Future-conditioned Unsupervised Pretraining for Decision Transformer

Zhihui Xie [1] **Zichuan Lin** [2] **Deheng Ye** [2] **Qiang Fu** [2] **Wei Yang** [2] **Shuai Li** [1]

Abstract

Recent research in offline reinforcement learning (RL) has demonstrated that return-conditioned supervised learning is a powerful paradigm for decision-making problems. While promising, return conditioning is limited to training data labeled with rewards and therefore faces challenges in learning from unsupervised data. In this work, we aim to utilize generalized future conditioning to enable efficient unsupervised pretraining from reward-free and sub-optimal offline data. We propose Pretrained Decision Transformer (PDT), a conceptually simple approach for unsupervised RL pretraining. PDT leverages future trajectory information as a privileged context to predict actions during training. The ability to make decisions based on both present and future factors enhances PDT's capability for generalization. Besides, this feature can be easily incorporated into a return-conditioned framework for online finetuning, by assigning return values to possible futures and sampling future embeddings based on their respective values. Empirically, PDT outperforms or performs on par with its supervised pretraining counterpart, especially when dealing with sub-optimal data. Further analysis reveals that PDT can extract diverse behaviors from offline data and controllably sample high-return behaviors by online finetuning. Code is available at here.

1. Introduction

Large-scale pretraining has achieved phenomenal success in the fields of computer vision (Chen et al., 2020; He et al., 2022) and natural language processing (Devlin et al., 2019; Radford et al., 2018), where fast adaptation to various downstream tasks can be achieved with a unified model trained on a diverse corpus of data. This trend has spurred interest in applying similar paradigms to reinforcement learning (RL), resulting in the prevalence of offline RL (Levine et al., 2020; Kumar et al., 2020; Agarwal et al., 2020). Offline RL aims to learn a task-solving policy exclusively from a static dataset of reward-labeled trajectories. Given the resemblance between offline RL and supervised learning, recent research has further explored the direction of converting offline RL (Chen et al., 2021; Janner et al., 2021) or offline-to-online RL (Zheng et al., 2022) to a sequence modeling problem and using the expressive Transformer architecture (Vaswani et al., 2017) for decision making.

At the heart of these Transformer-based approaches is the idea of conditioning policies on a desired outcome. For example, Decision Transformer (DT, Chen et al. 2020) learns a model to predict actions based on historical context and a target future return. By associating decisions with future returns, DT can perform credit assignment across long time spans, showing strong performance on various offline tasks. While promising, DT presents an incomplete picture as reward-labeled datasets are required to train return-conditioned policies. In practice, task rewards are usually hard to access and poorly scalable to large-scale pretraining. Besides, eschewing rewards during pretraining also allows the model to acquire generic behaviors that can be easily adapted for use in different downstream tasks.

In this work, we aim to equip DT with the ability to learn from reward-free and sub-optimal data. Specifically, we consider the *pretrain-then-finetune* scenario, in which the model is first trained on offline reward-free trajectories and then finetuned on the target task via online interactions. To effectively pretrain a model, it must be able to extract reusable and versatile learning signals in the absence of rewards. During finetuning, the model is required to quickly adapt to task rewards, which presents another challenge as to what learning signals can be aligned with rewards.

To address the above challenges, we propose an unsupervised RL pretraining method called Pretrained Decision Transformer (PDT). Inspired by recent study of future-conditioned supervised learning (Furuta et al., 2022; Venuto et al., 2022; Yang et al., 2023), we enable PDT to condition on the more generalized future trajectory information for

Work done during the first author's internship at Tencent AI Lab. [1]John Hopcroft Center for Computer Science, Shanghai Jiao Tong University, Shanghai, China [2]Tencent AI Lab, Shenzhen, China. Correspondence to: Deheng Ye <dericye@tencent.com>, Shuai Li <shuaili8@sjtu.edu.cn>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

action prediction, which allows the model to learn from unsupervised offline data. PDT jointly learns an embedding space of future trajectory as well as a future prior conditioned only on past information. By conditioning action prediction on the target future embedding, PDT is endowed with the ability to reason over the future. This ability is naturally task-agnostic and can be generalized to different task specifications. To achieve efficient online finetuning in downstream tasks, one can easily retrofit the future-conditioned framework into return-conditioned supervised learning by associating each future embedding to its return. This is realized by training a return prediction network to predict the expected return value for each future embedding, which can be justified from the views of controllable generation (Dathathri et al., 2020) and successor features (Barreto et al., 2017). At evaluation, PDT utilizes the learned future prior together with the return prediction network to controllably sample high-return futures.

We evaluate PDT on a set of Gym MuJoCo tasks from the D4RL benchmark (Fu et al., 2020). Compared with its supervised counterpart (Zheng et al., 2022), PDT exhibits very competitive performance, especially when the offline data is far from expert behaviors. Our analysis further verifies that PDT can: 1) make different decisions when conditioned on various target futures, 2) controllably sample futures according to their predicted returns, and 3) efficiently generalize to out-of-distribution tasks.

2. Related Work

Our work considers **future conditioning** as a powerful approach for **unsupervised RL pretraining**. In this section, we review relevant works in these two research directions.

2.1. Future-conditioned Supervised Learning

Future-conditioned supervised learning has been gaining popularity recently in the field of RL due to its simplicity and competitiveness. The idea is to predict actions conditioned on desired future outcomes, seeking to learn a policy with a future-conditioned supervised loss. Among all kinds of outcomes, future rewards or returns (Schmidhuber, 2019; Kumar et al., 2019) are most commonly used. Chen et al. (2021) propose Decision Transformer (DT), a Transformer-based (Vaswani et al., 2017) approach to learn a return-conditioned policy for offline RL. Zheng et al. (2022) further extend DT to the offline-to-online RL setting by equipping DT with stochastic policies for online exploration. Emmons et al. (2022) show that simple feed-forward MLPs are also capable to learn powerful future-conditioned policies. However, Transformer-based approaches usually show good scaling properties (Lee et al., 2022), in accord with results in the language domain (Kaplan et al., 2020).

Other than target returns, future information such as goals (Andrychowicz et al., 2017), trajectory statistics (Furuta et al., 2022), or learned trajectory embeddings (Yang et al., 2023; Furuta et al., 2022) can also be used to condition the policy. Leveraging future information has been explored as means to combat environment stochasticity (Villaflor et al., 2022; Yang et al., 2023) or to improve value estimations for model-free RL (Venuto et al., 2022). Although our approach potentially enjoys these advantages, in this work we are motivated in a different way and focus on how to tame future conditioning for unsupervised pretraining.

2.2. Unsupervised Pretraining in RL

Our work falls into the category of unsupervised pretraining in RL. A number of works have sought to improve sample efficiency of RL by pretraining the agent with online interactions (Eysenbach et al., 2019; Laskin et al., 2021) or offline trajectories (Ajay et al., 2021; Schwarzer et al., 2021; Lin et al., 2022) prior to finetuning on the target task.

Online unsupervised pretraining aims at learning generic skills by interacting with the environment. During pretraining, the agent is allowed to collect large-scale data from the environment without access to extrinsic rewards. To learn generic skills, existing methods use intrinsic rewards as a principled mechanism to encourage the agent to build its own knowledge. Based on how the intrinsic rewards are designed (Xie et al., 2022), prior works can be categorized into three classes: curiosity-driven exploration (Burda et al., 2019), skill discovery (Eysenbach et al., 2019) and data coverage maximization (Lee et al., 2019).

