DISTRIBUTIONAL DECISION TRANSFORMER FOR OFFLINE HINDSIGHT INFORMATION MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

How to extract as much learning signal from each trajectory data has been a key problem in reinforcement learning (RL), where sample inefficiency has posed serious challenges for practical applications. Recent works have shown that using expressive policy function approximators and conditioning on future trajectory information – such as future states in hindsight experience replay (HER) or returns-to-go in Decision Transformer (DT) – enables efficient learning of context-conditioned policies, where at times online RL can be fully replaced by offline behavioral cloning (BC), e.g. sequence modeling. Inspired by distributional and state-marginal matching literatures in RL, we demonstrate that all these approaches are essentially doing hindsight information matching (HIM) – training policies that can output the rest of trajectory that matches a given future state information statistics. We first present Distributional Decision Transformer (DDT) and its practical instantiation, Categorical DT, and show that this simple modification to DT can enable effective offline state-marginal matching that generalizes well to unseen, even synthetic multi-modal, reward or state-feature distributions. We perform experiments on Gym's MuJoCo continuous control benchmarks and empirically validate performances. Additionally, we present and test another simple modification to DT called Unsupervised DT (UDT), show its connection to distribution matching, inverse RL and representation learning, and empirically demonstrate their effectiveness for offline imitation learning. To the best of our knowledge, DDT and UDT together constitute the first successes for offline state-marginal matching and inverse-RL imitation learning, allowing us to propose first benchmarks for these two important subfields and greatly expand the role of powerful sequence modeling architectures in modern RL.

1 INTRODUCTION

Reinforcement learning (RL) suffers from the problem of sample inefficiency, and a central question is how to extract as much learning signals, or constraint equations (Pong et al., 2018; Tu & Recht, 2019; Dean et al., 2020), from each trajectory data as possible. As dynamics transitions and Bellman equation provide a rich source of supervisory objectives and constraints, many algorithms combined model-free with model-based, and policy-based with value-based in order to achieve maximal sample efficiency, while approximately preserving stable, unbiased policy learning (Heess et al., 2015; Gu et al., 2016; 2017; Buckman et al., 2018; Pong et al., 2018; Tu & Recht, 2019).

Orthogonal to these, in the recent years we have seen a number of algorithms that are derived from different motivations and frameworks, but share the following common trait: they use **future trajectory information** $\tau_{\mathbf{t}:\mathbf{T}}$ to accelerate optimization of a **contextual policy** $\pi(\mathbf{a_t}|\mathbf{s_t}, \mathbf{z})$ with **context** $\mathbf{z}$ with respect to a **parameterized reward function** $\mathbf{r}(\mathbf{s_t}, \mathbf{a_t}, \mathbf{z})$ (see Section 3 for notations). These *hindsight* algorithms have enabled Q-learning with sparse rewards (Andrychowicz et al., 2017), temporally-extended model planning with Q-function (Pong et al., 2018), mastery of 6-DoF object manipulation in cluttered scenes from human play (Lynch et al., 2019), efficient multi-task RL (Eysenbach et al., 2020; Li et al., 2020), offline self-supervised discovery of manipulation primitives from pixels (Chebotar et al., 2021), and offline RL using return-conditioned supervised learning with transformers (Chen et al., 2021a; Janner et al., 2021). We derive a generic problem formulation covering all these variants, and observe that this **hindsight information matching**

(HIM) framework, *with behavioral cloning (BC) as the learning objective*, can learn a conditional policy to generate trajectories that each satisfy any properties, including *distributional* properties.

Inspired by recent works in distribution RL (Bellemare et al., 2017; Dabney et al., 2018), state-marginal matching (Hazan et al., 2019; Lee et al., 2020; Ghasemipour et al., 2020; Gu et al., 2021), and offline RL with transformers (Chen et al., 2021a; Janner et al., 2021), we propose a family of algorithms for distribution matching using **hindsight behavioral cloning with transformers**. We propose the first benchmarks and practical algorithms for offline state-marginal matching and inverse RL imitation learning, greatly expanding the applicability of transformers and other powerful sequential modeling architectures within RL. In summary, our key contributions are:

- We introduced the hindsight information matching (HIM) problem (Section 4), factorized and summarized existing hindsight algorithms based on design attributes, and identified distribution matching as a missing but plausible learning objective under our framework (Table 1).
- We developed Distributional Decision Transformer (DDT) with a practical instantiation, Categorical DT (CDT) (Section 5), validated its empirical performance to match reward or feature distributions (even generalizes to a synthetic bi-modal target distribution at times), and proposed the first benchmark for offline state-marginal matching.
- We developed Unsupervised (UDT) which effectively learns to do moment matching in full state space (Section 5.1), validated its empirical performances to imitate any test trajectory, and proposed the first benchmark for offline inverse RL imitation learning.

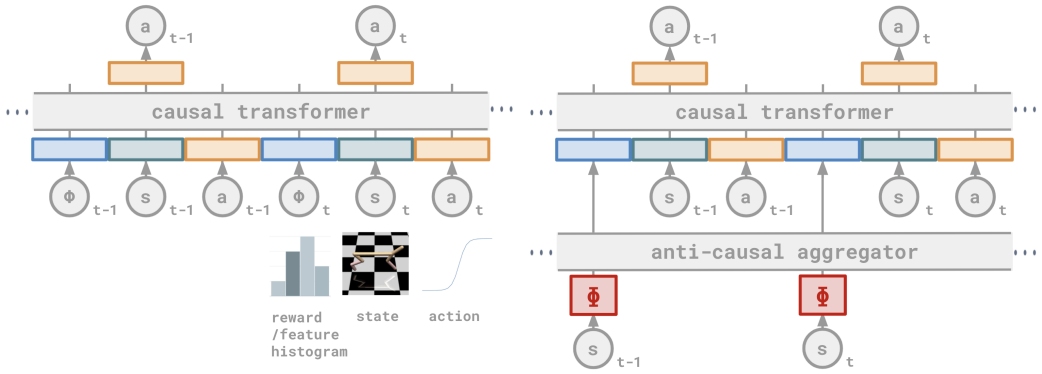


Figure 1: (Left) Categorial DT and (right) Unsupervised DT. Blocks represent parameterized functions to learn, while circles represent inputs and outputs in this sequence modeling formulation. CDT is very close to DT (Chen et al., 2021a) except it takes in (discounted) histograms of future features $\phi_{t:T}$ (e.g. rewards or state-features) instead of discounted sums. UDT, unlike DT and CDT, does not observe an additional feature ϕ and instead learns the feature function $\Phi(s)$. We use parameter-free anti-causal aggregation that is the same procedure for computing discounted sums of returns $\hat{R}$ for DT (Chen et al., 2021a) except that $\Phi(\cdot)$ could be a vector.

