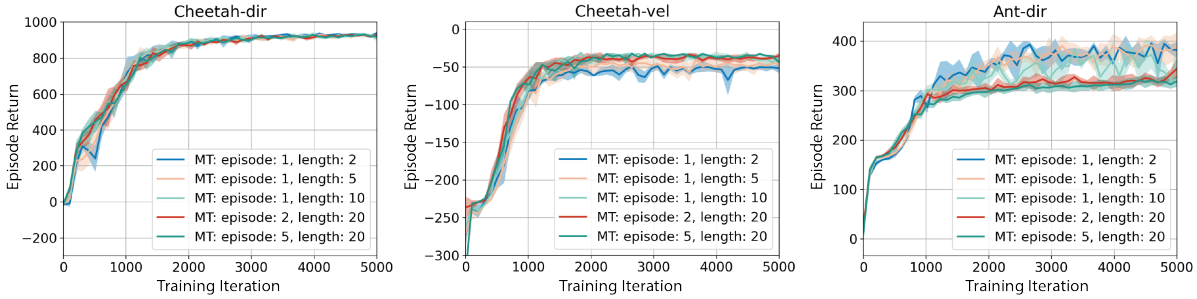

Figure 6. Ablation: The effect of trajectory prompt length to Prompt-MT-BC's performance. Each plot is run with 3 seeds.

C.2. The Effect of Finetune Data's Quantity on MT-BC-Finetune

We show MT-BC-Finetune's performance with various adaptation batch sizes in Figure 7. With limited finetune data, MT-BC-Finetune has difficulties in adapting to every task. MT-BC-Finetune has a large performance variance in Cheetah-dir with an adaptation batch of size 256 and smaller variances in Cheetah-vel and Ant-dir. The large variance in Cheetah-dir may result from the disjoint state distribution in the two tasks with opposite rewards by design.

Note that in Figure 7, we use 10 adaptation steps. We notice that with 100 finetune steps and the adaptation batch size of 1280, MT-BC-Finetune could adapt to test tasks as shown in Table 7.

Environments	Adaptation Batch Size	Finetune Steps	Converged performance
Cheetah-dir	1280	100	930
Cheetah-vel	1280	100	-32
Ant-dir	1280	100	470

Table 7. The performance of MT-BC-Finetune with adequate adaptation data and adaptation steps.