Although online unsupervised pretraining provides an effective framework to learn prior skills for downstream tasks, it requires abundant online samples which makes it sample inefficient. To address this issue, offline unsupervised pretraining has attracted attention. Existing works on offline unsupervised pretraining mainly fall into two classes. The first class is pretraining representation, aiming to learn good representation from large-scale offline data. To name a few, Yang & Nachum (2021) propose attentive contrastive learning (ACL) which borrows the idea of BERT to pretraining representations that can be used to accelerate the behavior learning on downstream tasks. SGI (Schwarzer et al., 2021) combines self-predictive representation, inverse dynamics prediction and goal-conditioned RL to learn powerful representations for visual observations. The second class is pretraining behavior. Baker et al. (2022) propose video pretraining (VPT), a semi-supervised scheme to utilize large-scale offline data without action information. Specifically, VPT learns an inverse dynamic model on a small-scale supervised data with action information and use the model to provide action for effective behavior cloning and finetuning. Pertsch et al. (2020) propose a deep latent variable

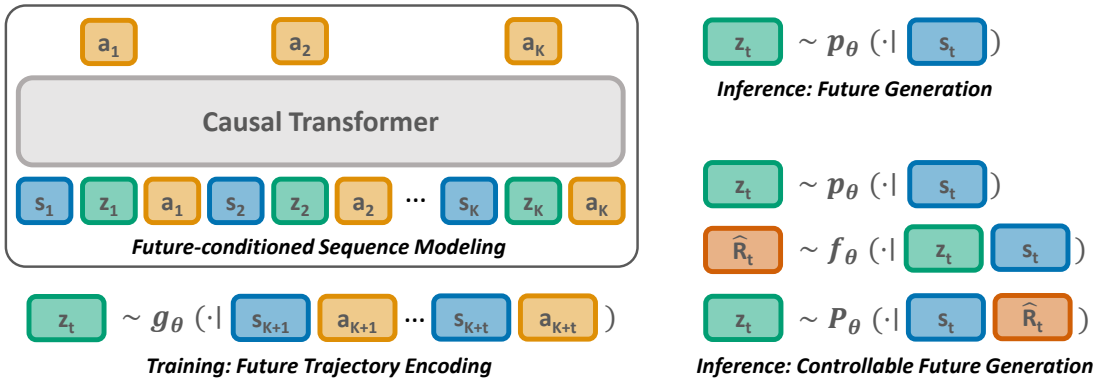


Figure 1. **Overview of the proposed PDT model.** PDT learns to make decisions based on future trajectory information. During training, future embeddings are extracted from the succeeding transitions by future encoder g_θ. At evaluation, PDT samples future embeddings from future prior p_θ conditioned on the current state. When rewards are given in the finetuning phase, PDT learns a return predictor f_θ to steer the sampling procedure towards high-return future embeddings by Bayes' rule $P_\theta(z_t \mid \hat{R}_t, s_t) \propto p_\theta(z_t \mid s_t) f_\theta(\hat{R}_t \mid z_t, s_t)$.

model that jointly learns an embedding space of skills and the skill prior from offline agent experience. Different from these works, we seek to address the issue of reward-centric optimization in existing approaches and enable efficient pretraining on large-scale unsupervised datasets.

3. Preliminaries

3.1. Markov Decision Process

Reinforcement learning (RL) typically models the environment as a Markov decision process (MDP), which can be described as $(\mathcal{S}, \mathcal{A}, P, r, \rho, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s_{t+1} \mid s_t, a_t)$ is the probability distribution over transitions, r is the reward function, ρ is the initial state distribution, and $\gamma \in (0, 1)$ is the discount factor. The objective is to learn a policy that maximizes the cumulative return $\mathbb{E}\left[\sum_{t=1}^{T} r_t\right]$.

Let $\tau = (s_t, a_t)_{t=1}^{T} = (s_1, a_1, s_2, a_2, \cdots, s_T, a_T)$ denote a trajectory composed of a sequence of states and actions, and $\tau_{i:j} = (s_t, a_t)_{t=i}^{j}$ denote a sub-trajectory of τ[1]. Separating rewards from state-action dynamics is deliberate, as rewards are task-specific and usually human-provided.

3.2. RL via Supervised Sequence Modeling

Previous work (Chen et al., 2021; Janner et al., 2021; Lee et al., 2022) has investigated casting offline RL as a sequential modeling task. Specifically, Decision Transformer (DT, Chen et al. 2021) takes the following trajectory representation as input, concatenating τ with rewards:

$$\hat{\tau} = (\hat{R}_1, s_1, a_1, \hat{R}_2, s_2, a_2, \ldots, \hat{R}_T, s_T, a_T),$$

where $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$, usually termed *return-to-go* or simply *return*, is the sum of future rewards from timestep t. After tokenization, $\hat{\tau}$ is fed into a GPT-based Transformer (Radford et al., 2018) which plays the role of an expressive policy function approximator π_θ to predict the next action[2]. For offline RL, the policy is trained to maximize the likelihood of actions in the reward-labeled offline dataset $\hat{\mathcal{D}} = \left\{\hat{\tau}^{(m)}\right\}_{m=1}^{M}$:

$$\mathcal{L}_{\mathrm{DT}} = \mathbb{E}_{\hat{\tau} \sim \hat{\mathcal{D}}}\left[\sum_{t=1}^{T} -\log \pi_\theta(a_t \mid \hat{\tau}_{1:t-1}, s_t, \hat{R}_t)\right].$$

To enable online finetuning, Zheng et al. (2022) further propose Online Decision Transformer (ODT), equipping DT with a stochastic policy and training the model via an additional max-entropy objective:

$$\mathcal{L}_{\mathrm{ODT}} = \mathcal{L}_{\mathrm{DT}} - \alpha \mathbb{E}_{\hat{\tau} \sim \hat{\mathcal{D}}}\left[\sum_{t=1}^{T} H(\pi_\theta(\cdot \mid \hat{\tau}_{1:t-1}, s_t, \hat{R}_t))\right],$$

where α is the temperature parameter (Haarnoja et al., 2018) to control its stochasticity.

While return-conditioned approaches like DT and ODT have gained great attention due to their simplicity and effectiveness when applying to offline RL problems, it requires reward signals to train, which is naturally hard to scale to

[1] $\tau_{t:t-1}$ represents an empty sub-trajectory.

[2] For simplicity, we use θ to represent all the learned parameters.

large-scale pretraining. Besides, using a single scalar value (i.e., the target return) as input could fail to capture sufficient future information. These deficiencies motivate us to design new pretraining algorithms.

3.3. Unsupervised Pretraining

We consider the unsupervised pretraining regime where the goal is to leverage easy-to-collect offline data free of rewards for more data-efficient reinforcement learning.

In the pretraining phase, we assume reward-free offline dataset $\mathcal{D} = \left\{\tau^{(m)}\right\}_{m=1}^{M}$is available. Similar to what we have witnessed in the fields of computer vision (Chen et al., 2020; He et al., 2022) and natural language processing (Devlin et al., 2019; Radford et al., 2018), it is a promising direction to leverage highly sub-optimal and unlabeled training data to conduct RL pretraining.

In the downstream phase, rewards associated with a task are revealed to the agent. We expect the agent to quickly adapt to the task by reusing its prior knowledge learned in the pretraining phase. While offline finetuning is also reasonable, in this work we focus on online finetuning as it requires agents to trade off exploration and exploitation and is usually considered harder (Zheng et al., 2022).

4. Methodology

In this section, we first describe the future conditioning framework that scaffolds PDT. We then present how to pretrain PDT in an unsupervised manner and use the pretrained model for efficient task adaptation. Finally, we show the connection between the ideas of PDT and successor features (Barreto et al., 2017). Figure 1 gives an overview of the proposed PDT.

4.1. Learning to Act by Incorporating the Future

As discussed earlier in Section 3.2, return-conditioned approaches are greatly restricted by rewards. To eliminate the need for reward information, we draw inspiration from future-conditioned RL (Furuta et al., 2022; Emmons et al., 2022) and use it to ground unsupervised pretraining.