2 Related Work

Hindsight Reinforcement Learning and Behavior Cloning Hindsight techniques (Kaelbling, 1993; Andrychowicz et al., 2017; Pong et al., 2018) have revolutionized off-policy optimization with respect to parameterized reward functions. Two key insights were (1) for off-policy algorithms such as Q-learning (Mnih et al., 2015; Gu et al., 2016) and actor-critic methods (Lillicrap et al., 2016; Haarnoja et al., 2018; Fujimoto et al., 2018; Furuta et al., 2021a), the same transition samples can be used to learn with respect to any reward parameters, as long as the reward function is re-computable, i.e. "relabel"-able, like goal reaching rewards, and (2) if policy or Q-functions are smooth with respect to the reward parameter, generalization can speed up learning even with respect to "unexplored" rewards. In goal-based RL where future states can inform "optimal" reward parameters with respect to the transitions' actions, hindsight methods were applied successfully to enable effective training of goal-based Q-function for sparse rewards (Andrychowicz et al., 2017),

derive exact connections between Q-learning and classic model-based RL (Pong et al., 2018), data-efficient off-policy hierarchical RL (Nachum et al., 2018), multi-task RL (Eysenbach et al., 2020; Li et al., 2020), offline RL (Chebotar et al., 2021), and more (Eysenbach et al., 2021; Choi et al., 2021; Ren et al., 2019; Zhao & Tresp, 2018; Ghosh et al., 2021; Nasiriany et al., 2021). Additionally, Lynch et al. (2019) and Gupta et al. (2018) have shown that often BC is sufficient for learning generalizable parameterized policies, due to rich positive examples from future states, and most recently Chen et al. (2021a) and Janner et al. (2021), when combined with powerful transformer architectures (Vaswani et al., 2017a), it produced state-of-the-art offline RL and goal-based RL results. Lastly, while motivated from alternative mathematical principles and not for parameterized objectives, future state information was also explored as ways of reducing variance or improving estimations for generic policy gradient methods (Pinto et al., 2017; Guo et al., 2021; Venuto et al., 2021).

Distributional Reinforcement Learning and State-Marginal Matching Modeling the full distribution of returns instead of the averages led to the development of distributional RL algorithms (Bellemare et al., 2017; Dabney et al., 2018; 2020; Castro et al., 2018; Barth-Maron et al., 2018) such as Categorical Q-learning (Bellemare et al., 2017). While our work shares techniques such as discretization, these works focus on optimizing a non-conditional reward-maximizing RL policy and therefore our problem definition is closer to that of state-marginal matching algorithms (Hazan et al., 2019; Lee et al., 2020; Ghasemipour et al., 2020; Gu et al., 2021), or equivalently inverse RL algorithms (Ziebart et al., 2008; Ho & Ermon, 2016; Finn et al., 2016; Fu et al., 2018; Ghasemipour et al., 2020) whose connections to feature-expectation matching have been long discussed (Abbeel & Ng, 2004). Generalizing hindsight relabeling, we develop an effective offline state-marginal matching algorithm as well as an offline inverse RL imitation algorithm.

Transformer in RL When scaled to the extreme levels of data and computing, transformers (Vaswani et al., 2017b) can train models to master an impressive range of capabilities in natural language processing and computer vision (Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020; Radford et al., 2021; Ramesh et al., 2021; Chen et al., 2021b; Bommasani et al., 2021; Dosovitskiy et al., 2020). Comparing to their popularity in other areas, the adoption of transformers or architectural innovations in RL have been slow, partially due the difficulty of using transformers over temporal scales for online RL (Parisotto et al., 2020). Recent successes have focused on processing variable-length per-timestep information such as morphology (Kurin et al., 2020), sensory information (Tang & Ha, 2021), or leveraged offline learning (Chen et al., 2021a; Janner et al., 2021). Our work enables sequence modeling to solve state-marginal matching or inverse RL imitation learning problems, greatly expanding the impacts of transformers and other powerful sequence models in RL.

3 Preliminaries

We consider a Markov Decision Process (MDP) defined by the tuple of action space $\mathcal{A}$, state space $\mathcal{S}$, transition probability function $p(s'|s,a)$, initial state distribution $p(s_0)$, reward function $r(s,a)$, and discount factor $\gamma \in (0,1]$. In deep RL, a policy that maps the state space to the action space is parameterized by the function approximators, $\pi_\theta(a|s)$[1]. The RL objective is given by:

$$L_{\text{RL}}(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim\rho^\pi(s),a\sim\pi(\cdot|s)}\left[r(s,a)\right] \tag{1}$$

where $p_t^\pi(s) = \iint_{s_{0:t},a_{0:t-1}} \prod_t p(s_t|s_{t-1},a_{t-1})\pi(a_t|s_t)$ and $\rho^\pi(s) = (1-\gamma)\sum_t \gamma^t p_t^\pi(s_t = s)$ are short-hands for time-aligned and time-aggregated state marginal distributions following policy π.

3.1 State Marginal Matching

State marginal matching (SMM) (Lee et al., 2020; Hazan et al., 2019; Ghasemipour et al., 2020) has been recently studied as an alternative problem specification in RL, where instead of stationary-reward maximization, the objective is to find a policy minimizing the divergence D between its state marginal

[1]For simplicity of notations, we write Markovian policies; however, such notations can easily apply to non-Markov policies such as Decision Transformer (Chen et al., 2021a) by converting to an augmented MDP consisting of past N states, where N is the context window of DT.