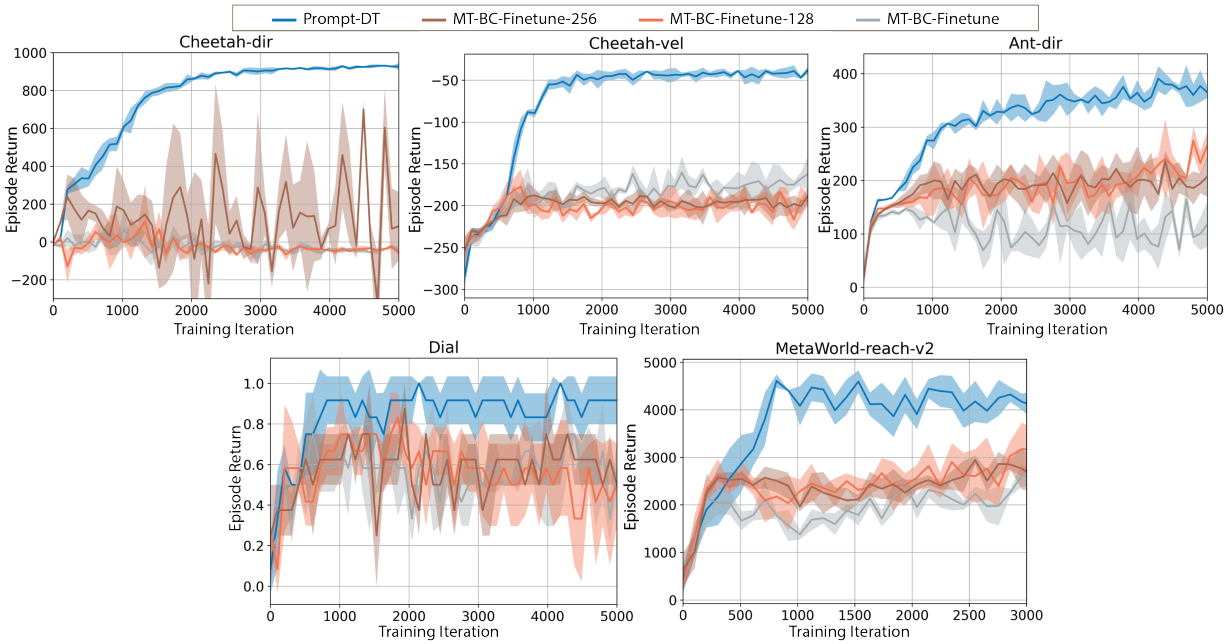

Figure 7. Ablation: The effect of finetune data's quantity on MT-BC-Finetune. MT-BC-Finetune-256 and MT-BC-Finetune-128 have a finetune batch size of 256 and 128 respectively.

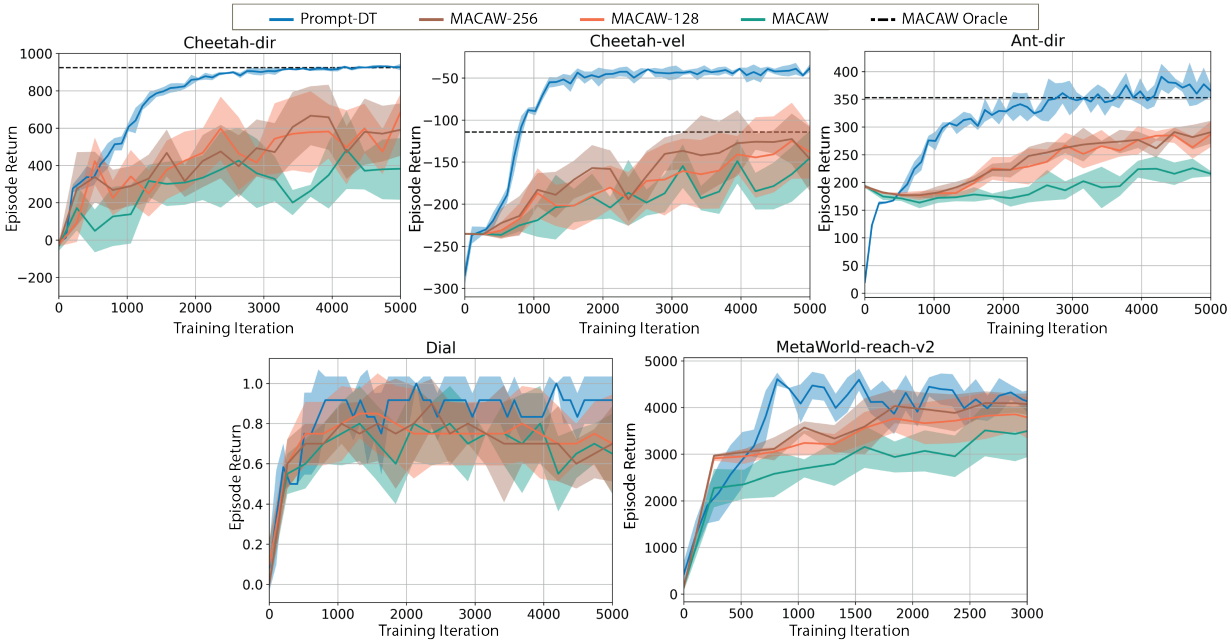

Figure 8. Ablation: The effect of finetune data's quantity on MACAW. MACAW-256 and MACAW-128 have a finetune batch size of 256 and 128 respectively.

C.3. The Effect of Finetune Data's Quantity on MACAW

We show MACAW's performance with various adaptation batch sizes in Figure 8. Each curve has 10 finetuning gradient steps. With an increasing adaptation batch size, MACAW's performance improves consistently across environments. However, MACAW-256 (MACAW with an adaptation batch size of 256) still underperforms Prompt-DT in Cheetah-dir, Cheetah-vel, and Ant-dir. In Meta-World reach-v2, Macaw-256 has a similar asymptotic performance with Prompt-DT but converges slower than Prompt-DT.