Specifically, we condition the policy on future latent variables instead of target returns. Let g_θ be a trajectory-level future encoder and π_θ be a policy network. We condition π_θ on future latent variables $z \sim g_\theta(\cdot \mid \tau)$, where $g_\theta(\cdot \mid \tau)$ outputs a multivariate Gaussian distribution. We expect the model to capture different behaviors seen in the offline data during unsupervised pretraining. These task-agnostic behaviors can be combined to form different policies based on task-specific information (i.e., rewards). Therefore, when the pretrained model is applied to downstream tasks, the agent is able to control which behaviors to sample based on the provided reward information. At test time, the algorithm takes the learned policy π_θ along with a learned prior $p_\theta\left(z \mid s_t\right)$ to take actions:

$$a_t \sim \pi_\theta\left(\cdot \mid \tau_{1:t-1}, s_t, z\right), z \sim p_\theta\left(\cdot \mid s_t\right).$$

The framework of future conditioning brings several benefits. Firstly, it allows us to disentangle rewards from target outcomes, opening up opportunities for large-scale unsupervised pretraining. Secondly, it alleviates the issue of inconsistent behaviors induced by return-conditioned supervised learning, where behaviors significantly deviate from the intended targets (Yang et al., 2023).

4.2. Future-conditioned Pretraining

In this work, similar to that used in DT (Chen et al., 2021; Zheng et al., 2022), we use a GPT-based Transformer architecture (Radford et al., 2018) to parameterize the policy network. Given an input sequence $(s_1, a_1, s_2, a_2, \ldots, s_T, a_T)$, the behavior cloning objective resembles that of ODT:

$$\begin{aligned}\mathcal{L}_{\mathrm{BC}} = & \mathbb{E}_{\substack{\tau \sim \mathcal{D} \\ z \sim g_\theta(\cdot \mid \tau)}}\left[\sum_{t=1}^{T} -\log \pi_\theta(a_t \mid \tau_{1:t-1}, s_t, z)\right] \\ & -\alpha \mathbb{E}_{\substack{\tau \sim \mathcal{D} \\ z \sim g_\theta(\cdot \mid \tau)}}\left[\sum_{t=1}^{T} H(\pi_\theta(\cdot \mid \tau_{1:t-1}, s_t, z)\right],\end{aligned} \tag{1}$$

where $H(\pi_\theta)$ denotes the entropy of action distribution, and α represents a hyperparameter that trades off the contribution of entropy maximization. Note that here we apply the reparameterization trick to allow gradients to backpropagate through the future encoder.

Training the future encoder with the above objective allows us to efficiently leverage future trajectory information to predict actions. However, without explicit regularization, the future encoder can collapse and fail to capture the full distribution of future information. Besides, during execution, we need a prior to guide the agent to sample from the future embedding space. Therefore, inspired by previous work (Ajay et al., 2021; Pertsch et al., 2020), we use the following objective to train the future encoder and a prior:

$$\begin{aligned}\mathcal{L}_{\text{future}} = & \beta \mathbb{E}_{\tau \sim \mathcal{D}}\left[D_{\mathrm{KL}}\left(g_\theta(z \mid \tau) \| \mathcal{N}(\mathbf{0}, I)\right)\right] \\ & + \mathbb{E}_{\tau \sim \mathcal{D}}\left[D_{\mathrm{KL}}\left(\lfloor g_\theta(z \mid \tau) \rfloor \| p_\theta(z \mid s_t)\right)\right],\end{aligned} \tag{2}$$

where D_{KL} denotes the Kullback–Leibler (KL) divergence, $\lfloor \cdot \rfloor$ denotes the stop-gradient operator and β is a hyperparameter. The former serves as a regularization term on the

Algorithm 1 Future-conditioned Pretraining

input policy π_θ, future prior p_θ, future encoder g_θ, reward-free offline dataset $\mathcal{D}$, context length K, batch size B, training iteration I

for $i = 1, \ldots, I$ **do**
  Sample B trajectories from $\mathcal{D}$ according to $p(\tau) = |\tau| / \sum_{\tau' \in \mathcal{D}} |\tau'|$
  for each sampled trajectory τ **do**
    $\tau_{t:t+K-1} \leftarrow$ a length-K sub-trajectory uniformly sampled from τ
    $\tau_{t+K:t+2K-1} \leftarrow$ a length-K future sub-trajectory
    Sample $z \sim g_\theta(\cdot|\tau_{t+K:t+2K-1})$
    Predict actions $a_{t'} \sim \pi_\theta(\cdot \mid \tau_{t:t'-1}, s_{t'}, z), \forall t' = t, \ldots, t+K-1$
    Calculate $\mathcal{L} = \mathcal{L}_{\text{BC}} + \mathcal{L}_{\text{future}}$ (Equation 1,2)
  end for
  Update θ by gradient descent
end for

future encoder to constrain the capacity of the latent z (Higgins et al., 2017). The second term is applied to learn a prior model $p_\theta(z|s_t)$ which helps sample behaviors based on the distribution of offline data. In Appendix A, we discuss the differences between PDT and OPAL (Ajay et al., 2021) in terms of their learning objectives.

Algorithm 1 summarizes the future-conditioned pretraining procedure for PDT. In practice, PDT takes a sub-trajectory as the history context and encodes the succeeding sub-trajectory to obtain the future embedding. Intuitively, we empower the model to take actions based on future trajectory information. This allows PDT to learn robust and general behaviors in an unsupervised manner.

4.3. Return-conditioned Finetuning

Albeit useful to sample future latent variables and generate behaviors imitating the distribution of offline data, $p_\theta(z \mid s_t)$ fails to encode any task-specific information. Therefore, it is required to steer $p_\theta(z \mid s_t)$ to sample futures embeddings that lead to high future return during finetuning.

This leads to controllable generation, an active area of research in computer vision (Nie et al., 2021; Dhariwal & Nichol, 2021) and natural language processing (Dathathri et al., 2020). Lee et al. (2022) consider applying controllable generation to generate expert behaviors for return-conditioned DT. In contrast to controlling a return-conditioned policy by assigning a scalar target return, we need to address a more challenging problem of assigning credits to the future. By Bayes' rule, we have:

$$p(z \mid \hat{R}_t, s_t) \propto p(z \mid s_t) p(\hat{R}_t \mid z, s_t),$$

which suggests that we can sample the desired high-return future embedding by steering the future prior with $p(\hat{R}_t \mid z, s_t)$. Since this distribution is unknown, we use a return prediction network $f_\theta(\cdot \mid z, s)$ to predict $p(\hat{R}_t \mid z, s_t)$. We parameterize $p(\hat{R}_t \mid z, s_t)$ as a Gaussian distribution with learned mean and variance. The return prediction network is trained along with all the other objectives during finetuning:

$$\mathcal{L}_{\text{return}} = \mathbb{E}_{\substack{\hat{\tau} \sim \hat{\mathcal{D}} \\ z \sim g_\theta(\cdot|\hat{\tau})}} \left[-\log f_\theta \left(\hat{R}_t \mid z, s_t \right) \right]. \quad (3)$$

Similar to Equation 1, we apply the reparameterization trick for the future encoder g_θ in Equation 3. This allows gradients to backpropagate to the future encoder, regularizing it to encode task-specific reward information during finetuning.

In practice, we warm-up the return predictor with online exploration transitions at the beginning of the finetuning phase. We consider directly sampling high-return futures, rather than conditioning on a high target return. Specifically, a batch of future embeddings is randomly sampled from the future prior model p_θ and the one with the highest predicted return is selected to condition π_θ during inference, which eliminates the need for a pre-determined target return. Algorithm 2 summaries the finetuning procedure for PDT.