Method	Algo. Type	Training	$\mathbf{I}_{\mathbf{z}}^{\mathbf{\Phi}}(\tau) \propto$	$\mathbf{z}$	Architectures
Lynch et al. (2019)	BC	Offline	$\sum_t \gamma^t \delta(\phi = \phi_t)$	$z_{\text{goal_mean}}$	Stochastic RNN
Ghosh et al. (2021)	BC	Online	$\sum_t \gamma^t \delta(\phi = \phi_t)$	$z_{\text{goal_mean}}$	MLP
Li et al. (2020)	RL	Online	$\exp\left(\sum_t \gamma^t \phi(s_t, a_t, z_{\text{task}})\right)$	$\{z_{\text{task}}, z_{\text{opt}}\}$	MLP
Eysenbach et al. (2020)	BC/RL	On/Offline	$\exp\left(\sum_t \gamma^t \phi(s_t, a_t, z_{\text{task}})\right)$	$\{z_{\text{task}}, z_{\text{opt}}\}$	MLP
Srivastava et al. (2019)	BC	Online	$\sum_t \gamma^t \phi_t$	$\{z_{\text{return_mean}}, T\}$	Fast Weights
Chen et al. (2021a)	BC	Offline	$\sum_t \gamma^t \phi_t$	$z_{\text{return_mean}}$	Transformer
Janner et al. (2021)	BC	Offline	$\sum_t \gamma^t \phi_t$ or $\sum_t \gamma^t \delta(\phi = \phi_t)$	$z_{\text{return_mean}}$ or $z_{\text{goal_mean}}$	Transformer
Kumar et al. (2019)	BC	Online	$\sum_t \gamma^t \phi_t$	$z_{\text{return_mean}}$	MLP
Andrychowicz et al. (2017)	RL	Online	$\sum_t \gamma^t \delta(\phi = \phi_t)$	$z_{\text{goal_mean}}$	MLP
Pong et al. (2018)	RL	Online	$\delta(\phi = \phi_t)$	$\{z_{\text{goal_mean}}, t\}$	MLP
Chebotar et al. (2021)	RL	Offline	$\sum_t \gamma^t \delta(\phi = \phi_t)$	$z_{\text{goal_mean}}$	CNN
Distributional DT (ours)	BC	Offline	$\sum_t \gamma^t \delta(\phi = \phi_t)$	Any	Transformer
Categorical DT (ours)	BC	Offline	$\sum_t \gamma^t \delta(\phi = \phi_t)$	$z_{\Phi_{\text{hist}}}$	Transformer

Table 1: A coarse summary of hindsight information matching (HIM) algorithms. The notation follows Section 4. All entries in the table essentially belong to three generic problem types: goal-based (e.g. Andrychowicz et al. (2017)), multi-task (e.g. Li et al. (2020)), and reward-conditioned (e.g. Chen et al. (2021a)) (see Appendix C for full descriptions). Pong et al. (2018) and Srivastava et al. (2019) include horizons in z. While γ is noted throughout, distribution matching methods often use $\gamma = 1$ for the objective, as in our experiments. "RL" methods mostly use "Online", i.e. interactive, training, except Chebotar et al. (2021). "BC" methods could be adopted easily "Online" learning through recursive data collections (Ghosh et al., 2021; Kumar et al., 2019).

distribution $\rho^\pi(s)$ to a given target distribution $p^*(s)$[2]:

$$L_{\text{SMM}}(\pi) = -D(\rho^\pi(s), p^*(s)) \tag{2}$$

where D is a divergence measure such as Kullback-Leibler (KL) divergence (Lee et al., 2020; Fu et al., 2018) or, more generally, some f-divergences (Ghasemipour et al., 2020). For the target distribution $p^*(s)$, Lee et al. (2020) set a uniform distribution to enhance the exploration over the entire state space; Ghasemipour et al. (2020) and Gu et al. (2021) set through scripted *distribution sketches* to generate desired behaviors; and adversarial inverse RL methods (Ho & Ermon, 2016; Fu et al., 2018; Ghasemipour et al., 2020; Kostrikov et al., 2019) set as the expert data for imitation learning. For example, as shown in Ghasemipour et al. (2020), choosing reverse KL divergence for D recovers the optimization objective for SMM (Lee et al., 2020) and AIRL (Fu et al., 2018):

$$L_{\text{SMM, rKL}}(\pi) = -\mathbb{E}_{s\sim\rho^\pi(s)}\left[\log \rho^\pi(s) - \log p^*(s)\right] \tag{3}$$

Notably, unlike the RL objective in Eq.1, SMM objectives like Eq.3 no longer depend on task rewards and are only functions of *state information*.

3.2 Parameterized RL Objectives

Lastly, we discuss the basis for methods like HER and TDM (Andrychowicz et al., 2017; Pong et al., 2018), LfP (Lynch et al., 2019), and return-conditioned or upside-down RL (Srivastava et al., 2019; Kumar et al., 2019; Chen et al., 2021a; Janner et al., 2021): parameterized RL objectives. Given parameterized reward functions[3] or target divergences with parameter $z \in \mathcal{Z}$, a conditional policy $\pi(a|s, z)$ is learned with respect to multiple values of z simultaneously weighted by $p(z)$. As examples, the RL and SMM objectives in Eq.1 and Eq.2 become:

$$L_{\text{RL}}(\pi) = \mathbb{E}_z\left[L_{\text{RL}}(\pi, z)\right] = \frac{1}{1-\gamma}\mathbb{E}_{z\sim p(z), s\sim\rho_z^\pi(s), a\sim\pi(\cdot|s,z)}\left[r_z(s,a)\right] \tag{4}$$

$$L_{\text{SMM}}(\pi) = \mathbb{E}_z\left[L_{\text{SMM}}(\pi, z)\right] = -\mathbb{E}_{z\sim p(z)}\left[D(\rho_z^\pi(s), p^*(s))\right] \tag{5}$$

where the state marginal ρ_z^π is from rolling out a conditioned policy $\pi(\cdot|\cdot, z)$.