4.4. Connection to Successor Features

As discussed in the previous section, our approach assigns return values to future embeddings for fast task adaptation. This bears resemblance to successor features (SFs, Barreto et al. 2017), a framework for transfer learning in RL. SFs assume that the one-step rewards can be written as:

$$r(s, a) = \phi(s, a)^\top \mathbf{w}, \quad (4)$$

where $\phi(s, a)$ represents the task-agnostic dynamics of the environment whereas $\mathbf{w}$ specifies the task preference. Based on this assumption, the action-value function widely applied in model-free RL algorithms is given by $Q^\pi(s, a) = \psi^\pi(s, a)^\top \mathbf{w}$, where $\psi^\pi(s, a) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a \right]$ is the SF summarizing the dynamics induced by π in the future.

We establish the connection between PDT and SFs by applying the assumption in Equation 4 to returns:

$$\hat{R}_t = \sum_{t'=t}^{T} r_{t'} = \left[\sum_{t'=t}^{T} \phi(s_{t'+1}, a_{t'+1}) \right]^\top \mathbf{w}.$$

Here, feature encoder ϕ needs to be determined, and our future encoder g_θ can be a good candidate to directly encode the summation. Since g_θ is pretrained on reward-free offline trajectories, it naturally encodes information about the environment dynamics. This allows fast task adaptation

dataset	Mean	Min	Max	SAC	ACL	PDT-0	PDT	δ_{PDT}	ODT-0	ODT	δ_{ODT}
hopper-m	44.32	10.33	99.63	24.22 ± 10.55	57.66 ± 6.23	53.74	**95.26** ± 1.77	**41.52**	66.01	87.22 ± 6.85	21.22
hopper-m-r	14.98	0.58	98.73		51.68 ± 48.74	28.56	**84.96** ± 5.49	**56.40**	74.36	75.31 ± 6.22	0.95
walker2d-m	62.09	-0.18	92.04	35.26 ± 23.51	60.21 ± 27.08	73.70	**75.24** ± 4.60	**1.53**	72.80	72.62 ± 5.51	-0.18
walker2d-m-r	14.84	-1.13	89.97		**87.54** ± 7.31	15.64	58.58 ± 14.78	**42.94**	73.27	70.54 ± 2.89	-2.73
halfcheetah-m	40.68	-0.24	45.02	**57.05** ± 3.89	46.59 ± 2.71	42.86	37.93 ± 1.82	**-4.93**	42.69	35.07 ± 10.40	-7.62
halfcheetah-m-r	27.17	-2.89	42.41		50.56 ± 3.74	24.83	29.70 ± 4.97	**4.88**	40.95	35.60 ± 1.68	-5.35
ant-m	80.30	-4.85	107.31	33.30 ± 12.10	28.44 ± 10.78	**93.86**	89.10 ± 6.49	**-4.77**	93.08	73.80 ± 16.77	-19.28
ant-m-r	30.95	-8.87	96.56		9.53 ± 1.80	53.78	48.18 ± 9.59	**-5.60**	**90.37**	60.48 ± 6.23	-29.89
sum	315.33	-7.25	671.67		392.22	386.96	518.93	**123.94**	**553.53**	510.65	-42.87

Table 1. **Performance on Gym MuJoCo tasks.** We run each instance for 200k online transitions, and measure the finetuning performance by the averaged normalized return over 3 random seeds. We also report the zero-shot performance of the pretrained model with suffix "-0". δ_{x} shows the performance gain during online finetuning. The best performance for each task is highlighted in **bold**.

in the downstream finetuning phase, which is achieved by learning task-specific $\mathbf{w}$ with the return prediction network.

5. Experiments

We conduct several experiments to ascertain the effectiveness of PDT, with the aim to gain insights into the following:

- Can PDT extract rich prior knowledge from reward-free offline data to facilitate downstream learning?
- Is unsupervised pretraining comparable with supervised pretraining? Does PDT achieve better generalization than its supervised counterpart?
- How do future conditioning and controllable sampling respectively contribute to PDT's performance?

5.1. Baselines

We compare the performance of PDT to several competitive baselines, including both unsupervised and supervised pretraining methods.

- **Soft Actor-Critic** (**SAC**, Haarnoja et al. 2018) trains a off-policy agent from scratch. We include this baseline to test the benefit of leveraging prior experience.
- **Attentive Contrastive Learning** (**ACL**, Yang & Nachum 2021) considers various representation learning objectives to pretrain state representations on offline data. The pretrained representations are then combined with SAC to solve downstream tasks.
- **Online Decision Transformer** (**ODT**, Zheng et al. 2022) pretrains a stochastic return-conditioned policy using reward-labeled offline data. This baseline serves as a supervised counterpart of PDT, and we include it to test whether supervised pretraining is superior to the proposed unsuperivsed method.

For SAC, we use the open-source codebase[3] and the hyperparameters in Haarnoja et al. (2018). For ACL, we use the official implementation[4], and choose the most competitive reward-free variant and the hyperparameters in Yang & Nachum (2021). We use the official implementation[5] for ODT. Our PDT implementation is based on the ODT codebase. Please see Appendix C.1 for more details.

5.2. Benchmark Datasets

We evaluate our method on the Gym MuJoCo datasets from D4RL (Fu et al., 2020). These datasets consiste of offline trajectories collected by partially trained policies in four simulated locomotion domains: halfcheetah, hopper, walker2d, and ant. Since we care most about how well PDT learns from sub-optimal data, we choose the medium and medium-replay datasets whose trajectories are far from task-solving.

Different from offline RL, unsupervised pretraining assumes that rewards are not available to agents when pretraining on the offline data. After the pretraining phase, agents are allowed to interact with the environment, finetuning their policy from its own behaviors and the corresponding rewards. Following Zheng et al. (2022), we consider a relatively small budget of 200k online interactions. This requires the agent to learn from unlabeled data effectively and adapt to downstream tasks quickly. While Zheng et al. (2022) use offline pretraining data to initialize the replay buffer for finetuning, it is infeasible for the considered baselines to utilize reward-free data. Therefore, we adopt the standard protocol where the agent uses its own rollouts to initialize the replay buffer for finetuning.

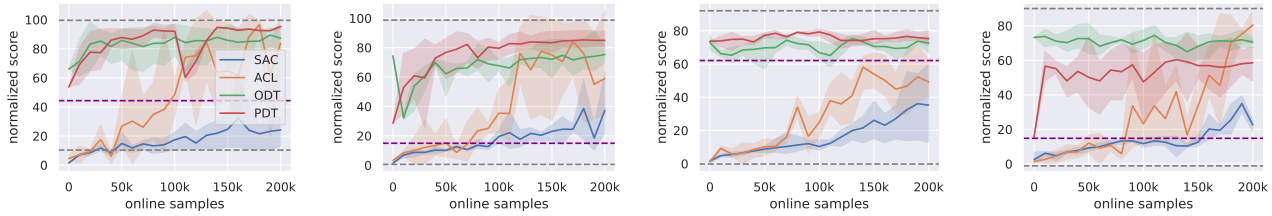
5.3. Gym MuJoCo

We train all instances with 3 random seeds. At evaluation, we run the policy for 10 episodes and compute the average

[3] https://github.com/denisyarats/pytorch_sac

[4] https://github.com/google-research/google-research/tree/master/rl_repr

[5] https://github.com/facebookresearch/online-dt



(a) hopper-medium (b) hopper-medium-replay (c) walker2d-medium (d) walker2d-medium-replay

Figure 2. **Downstream performance on Gym MuJoCO tasks.** Each instance is evaluated over 10 episodes every 10k transitions. Results are averaged over 3 random seeds. The shaded area shows the 95% confidence interval. The purple dashed line represents the averaged episodic return of offline trajectories, whereas the dashed lines in gray represent the min and max values.

return. The reported results are normalized according to Fu et al. (2020). Table 1 summaries the main results and Figure 2 shows the evaluation results during finetuning.