4 Hindsight Information Matching

We show how HER and TDM (Andrychowicz et al., 2017; Pong et al., 2018), LfP (Lynch et al., 2019), hindsight multi-task RL (Li et al., 2020; Eysenbach et al., 2020), and return-conditioned

[2]As discussed in Fu et al. (2018), and Ghasemipour et al. (2020), it's also straight-forward to define state-action-marginal matching with respect to $\rho^\pi(s,a) = \rho^\pi(s)\pi(a|s)$ and the exact same algorithms apply.

[3]These can be considered as a special case of contextual MDPs (Jiang et al., 2017).

or upside-down RL (Srivastava et al., 2019; Kumar et al., 2019; Chen et al., 2021a; Janner et al., 2021) all belong to *hindsight* algorithms with a shared idea of **using future state information to automatically mine for positive, or "optimal", examples with respect to certain contextual parameter values**, where these examples can accelerate RL or be used for behavior cloning (BC), i.e. supervised learning. We start by defining additional notations.

Given a partial trajectory from state s_t as $\tau_t = \{s_t, a_t, s_{t+1}, a_{t+1}, \dots\}$, we define its *information statistics* as $I(\tau_t)$. $I(\tau_t)$ **could be any function of a trajectory** that captures some statistical properties, such as sufficient statistics of a distribution like mean and variance, or a distribution itself. Since computing distributional statistics on raw state representation are often not practical (Gu et al., 2021), we further define a feature function $\Phi(\cdot,\cdot) : S \times A \to F$[4], where the trajectory is then noted as $\tau_t^\Phi = \{\phi_t, \phi_{t+1}, \dots, \phi_T\}, \phi_t = \Phi(s_t, a_t) \in F$ and the information statistics as $I^\Phi(\tau_t)$. Inspired by moment matching and sufficient statistics in exponential families derivation (Wainwright & Jordan, 2008), we define *information matching (IM)* problems as the following, where the goal is to learn a conditional maximum state-entropy policy that whose future state rollouts satisfy some desired information statistics I_z^* parameterized by z:

$$\min_\pi \mathbb{E}_{z\sim p(z), \tau\sim\rho_z^\pi(\tau)} \left[D(I_z^\Phi(\tau), I_z^*) \right] \tag{6}$$

Note the similarity of Eq.6 with SMM objective (Eq.5) where the primary difference is the definition of I at trajectories τ level. D is defined loosely here, since I can be parameters of some distribution, or a distribution itself (see Table 1).

An important observation for the IM objective (Eq.6) is that **for any given trajectory τ, setting:**

$$\mathbf{z} = \mathbf{z}^\tau \quad \text{such that} \quad \mathbf{D}(\mathbf{I}_{\mathbf{z}^\tau}^{\mathbf{\Phi}}(\tau), \mathbf{I}_{\mathbf{z}^\tau}^*) = \mathbf{0} \tag{7}$$

will minimize the inner term divergence $\mathrm{D} = 0$ and therefore τ states and actions are optimal with respect to $\mathbf{z} = \mathbf{z}^\tau$ and samples of $(\tau_\mathbf{i}, \mathbf{z}^{\tau_\mathbf{i}})$ can be used to accelerate RL or do BC. We call these algorithms ***hindsight information matching (HIM)***[5] algorithms, and summarize how the existing methods, along with our proposals, related to one another in Table 1 with further details elaborated in Appendix C.

5 Distributional and Unsupervised Decision Transformer

Table 1 shows that all prior hindsight algorithms only care about matching the mean of information statistics, such as $z_{\text{goal_mean}}$ and $z_{\text{return_mean}}$, and ignores other distributional information within τ. The hindsight technique itself, however, can easily be adapted to match any statistical information of τ, even the full feature-marginal distribution $\sum_t \gamma^t \delta(\phi = \phi_t)$. Inspired by the recent successes in distributional RL (Bellemare et al., 2017; Dabney et al., 2018; Castro et al., 2018; Dabney et al., 2020), offline RL (Fujimoto et al., 2019; Jaques et al., 2020; Ghasemipour et al., 2021; Fujimoto & Gu, 2021; Chen et al., 2021a) and state-marginal matching (Lee et al., 2020; Ghasemipour et al., 2020; Gu et al., 2021), we introduce **Distributional Decision Transformer (DDT)** for offline and hindsight BC-based distribution matching as shown at the bottom of Table 1. The choice of BC as the algorithm is not arbitrary: a generic information matching objective in Eq.6 is intractable to solve with almost any sample-based RL algorithms[6][7] (consider different choices of divergence measures D (Nowozin et al., 2016; Li & Turner, 2016; Arjovsky et al., 2017) and moments to match through $I_z^\Phi(\tau)$), but in *hindsight* algorithms where Eq.7 by definition can be solved efficiently, **hindsight BC algorithms always work with samples $(\tau_\mathbf{i}, \mathbf{z}^{\tau_\mathbf{i}})$ irrespective of the choice of information statistics $\mathbf{I}_\mathbf{z}^\mathbf{\Phi}$**. The

[4]Recent mutual information maximization or empowerment methods (Eysenbach et al., 2019; Sharma et al., 2020; Choi et al., 2021) also make similar assumptions; see Gu et al. (2021) for more details.

[5]We also considered calling HIM as Hindsight Moment Matching (HMM) algorithms (Wainwright & Jordan, 2008; Li et al., 2015).

[6]Evolutionary strategies (Salimans et al., 2017), technically a non-RL black-box algorithm, could be applied, but accurate estimations of D in Eq.6 could require prohibitively many samples.

[7]DisCo RL (Nasiriany et al., 2021) conditions on a prameterized goal distribution and uses hindsight techniques; however, their RL objective in Equation 1 $\mathbb{E}_{s\sim\rho_z^\pi(s)} [\log p_z^*(s)]$, in contrast to a proper divergence objective like our Eq.3, is missing the $\mathcal{H}(\rho_z^\pi(s))$ entropy term and is essentially just solving for parameterized stationary reward maximization, as also stated in their Remark 1. This also corresponds to the MLE pseudo-objective benchmarked in Gu et al. (2021).

choice for the architecture is more flexible; however, we decided to use transformers (Vaswani et al., 2017b) in this work due to their tremendous scaling successes in language and vision domains (Dosovitskiy et al., 2020; Brown et al., 2020; Ramesh et al., 2021) and recent results in offline RL (Chen et al., 2021a; Janner et al., 2021).