In comparison to learning from scratch using SAC, all the pretraining approaches improve sample efficiency substantially. A closer look at the finetuning phase suggests that PDT not only shows good initial performance by learning to act based on the future, but also quickly adapts to the target task under the supervision of task rewards. Due to the lack of prior knowledge, SAC starts with poor initial performance and heavily relies on online exploration to collect transitions, which results in relatively low sample efficiency. The results of ACL suggest that pretrained representations can provide improvements in terms of sample efficiency in most cases, but we also observe the phenomenon of negative transfer on ant-medium-replay where ACL performs far behind SAC. Besides, since ACL only pretrains state representations, it lacks the ability to reuse behaviors that can potentially benefit the target task.

Perhaps a more interesting observation is that PDT performs on par with its supervised counterpart ODT. If we only consider offline pretraining, ODT outperforms PDT substantially with the help of supervision. But when finetuning, PDT exhibits better sample efficiency, especially when the model is pretrained on the replay datasets (e.g., hopper-medium-replay) composed of highly sub-optimal trajectories. This suggests that PDT can quickly associate futures with return values and latch on high-return behaviors for data-efficient finetuning.

To support that PDT achieves non-trivial performance, we also compare PDT with Rewardless-DT, a simple baseline that builds on ODT but masks out return embeddings during pretraining. Figure 7 shows the finetuning performances of the sequential modeling methods. We observe that Rewardless-DT exhibits relatively good performance on the medium datasets. However, it struggles on the medium-replay datasets. This demonstrates that PDT can extract reusable learning signals from unsupervised pretraining, particularly in cases where the offline data is sub-optimal.

5.4. Analysis

In this section, we seek to gain insight into the key components of PDT, including future conditioning and controllable sampling. Out investigation focuses on how the choice of future embeddings impacts generation, as well as the effect of regularization on future embeddings. We also conduct an evaluation to test whether PDT exhibits better generalization capabilities as compared to its supervised counterpart.

Future conditioning. Ideally, future conditioning enables PDT to behave differently based on different target futures. The ability to reason about the future can be very beneficial, especially at the early stage of an episode when the agent has few history transitions to ground decisions.

Figure 3 shows the action distributions given by a pretrained PDT agent at the initial state of an episode, conditioned on three different future embeddings sampled from the future prior. We can observe that PDT makes clearly different decisions in response to the choice of target future. Figure 8 further demonstrates that, as the episode continues, PDT relies more on its history transitions to make decisions whereas future information plays a less important role.

Performance vs. regularization. β, which controls the regularization strength over future embeddings, is an important hyperparameter for PDT. Intuitively, a large β value discourages the policy to learn diverse behaviors, as minimizing the KL divergence limits how much future information is contained in the latent vectors. In contrast, if β is too small, PDT tends to overly rely on the privileged future information, in the sense that the latent vectors will contain sufficient information so that PDT ignores its own past. Figure 4 shows that, when pretrained on the medium datasets, PDT usually achieves better finetuning performance with larger β. When pretrained on the medium-replay datasets, PDT favors a smaller β value for finetuning.

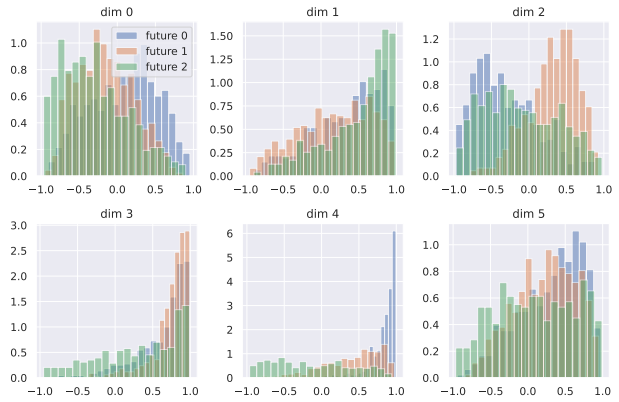


Figure 3. **Future conditioning** enables diverse behavior generation. We plot the histogram of action distribution produced by PDT for each dimension of the action space. The result is taken from the initial state of an episode when running PDT pretrained on the walker2d-medium dataset. See Appendix C.3 for details.

Behavior diversity vs. regularization. To further investigate the connection between the strength of regularization and the diversity of generated behaviors, we conduct another set of experiments. For each PDT policy, we evaluate its behavior diversity by how action distributions vary with different future embeddings to condition the policy. The more dissimilar the action distributions are, the more diverse behaviors PDT can generate by sampling different future embeddings.

As shown in Figure 9, $\beta = 1$ consistently leads to less diverse behaviors. Besides, we observe that phenomenon becomes more pronounced on x-medium-replay datasets, indicating that we can regulate the behavior diversity exhibited by PDT when dealing with diverse data.

Controllable sampling. To steer the pretrained model to perform certain behaviors as specified by the downstream task, PDT uses a learned prior of future together with a return prediction network to sample high-return target futures

Figure 4. **Ablation results of future regularization** for downstream performance. We run each instance for 200k online transitions and report the averaged normalized return over 3 random seeds. The error bar shows the 95% confidence interval.

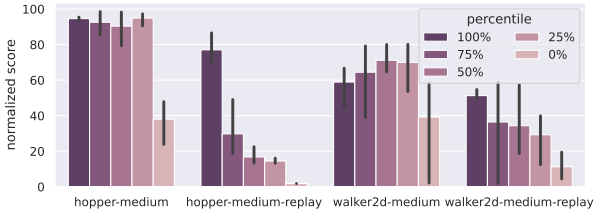


Figure 5. **Ablation of controllable sampling** for downstream performance. We condition PDT on future embeddings with different percentiles of predicted return. We run each instance for 250 online exploration episodes and report the averaged normalized return over 3 random seeds. The error bar shows the 95% confidence interval. See Appendix C.4 for more details.

as contexts to take actions. Hence, it is crucial for the return prediction network to faithfully associate futures with their ground-truth returns. To evaluate this, we condition PDT on the x-percentile of the predicted returns during online finetuning (e.g., $x = 100\%$ recovers the original strategy), and varies the choice of x.

Figure 5 shows the performance of PDT after 250 online exploration episodes. We can observe that PDT achieves varied outcomes when conditioned on different target futures. This again verifies our findings in our former analysis that PDT relies heavily on target future information to make decisions. Besides, the results demonstrate that the return prediction network is effective in filtering high-return futures out of all the possibilities. This property is pivotal, as those reusable target futures can be retrieved in the finetuning phase for fast adaptation. We also find that the performance differences are more prominent on the medium-replay datasets. This could be explained by the fact that medium-replay datasets consist of more diverse behaviors. In such cases, controllable sampling plays an important role to distinguish the desired ones.

More ablation studies. We also compare PDT with its variants to examine how the inclusion of future latent variables affects performance.

To investigate how future conditioning contributes to pretraining performance, we compare PDT with a variant that masks out future embedding input for action prediction during pretraining. Masking out future embeddings disables PDT to take actions based on the future and hence is helpful in ascertaining whether PDT benefits from future information or just memorizes offline behaviors. We observe that pretraining with future embeddings masked leads to significantly degraded performance, as shown in Figure 6.

Next, we compare PDT with a variant that freezes all the parameters except for those of the return predictor during finetuning. Figure 6 shows that, by mining reward-maximizing

(a) hopper-medium (b) hopper-medium-replay

Figure 6. **Ablation results of PDT training.** We report finetuning results averaged over 3 random seeds. The shaded area shows the 95% confidence interval. The purple dashed line represents the averaged episodic return of offline trajectories, whereas the dashed lines in gray represent the min and max values.

actions out of diverse behaviors, the return prediction network plays an essential role so that PDT can continually refine its actions in the finetuning phase. On the flip side, the results also demonstrate that solely relying on controllable sampling is insufficient for online finetuning.

Generalization. In the previous experiments we describe the performance of different pretraining algorithms on the standard D4RL datasets. However, given that the offline trajectories used for pretraining are collected from agents solving the same tasks, it raises a question on how well do the pretrained models generalize to a variety of downstream tasks. To examine this ability of PDT, we modify the reward functions (i.e., task specifications) of halfcheetah and walker2d in the finetuning phase and test whether the pretrained models can quickly adapt to new tasks.

Results are presented in Table 2. After 200k online transitions of downstream task, PDT outperforms ODT by a large margin. This reveals the advantages of unsupervised pretraining over supervised pretraining. Since ODT learns from reward-labeled data, it tends to rely on target returns for decision making and hence struggles to improve when a new task is specified. In contrast, PDT associates decisions to the task-agnostic future information, which can generalize well across different tasks.

6. Conclusion

In this paper, we present Pretrained Decision Transformer (PDT), an unsupervised pretraining algorithm for reinforcement learning (RL). By learning to act based on the future, PDT is able to extract rich prior knowledge from offline data. This ability to leverage future information can be further exploited in the finetuning phase, as PDT associates each future possibility to its corresponding return and samples the one with the highest predicted return to make better decisions. Experimental results demonstrate the effectiveness of PDT in comparison to a variety of competitive baselines.

task	ODT	PDT
halfcheetah-forward-jump	**87.27** ± 14.41	83.80 ± 2.28
halfcheetah-jump	-31.00 ± 49.08	**70.39** ± 16.56
walker2d-forward-jump	29.36 ± 4.55	**45.31** ± 36.81
walker2d-jump	15.81 ± 14.75	**68.70** ± 2.90
sum	101.45	**268.21**

Table 2. **Generalization performance.** We run each instance for 200k online transitions and report the averaged normalized return over 3 random seeds. See Appendix C.6 for more details.

One limitation of PDT is that it requires more training time and computational resources compared to DT and ODT. This could result in practical challenges when the available resources are limited. Besides, the objective in Equation 2 creates a trade-off between diversity of the learned behaviors and behavior consistency. Empirically we find that the optimal value is dataset-specific. To improve future encoding, more advanced techniques such as VQ-VAE (van den Oord et al., 2017) could be applied. For future work, we are interested in exploring how more expressive generative methods (e.g., diffusion models as policies (Janner et al., 2022; Wang et al., 2023)) can benefit PDT.

Acknowledgements

The corresponding author Shuai Li is supported by National Key Research and Development Program of China (2022ZD0114804) and National Natural Science Foundation of China (92270201).

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 104–114. PMLR, 2020. URL http://proceedings.mlr.press/v119/agarwal20c.html.

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. OPAL: offline primitive discovery for accelerating offline reinforcement learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=V69LGwJ0lIN.

Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and

Zaremba, W. Hindsight experience replay. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5048–5058, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html.

Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (VPT): learning to act by watching unlabeled online videos. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9c7008aff45b5d8f0973b23e1a22ada0-Abstract-Conference.html.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., Silver, D., and van Hasselt, H. Successor features for transfer in reinforcement learning. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4055–4065, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/350db081a661525235354dd3e19b8c05-Abstract.html.

Burda, Y., Edwards, H., Storkey, A. J., and Klimov, O. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

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. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 15084–15097, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/7f489f642a0ddb10272b5c31057f0663-Abstract.html.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20j.html.

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., and Liu, R. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1edEyBKDS.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Dhariwal, P. and Nichol, A. Q. Diffusion models beat gans on image synthesis. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 8780–8794, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html.

Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. Rvs: What is essential for offline RL via supervised learning? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=S874XAIpkR-.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020. URL https://arxiv.org/abs/2004.07219.

Furuta, H., Matsuo, Y., and Gu, S. S. Generalized decision transformer for offline hindsight information matching. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=CAjxVodl_v.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. B. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pp. 15979–15988. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01553. URL https://doi.org/10.1109/CVPR52688.2022.01553.

Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M. M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Sy2fzU9gl.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 1273–1286, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/099fe6b0b444c23836c4a5d07346082b-Abstract.html.

Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. Planning with diffusion for flexible behavior synthesis. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Kumar, A., Peng, X. B., and Levine, S. Reward-conditioned policies. *CoRR*, abs/1912.13465, 2019. URL http://arxiv.org/abs/1912.13465.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/0d2b2061826a5df3221116a5085a6052-Abstract.html.

Laskin, M., Yarats, D., Liu, H., Lee, K., Zhan, A., Lu, K., Cang, C., Pinto, L., and Abbeel, P. URLB: unsupervised reinforcement learning benchmark. In Vanschoren, J. and Yeung, S. (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/091d584fced301b442654dd8c23b3fc9-Abstract-round2.html.

Lee, K., Nachum, O., Yang, M., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., and Mordatch, I. Multi-game decision transformers. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b2cac94f82928a85055987d9fd44753f-Abstract-Conference.html.

Lee, L., Eysenbach, B., Parisotto, E., Xing, E. P., Levine, S., and Salakhutdinov, R. Efficient exploration via state marginal matching. *CoRR*, abs/1906.05274, 2019. URL http://arxiv.org/abs/1906.05274.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives

on open problems. *CoRR*, abs/2005.01643, 2020. URL https://arxiv.org/abs/2005.01643.

Lin, Z., Li, J., Shi, J., Ye, D., Fu, Q., and Yang, W. Juewu-mc: Playing minecraft with sample-efficient hierarchical reinforcement learning. In Raedt, L. D. (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pp. 3257–3263. ijcai.org, 2022. doi: 10.24963/ijcai.2022/452. URL https://doi.org/10.24963/ijcai.2022/452.

Nie, W., Vahdat, A., and Anandkumar, A. Controllable and compositional generation with latent-space energy-based models. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 13497–13510, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/701d804549a4a23d3cae801dac6c2c75-Abstract.html.

Pertsch, K., Lee, Y., and Lim, J. J. Accelerating reinforcement learning with learned skill priors. In Kober, J., Ramos, F., and Tomlin, C. J. (eds.), *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pp. 188–204. PMLR, 2020. URL https://proceedings.mlr.press/v155/pertsch21a.html.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018. URL http://openai-assets.s3.amazonaws.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Schmidhuber, J. Reinforcement learning upside down: Don't predict rewards - just map them to actions. *CoRR*, abs/1912.02875, 2019. URL http://arxiv.org/abs/1912.02875.

Schwarzer, M., Rajkumar, N., Noukhovitch, M., Anand, A., Charlin, L., Hjelm, R. D., Bachman, P., and Courville, A. C. Pretraining representations for data-efficient reinforcement learning. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 12686–12699, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/69eba34671b3ef1ef38ee85caae6b2a1-Abstract.html.

Tomczak, J. M. and Welling, M. VAE with a vampprior. In Storkey, A. J. and Pérez-Cruz, F. (eds.), *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1214–1223. PMLR, 2018. URL http://proceedings.mlr.press/v84/tomczak18a.html.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. Neural discrete representation learning. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6306–6315, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Venuto, D., Lau, E., Precup, D., and Nachum, O. Policy gradients incorporating the future. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=EHaUTlm2eHg.

Villaflor, A. R., Huang, Z., Pande, S., Dolan, J. M., and Schneider, J. Addressing optimism bias in sequence modeling for reinforcement learning. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 22270–22283. PMLR, 2022. URL https://proceedings.mlr.press/v162/villaflor22a.html.

Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning.