While DDT applies to any I_z^Φ, we study a installation of it for offline reward-distribution or state-distribution matching: Categorical Decision Transformer (Table 1). See Algorithm 1 for the full pseudocode and the rest of details in Appendix D. We leave other possible instantiations, such as Gaussian or GMM DTs as future works.

Categorical Decision Transformer (CDT) Following the prior works (Bellemare et al., 2017; Furuta et al., 2021b), we employ the discretization of feature spaces to form categorical approximations of continuous distributions. Because we focus on low-dimensional feature spaces, e.g. Φ can be rewards or goal dimensions like XY velocities, such discretization scheme is sufficient. To compute $I_z^\Phi(\tau_{t:T})$ efficiently for all timesteps t given a trajectory $\tau_{1:T}$, we use similar recursive Bellman-like computation inspired by Bellemare et al. (2017).

5.1 Unsupervised Decision Transformer for Offline Inverse RL

Imitation learning setting does not assume that rewards r or some feature representations Φ are given. We therefore provide an extension of DT, **Unsupervised Decision Transformer (UDT)**, which does not receive any explicit feature function Φ and rather obtain it in an unsupervised manner. While UDT applies to both DT and DDT, in this work we exclusively study its combination with DT. The reason why DDT is unnecessary in the case of UDT is that if UDT learns sufficient features ϕ of s, matching their means, i.e. *moments*, are enough to match any distribution $p^*(s)$ to an arbitrary precision (Wainwright & Jordan, 2008; Li et al., 2015)[8]. **To the best of our knowledge, this is the first offline inverse RL imitation learning method, or full-state state-marginal matching method**[9], and as a bonus it is also **inherently multi-task and learns to match any trajectory segment statistics $\tau_{\mathbf{t:T}}$ in the training data**.

There are multiple possible design choices for UDT. First, we note that the DT objective (Chen et al., 2021a) itself is differentiable with respect to Φ, which means it can be used for learning Φ as we learn the policy. While we found some positive results for this single-objective approach, we observe that applying simple, popular unsupervised regularizers could result in substantially better results. Specifically, we experimented with (a) auto-encoder, and (b) contrastive loss (Srinivas et al., 2020; van den Oord et al., 2018). See Algorithm 1 for the pseudocode and the rest of details in Appendix E.

6 Experiments

We empirically investigate:

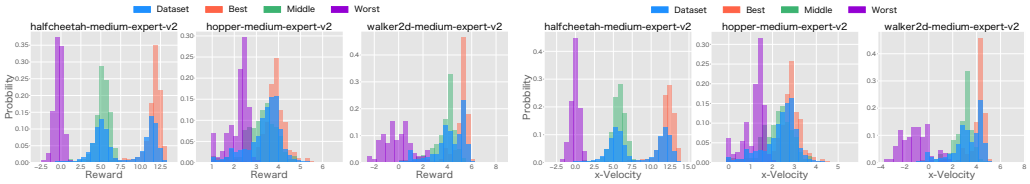
1. Can DDT match unseen reward distributions?
2. Can DDT match unseen state-feature distributions?
3. How does DDT behave with synthesized and unrealistic distributions?
4. Can UDT perform offline imitation learning in full state?

We experiment on the OpenAI Gym, MuJoCo tasks (`HalfCheetah-v3`, `Hopper-v3`, `Walker2d-v3`), a common benchmark for continuous control (Brockman et al., 2016; Todorov et al., 2012). Through the experiments, we use the `medium-expert` datasets in D4RL (Fu et al., 2020) to ensure the decent coverage of possible trajectories (Figure 2). At the beginning of the training, we sort all the trajectories based on their cumulative rewards, hold out five best trajectories and five 50 percentile trajectories as a test set (10 trajectories in total), and use the rest as a train

[8] We therefore also considered calling UDT as Moment Matching DT (MMDT).

[9] ValueDice (Kostrikov et al., 2019) had a small fully-offline inverse RL experiment in their Appendix, but they used sample size of 1 trajectory, i.e. 1000 timesteps, vs 1000 trajecotries, i.e. 1 million steps commonly used in offline RL (Fujimoto et al., 2019; Chen et al., 2021a) and in this work. Since it is known that offline RL performs highly reliably in this data regime (Matsushima et al., 2021) and their base algorithm (Nachum et al., 2019) has not yet been shown to be fully competitive in offline RL, we leave it for future work to do fair comparisons between UDT and their algorithm.

set. For the quantitative evaluations of distribution matching, we utilize the Wasserstein-1 distance between one-dimensional categorical distributions of features in rollout or test set trajectories, where the smaller the better. We report the results averaged over 20 rollouts per random seed (and run four seeds) to reduce the variance. Unless otherwise specified, we use Categorical DT (CDT) for DDT experiments as it can represent multi-modal distributions. We share our implementation to ensure the reproducibility[10]; see supplemental materials for the details.



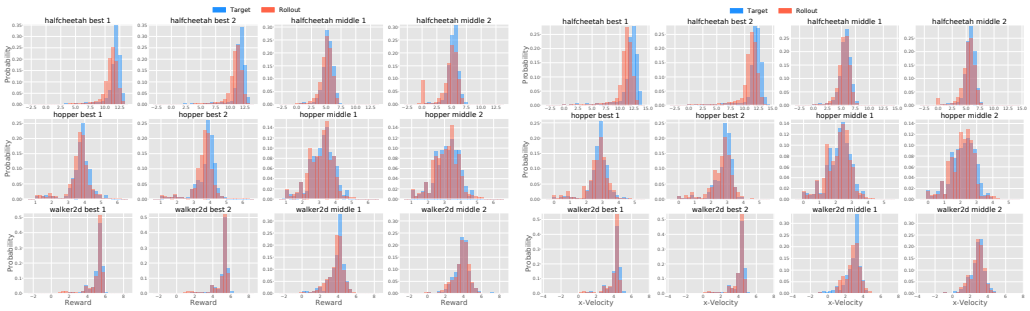
(a) Reward (b) x-Velocity

Figure 2: Distributions of the features (reward, and x-velocity) in the datasets. When we sort all the trajectories based on their cumulative rewards, each quality of trajectory (best, middle, worst) shows a different shape of distribution, and the distribution of the whole dataset seems a weighted mixture of those.