In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

Xie, Z., Lin, Z., Li, J., Li, S., and Ye, D. Pretraining in deep reinforcement learning: A survey. *CoRR*, abs/2211.03959, 2022. doi: 10.48550/arXiv.2211.03959. URL https://doi.org/10.48550/arXiv.2211.03959.

Yang, M. and Nachum, O. Representation matters: Offline pretraining for sequential decision making. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11784–11794. PMLR, 2021. URL http://proceedings.mlr.press/v139/yang21h.html.

Yang, S., Schuurmans, D., Abbeel, P., and Nachum, O. Dichotomy of control: Separating what you can control from what you cannot. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=DEGjDDV22pI.

You, Y., Li, J., Reddi, S. J., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C. Large batch optimization for deep learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Syx4wnEtvH.

Yu, T., Kumar, A., Chebotar, Y., Hausman, K., Levine, S., and Finn, C. Conservative data sharing for multi-task offline reinforcement learning. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 11501–11516, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/5fd2c06f558321eff612bbbe455f6fbd-Abstract.html.

Zheng, Q., Zhang, A., and Grover, A. Online decision transformer. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27042–27059. PMLR, 2022. URL https://proceedings.mlr.press/v162/zheng22c.html.

A. Comparison to OPAL

The learning objective of future embeddings in Equation 2 bears resemblance to that of OPAL (Ajay et al., 2021). Both PDT and OPAL encode state-action pairs into a latent space and use a latent-conditioned policy as the action decoder. However, we want to clarify their differences as follows:

- **Primitive behaviors vs future information.** Essentially, OPAL does not leverage future information for learning control. For a given trajectory $\tau_{t:t+K-1}$, it extracts a latent vector and maximizes the conditional log-likelihood of actions in $\tau_{t:t+K-1}$ given the state and the latent vector. Since the latent vector already contains information about the actions in $\tau_{t:t+K-1}$, OPAL aims to distill behaviors into temporally extended primitives, rather than to learn a future-conditioned policy. In contrast, PDT learns a future encoder that embeds future trajectory $\tau_{t+K:t+2K-1}$ into latent space so that the latent-conditioned policy can leverage this privileged context to predict actions in $\tau_{t:t+K-1}$.
- **Single latent vs latent sequence.** OPAL encodes the whole trajectory of state-action pairs into a single latent vector, whereas PDT encodes the future trajectory into a sequence of latent vectors with causal masking. Similarly to the return-to-go sequence in DT (Chen et al., 2021; Zheng et al., 2022), latent sequences allow PDT to perceive more future information along the trajectory.
- **Regularization.** OPAL does not regularize the latent distribution, whereas PDT regularizes by minimizing the KL divergence between the latent distribution and the standard Gaussian distribution. There is no rule of thumb as to which is better. Empirically, we found that applying regularization is stable, probably because it can prevent PDT from over-reliance on the privileged future information (see Figure 4). However, we acknowledge that this design could be further improved, in directions like mixture of Gaussians regularization (Tomczak & Welling, 2018).

B. Pseudocode for PDT Finetuning

Algorithm 2 Online Finetuning

input policy π_θ, future prior p_θ, future encoder g_θ, return prediction network f_θ, replay buffer $\mathcal{D}$, context length K, batch size B, number of online rollouts iteration N, training iteration I

Initialize replay buffer $\hat{\mathcal{D}}$ with online rollouts
Warmup return predictor f_θ with $\hat{\mathcal{D}}$
for $n = 1, \dots, N$ **do**
  $\hat{\tau} \leftarrow$ rollout trajectory with π_θ, p_θ, f_θ using the controllable sampling mechanism in Section 4.3
  $\hat{\mathcal{D}} \leftarrow \{\hat{\mathcal{D}}\} \cup \{\hat{\tau}\}$
  for $i = 1, \dots, I$ **do**
    Sample B trajectories from $\hat{\mathcal{D}}$ according to $p(\hat{\tau}) = \hat{R}(\hat{\tau}) / \sum_{\hat{\tau}' \in \hat{\mathcal{D}}} \hat{R}(\hat{\tau}')$
    for each sampled trajectory $\hat{\tau}$ **do**
      $\hat{\tau}_{t:t+K-1} \leftarrow$ a length-K sub-trajectory uniformly sampled from $\hat{\tau}$
      $\hat{\tau}_{t+K:t+2K-1} \leftarrow$ a length-K future sub-trajectory
      Sample $z \sim g_\theta(\cdot | \tau_{t+K:t+2K-1})$
      Predict actions $a_{t'} \sim \pi_\theta(\cdot \mid \tau_{t:t'-1}, s_{t'}, z), \forall t' = t, \dots, t+K-1$
      Predict returns $\hat{R}_{t'} \sim f_\theta(\cdot \mid s_{t'}, z), \forall t' = t, \dots, t+K-1$
      Calculate $\mathcal{L} = \mathcal{L}_{\text{BC}} + \mathcal{L}_{\text{future}} + \mathcal{L}_{\text{return}}$ (Equation 1,2,3)
    end for
    Update θ by gradient descent
  end for
end for

C. Experiment Details

We conduct our experiments on a GPU cluster with 8 Nvidia 3090 graphic cards. Pretraining PDT for 50k gradient steps on a single GPU typically takes 2-3 hours, whereas finetuning for 200k environment steps takes 6-8 hours.

C.1. Hyperparameters

Our PDT implementation is based on the publicly available ODT codebase[6]. We parameterize the future encoder g_θ using the same Transformer network as the policy. The future prior network p_θ uses a pair of fully connected networks with 2 hidden layers and ReLU activation to obtain means and variances of future embedding. The return predictor f_θ also uses a pair of fully connected networks with 2 hidden layers and ReLU activation to obtain a 1-dimensional Gaussian variable.

We use the LAMB optimizer (You et al., 2020) to jointly optimize the policy π_θ, the future encoder g_θ, the future prior p_θ, and the return prediction network f_θ. The temperature parameter α is optimized by the Adam optimizer (Kingma & Ba, 2015). Table 3 summaries the common hyperparameters for PDT. For coefficient β, we search for each task in the range of $\{1, 1e\text{-}2, 1e\text{-}3, 1e\text{-}4\}$ on the main experiments, and fix the choice for other experiments. Table 4 lists the dataset-specific hyperparameters.

For ACL, we choose the most competitive unsupervised variant (i.e., bidir (T) + inputA/R (F) + finetune (T)) as reported in Yang & Nachum (2021). We also take into account the recommended embedding dimension (512) and the pretraining window size (4) that are reported to work best for online RL finetuning. ACL is pretrained for 200k gradient steps following the original paper. For ODT, we use the reported hyperparameters. As suggested in Zheng et al. (2022), a long period of pretraining might hurt the exploration performance. Therefore, we use the model checkpoint for each task once the pretrained model reaches the reported performance of offline pretraining. For finetuning, all ACL, ODT, and PDT collect 10k online transitions to initialize the replay buffer at the beginning.

description	value
number of layers	4
number of attention heads	4
embedding dimension	512
future latent dimension	16
training context length	20
evaluation context length	5
positional embedding	no
future sampling batch size	256
return prediction warmup steps	1500
dropout	0.1
nonlinearity function	ReLU
batch size	256
learning rate	0.0001
weight decay	0.001
gradient norm clip	0.25
learning rate warmup steps	10^4
target entropy	$-\dim(\mathcal{A})$

Table 3. Common hyperparameters that are used to train PDT in all the experiments.