6.1 Reward and State-feature Matching

First, we qualitatively evaluate whether DDT could match its rollout to the target distribution. We choose reward and state-feature, such as x-velocity of the agents, as feature spaces for DDT to match. To specify the target distribution during the evaluation, we construct the categorical representation of the test set trajectories, and feed them to the DDT (i.e. CDT).

Figure 3 visualize the results of a part of test trajectories, and shows that both reward and state-feature distributions of DDT induced from its rollout trajectories successfully match the target distributions from the test set that are unseen during training. In addition, we quantitatively compare DDT against deterministic (original) DT from distributional matching perspective. See Appendix F (Table 4 and Table 5) for the results.



(a) Reward (b) x-Velocity

Figure 3: (a) Reward and (b) state-feature distribution matching in halfcheetah (top), hopper (middle), and walker2d (bottom). The left two examples are the distributions from the best trajectories, and right two are the distributions from the middle trajectories. All these examples are from the test set that consists of 10 trajectories. The rollout distributions of DDT (red) match the target distributions (blue) very well in all cases.

6.2 Generalization to Unseen Target Distribution

To test the generalization to more extreme unseen, out-of-distribution cases, we intervene the target distributions in two different ways: (1) shifting the target distributions in the test set by constant offsets, and (2) synthesizing novel distributions via python scripts, as done in prior works (Ghasemipour et al., 2020; Gu et al., 2021). (1) could generate the seemingly realistic reward and velocity profiles, while (2) becomes likely unrealistic ones.

[10]We will publicly release in the de-anonymized version.

6.2.1 SHIFTING THE TARGET DISTRIBUTION

To generate the unseen but within the support of dataset distribution, we make the reward and velocities values of trajectories in the test set shift with a constant offset of: `bin_size` $\times\{-3.0, -2.0, -1.0, 0.0, +1.0, +2.0, +3.0\}$. Table 2 shows the qualitative comparison between CDT and original (deterministic) DT, based on Wasserstein metric between two distributions. CDT successfully handle the distribution shifts especially in halfcheetah datasets. In Appendix B, we provide the visualization results. It reveals that CDT can match the rollouts to the shifted target distributions when the shifted targets are within the support of dataset distributions (according to Figure 2). We further perform the same tests for the original DT with return-mean contexts, which also shows the generalization in some domains (e.g. walker middle trajectory), but not in others (e.g. hopper middle trajectory).

Method	halfcheetah			hopper			walker2d		
	Expert	Medium	Total	Expert	Medium	Total	Expert	Medium	Total
Categorical DT (ours)	**0.00815** ± **0.00413**	**0.00711** ± **0.00235**	**0.00763** ± **0.00340**	**0.00735** ± **0.00275**	0.00774 ± 0.00343	**0.00755** ± **0.00311**	0.01756 ± 0.00574	**0.00723** ± **0.00212**	0.01239 ± 0.00673
Deterministic DT	0.01103 ± 0.00413	0.01092 ± 0.00647	0.01098 ± 0.00543	0.01065 ± 0.00223	**0.00483** ± **0.00138**	0.00774 ± 0.00345	**0.00611** ± **0.00415**	0.01076 ± 0.00484	**0.00844** ± **0.00508**

Table 2: Shifted x-velocity distribution matching.We compare Categorical DT and Deterministic DT, and evaluate with Wasserstein metric (smaller is better).

6.2.2 SYNTHESIZING UNREALISTIC TARGET DISTRIBUTION

We synthesize six target distributions generating x-velocity samples from Gaussian distributions via python scripts (see Appendix B for the details), as done in prior works (Ghasemipour et al., 2020; Gu et al., 2021). While these targets are designed at least within the support of the dataset, we do not consider physical realizability. Although they might be unrealistic, Figure 4 implies that CDT tends to match the target distributions, even in the cases of bi-modal target distributions. Such generalization to synthesized distribution is an important benefit of distribution-conditioned training. We also quantify the performance of DDT against deterministic (original) DT from distributional matching perspective. See Appendix F (Table 7) for the results.

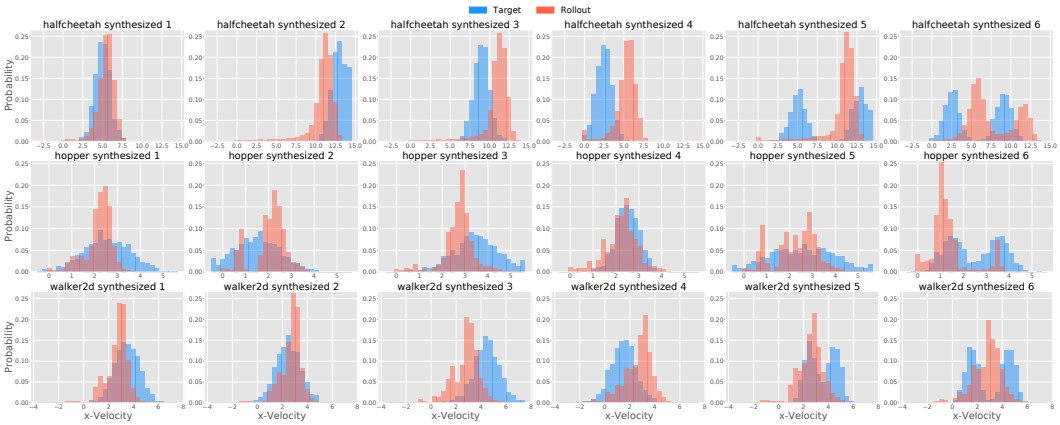


Figure 4: State-feature distribution matching in halfcheetah (top), hopper (middle), and walker2d (bottom). We synthesize the each target distribution from Gaussian distributions (see Appendix B for the details). While the results are worse than for realistic test target distributions in Figure 3, considering that many of these arbitrary synthetic targets could be unrealizable, seeing bi-modal matching results show that indeed CDT has learned to generalize something non-trivial.