C.2. Details about Rewardless-DT Baseline

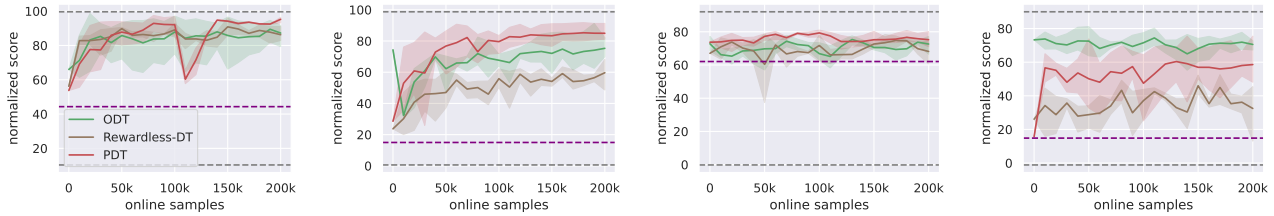
Rewardless-DT has the same network architecture as ODT. The only difference is that, during pretraining, the return embeddings are masked to enable unsupervised learning. We use the same training protocol and hyperparameters as those of ODT to train Rewardless-DT. Figure 7 compares the performance of Rewardless-DT, ODT, and PDT on four Gym MuJoCo tasks.

C.3. Details about Future Conditioning Evaluation

[6] https://github.com/facebookresearch/online-dt

dataset	pretraining updates	β_{pretrain}	β_{finetune}
hopper-medium	20000	1e-3	1e-4
hopper-medium-replay	50000	1	1e-4
walker2d-medium	20000	1e-3	1
walker2d-medium-replay	50000	1	1e-4
halfcheetah-medium	50000	1	1
halfcheetah-medium-replay	50000	1	1
ant-medium	20000	1e-3	1
ant-medium-replay	20000	1e-2	1e-2

Table 4. Hyperparameters we use to train PDT for each dataset.



(a) hopper-medium (b) hopper-medium-replay (c) walker2d-medium (d) walker2d-medium-replay

Figure 7. **Comparison between PDT and Transformer-based baselines** on downstream finetuning performance. Each instance is evaluated over 10 episodes every 10k transitions. Results are averaged over 3 random seeds. The shaded area shows the 95% confidence interval. The purple dashed line represents the averaged episodic return of offline trajectories, whereas the dashed lines in gray represent the min and max values. See Appendix C.2 for more details about Rewardless-DT.

The analysis is done with an PDT agent pretrained on the walker2d-medium dataset, using the same hyperparameters reported in Appendix C.1. The pretrained PDT is evaluated on the walker2d task and conditioned on different future embeddings sampled from the pretrained future prior. Each subfigure in Figure 3 shows the distributions of action on one dimension of the action space. The description for each dimension can be found in the Gym documentation[7].

Figure 8 shows the action distributions of the same PDT model, at time step 500 and 1000 of the episode, respectively.

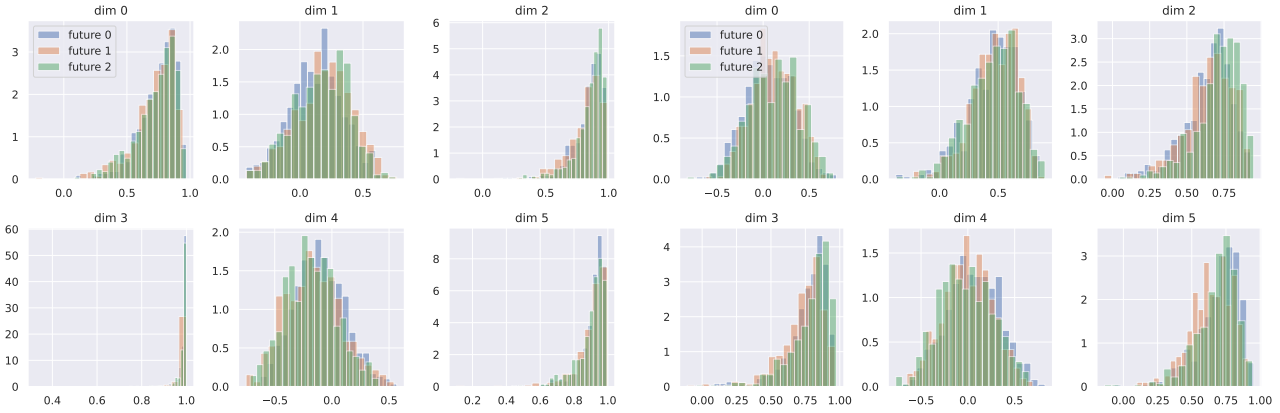
C.4. Details about Controllable Sampling Evaluation

The analysis is done with PDT agents pretrained on four D4RL datasets, using the same hyperparameters reported in Appendix C.1. But for finetuning, we modify the sampling procedure for online exploration and evaluation. Instead of choosing the future embedding with the highest predicted return (100%-percentile), we choose other percentiles.

C.5. Details about Behavior Diversity Evaluation

For each PDT policy, we evaluate its behavior diversity by how action distributions vary with different future embeddings to condition the policy. The more dissimilar the action distributions are, the more diverse behaviors PDT can generate by sampling different future embeddings. Specifically, for each timestep, we sample 10 different future latent sequences from the future prior and obtain the corresponding action distributions $P_1, \ldots P_{10}$. We measure their dissimilarity by average KL divergence: $D_{\text{KL}}\left(P_1, \ldots P_k\right) = \frac{1}{k(k-1)} \sum_{i,j=1}^{k} D_{\text{KL}}\left(P_i \| P_j\right)$. The dissimilarity is then averaged over all the timesteps of 10 episodes, resulting in a scalar value for each policy. We use the distributions before applying the squashing function to calculate KL divergence. Results are reported in Figure 9.

[7] https://www.gymlibrary.dev/environments/mujoco/walker2d/



(a) At timestep 500

(b) At timestep 1000

Figure 8. Comparison of the action distributions produced by PDT when conditioned on different future embeddings. We plot the histogram for each dimension of the action space. The result is taken from the state at timestep 500 (left) and timestep 1000 (right) of an episode when running PDT pretrained on the walker2d-medium dataset. See Appendix C.3.

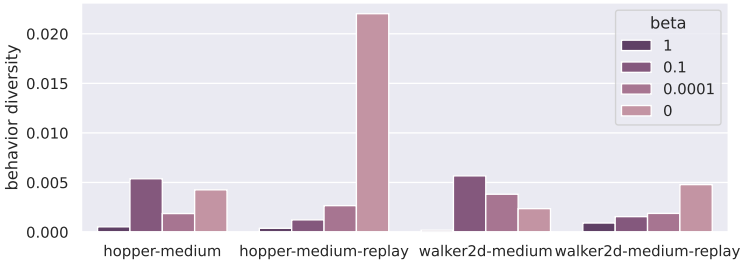


Figure 9. **Future regularization affects behavior diversity.** We compare the behavior diversity of PDT finetuned with different hyperparameter β. See Section C.5 for details.

C.6. Details about Generalization Tasks

We follow Yu et al. (2021) to construct four downstream tasks by changing the reward functions[8]. Specifically, we set the reward functions of task halfcheetah-forward-jump and halfcheetah-jump as $r(s,a) = -0.1 * \|a\|_2^2 + \text{vel}_x + 15 * \text{pos}_z$ and $r(s,a) = -0.1 * \|a\|_2^2 + 15 * \text{pos}_z$. For walker2d, we set the reward functions of task walker2d-forward-jump and walker2d-jump as $r(s,a) = -0.001 * \|a\|_2^2 + \text{vel}_x + 10 * \text{pos}_z$ and $r(s,a) = -0.001 * \|a\|_2^2 + 10 * \text{pos}_z$. We use the same hyperparameters as in Appendix C.1 for downstream online RL. We report scores normalized by computing $100 \times \frac{\text{score-random score}}{\text{best score-random score}}$.

[8]The original reward functions can be found in `gymlibrary.dev`.