6.3 UNSUPERVISED DT FOR OFFLINE IMITATION LEARNING

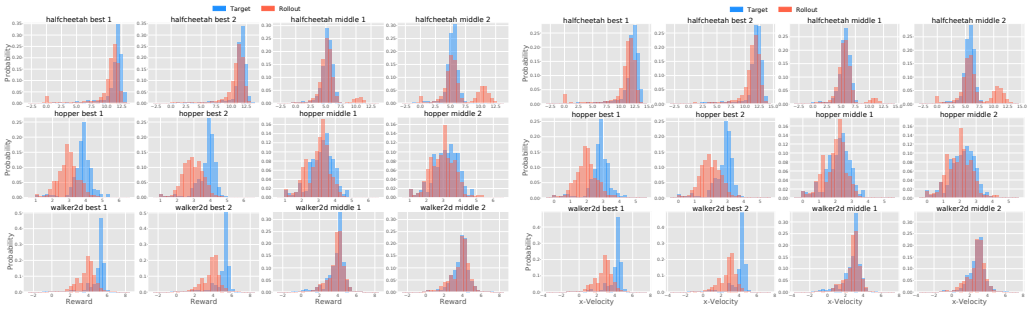
Lastly, we investigate UDT for offline imitation learning, where we do not observe rewards nor state features. We optimize m-dimensional ϕ from the encoder $\Phi(s)$ with $m \in \{1, 4, 16\}$. For unsupervised loss (UL) for Φ, we use auto-encoder (AE) or contrastive loss. For supervise loss (SL), we only use DT objective. We test the following settings for optimizing the encoder Φ:

- SL + UL: Optimize with supervised and unsupervised losses.

- UL: Only trains with unsupervised loss.
- SL: Only trains with supervised (DT) loss.
- UL-Fix: Pre-trains with unsupervised loss and fix the encoder during policy optimization.

We evaluate UDT with a similar hold-out distribution matching test as done in Figure 3, except now the states are processed through learned feature function Φ. Similar to our state-marginal matching evaluations, we are essentially evaluating a multi-task (conditional) policy on one-shot imitation performance.

As an example, we shows the visualization results for UL only training with $m = 4$ and AE loss (Figure 5), which manage to match the target trajectory distributions successfully, while other strategies or smaller dimensions of ϕ fail to capture the sufficient information statistics to match the dataset with multiple modes (such as halfcheetah, hopper medium, walker medium). We found that UL, SL + UL, UL-Fix benefits from the multi-dimensional representations ($m = 4, 16$), while SL only doesn't improve the performance over $m = 1$ case. Also see Appendix E for further details and qualitative results, and Appendix F (Table 8 and Table 9) for quantitative result summaries.



(a) Reward (b) x-Velocity

Figure 5: (a) Reward and (b) State-feature distribution matching in halfcheetah (top), hopper (middle), and walker2d (bottom). We train the encoder with auto-encoder loss to obtain $m = 4$ dimensional representations for the information statistics to match.

7 Conclusion

We provide a unified perspective on a wide range of *hindsight* algorithms, and generalize the problem formulation as hindsight information matching (HIM). Inspired by recent successes in distributional matching in online RL and applications of transformers in offline RL, we propose Distributional Decision Transformer (DDT) – with a instance as Categorical – and Unsupervised DT (UDT) for offline state-marginal matching and inverse-RL imitation learning respectively. We hope our proposed framework, algorithms, and benchmarks can empower the community to make progress in these two important sub-problems, and also inspire more supervised sequence modeling approaches in RL beyond classic reward maximization.

ETHICS STATEMENT

Since this paper mainly focuses on the reinterpretation of hindsight RL algorithms and experiments on existing benchmark datasets, we believe there are no ethical concerns.

REPRODUCIBILITY STATEMENT

We share our codes to ensure the reproductivity (and plan to open-source in a de-anonymized version). See supplemental materials. The details of hyperparameters are described in Appendix A and those of experiments are described in the beginning of Section 6, Appendix B, and E. We report the results averaged over 20 rollouts per random seed (and run four seeds) to reduce the variance.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in neural information processing systems*, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, 2017.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Kohd, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *arXiv preprint arXiv:1807.01675*, 2018.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G Bellemare. Dopamine: A research framework for deep reinforcement learning. *arXiv preprint arXiv:1812.06110*, 2018.

Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jake Varley, Alex Irpan, Benjamin Eysenbach, Ryan Julian, Chelsea Finn, and Sergey Levine. Actionable models: Unsupervised offline reinforcement learning of robotic skills. *arXiv preprint arXiv:2104.07749*, 2021.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021b.

Jongwook Choi, Archit Sharma, Honglak Lee, Sergey Levine, and Shixiang Shane Gu. Variational empowerment as representation learning for goal-based reinforcement learning. In *International Conference on Machine Learning*, 2021.

Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Will Dabney, Zeb Kurth-Nelson, Naoshige Uchida, Clara Kwon Starkweather, Demis Hassabis, Rémi Munos, and Matthew Botvinick. A distributional code for value in dopamine-based reinforcement learning. *Nature*, 577(7792):671–675, 2020.

Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. On the sample complexity of the linear quadratic regulator. *Foundations of Computational Mathematics*, 20(4):633–679, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning diverse skills without a reward function. In *international conference on learning representations*, 2019.

Benjamin Eysenbach, Xinyang Geng, Sergey Levine, and Ruslan Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement. In *Advances in neural information processing systems*, 2020.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2021.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2106.06860*, 2021.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, 2019.

Hiroki Furuta, Tadashi Kozuno, Tatsuya Matsushima, Yutaka Matsuo, and Shixiang Shane Gu. Co-adaptation of algorithmic and implementational innovations in inference-based deep reinforcement learning. *arXiv preprint arXiv:2103.17258*, 2021a.

Hiroki Furuta, Tatsuya Matsushima, Tadashi Kozuno, Yutaka Matsuo, Sergey Levine, Ofir Nachum, and Shixiang Shane Gu. Policy information capacity: Information-theoretic measure for task complexity in deep reinforcement learning. In *International Conference on Machine Learning*, 2021b.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, 2020.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, 2021.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*, 2021.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, 2016.

Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, Bernhard Schölkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. *arXiv preprint arXiv:1706.00387*, 2017.

Shixiang Shane Gu, Manfred Diaz, C Daniel Freeman, Hiroki Furuta, Seyed Kamyar Seyed Ghasemipour, Anton Raichuk, Byron David, Erik Frey, Erwin Coumans, and Olivier Bachem. Braxlines: Fast and interactive toolkit for rl-driven behavior generation beyond reward maximization, 2021.

Jiaming Guo, Rui Zhang, Xishan Zhang, Shaohui Peng, Qi Yi, Zidong Du, Xing Hu, Qi Guo, and Yunji Chen. Hindsight value function for variance reduction in stochastic dynamic environment. *arXiv preprint arXiv:2107.12216*, 2021.

Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

Elad Hazan, Sham M. Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, 2019.

Nicolas Heess, Greg Wayne, David Silver, Timothy Lillicrap, Yuval Tassa, and Tom Erez. Learning continuous control policies by stochastic value gradients. *arXiv preprint arXiv:1510.09142*, 2015.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, 2016.

Michael Janner, Qiyang Li, and Sergey Levine. Reinforcement learning as one big sequence modeling problem. *arXiv preprint arXiv:2106.02039*, 2021.

Natasha Jaques, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Shane Gu, and Rosalind Picard. Human-centric dialog training via offline reinforcement learning. *arXiv preprint arXiv:2010.05848*, 2020.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low bellman rank are pac-learnable. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2017.

Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence*, 1993.

Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. *arXiv preprint arXiv:1912.05032*, 2019.

Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019.

Vitaly Kurin, Maximilian Igl, Tim Rocktäschel, Wendelin Boehmer, and Shimon Whiteson. My body is a cage: the role of morphology in graph-based incompatible control. *arXiv preprint arXiv:2010.01856*, 2020.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2020.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Alexander C. Li, Lerrel Pinto, and Pieter Abbeel. Generalized hindsight for reinforcement learning. In *Advances in neural information processing systems*, 2020.

Yingzhen Li and Richard E Turner. Variational inference with rényi divergence. *stat*, 1050:6, 2016.

Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pp. 1718–1727. PMLR, 2015.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, 2019.

Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Shane Gu. Deployment-efficient reinforcement learning via model-based offline optimization. In *International Conference on Learning Representations*, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in neural information processing systems*, 2018.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Soroush Nasiriany, Vitchyr H. Pong, Ashvin Nair, Alexander Khazatsky, Glen Berseth, and Sergey Levine. Disco rl: Distribution-conditioned reinforcement learning for general-purpose policies. In *International Conference on Robotics and Automation*, 2021.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 271–279, 2016.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning*, pp. 7487–7498. PMLR, 2020.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control. *International Conference on Learning Representations*, 2018.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021.

Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. In *Advances in neural information processing systems*, 2019.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International conference on learning representations*, 2020.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, 2020.

Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

Yujin Tang and David Ha. The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning. *arXiv preprint arXiv:2109.02869*, 2021.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.

Stephen Tu and Benjamin Recht. The gap between model-based and model-free methods on the linear quadratic regulator: An asymptotic viewpoint. In *Conference on Learning Theory*, pp. 3036–3083. PMLR, 2019.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017a.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017b.

David Venuto, Elaine Lau, Doina Precup, and Ofir Nachum. Policy gradients incorporating the future. *arXiv preprint arXiv:2108.02096*, 2021.

Martin J Wainwright and Michael Irwin Jordan. *Graphical models, exponential families, and variational inference*. Now Publishers Inc, 2008.

Mengjiao Yang and Ofir Nachum. Representation matters: Offline pretraining for sequential decision making. In *International Conference on Machine Learning*, 2021.

Rui Zhao and Volker Tresp. Energy-based hindsight experience prioritization. In *Conference on Robot Learning*, 2018.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

APPENDIX

A HYPER-PARAMETER OF DECISION TRANSFORMERS

We implement Distributional Decision Transformer and Unsupervised Decision Transformer, built upon the official codebase released by Chen et al. (2021a) (https://github.com/kzl/decision-transformer). We follow the most of hyper-parameters as they did (Table 3).

Hyperparameter	**Value**
Number of layers	3
Number of attention heads	1
Embedding dimension	128
Nonlinearity function	ReLU
Batch size	64
Context length N	20
Dropout	0.1
Learning rate	1e-4
Grad norm clip	0.25
Weight decay	1e-4
Learning rate decay	Linear warmup for first 100k training steps
Training (Gradient) steps	1M
Number of bins for categorical distribution	31
Encoder size	2 layer MLP, (128, 128)
Coefficient of unsupervised loss	0.1

Table 3: Hyperparameters of original, Distributional, and Unsupervised Decision Transformer. We refer Chen et al. (2021a).

B DETAILS OF EXPERIMENTS

In this section, we describe the details of synthesizing unrealistic distribution, and the visualization of shifting target distribution.

For the experiments of Section 6.2.2, we synthesize six target distributions per environment manually via python script. Since we consider the one-dimensional feature space (x-velocity), we simply specify the mean and standard deviation of Gaussian distributions referring each dataset distribution, as shown in Figure 2, and then generate the 1000 samples per trajectory. While these targets are designed at least within the support of the dataset, we don't consider physical realizability.

- **HalfCheetah** (μ, σ) = (5.0, 1.0), (13.0, 1.0), (9.0, 1.0), (2.5, 1.0), {(5.0, 1.0), (13.0, 1.0)}, {(9.0, 1.0), (2.5, 1.0)}.
- **Hopper** (μ, σ) = (2.5, 1.0), (1.5, 1.0), (3.5, 1.0), (2.5, 0.5), {(3.5, 1.0), (1.5, 1.0)}, {(3.5, 0.5), (1.5, 0.5)}.
- **Walker2d** (μ, σ) = (3.5, 1.0), (2.5, 1.0), (4.5, 1.0), (1.5, 1.0), {(2.5, 0.5), (4.5, 0.5)}, {(1.5, 0.5), (4.5, 0.5)}.

For the last two, we sampled from two gaussian distributions 500 samples each.

B.1 ADDITIONAL RESULTS

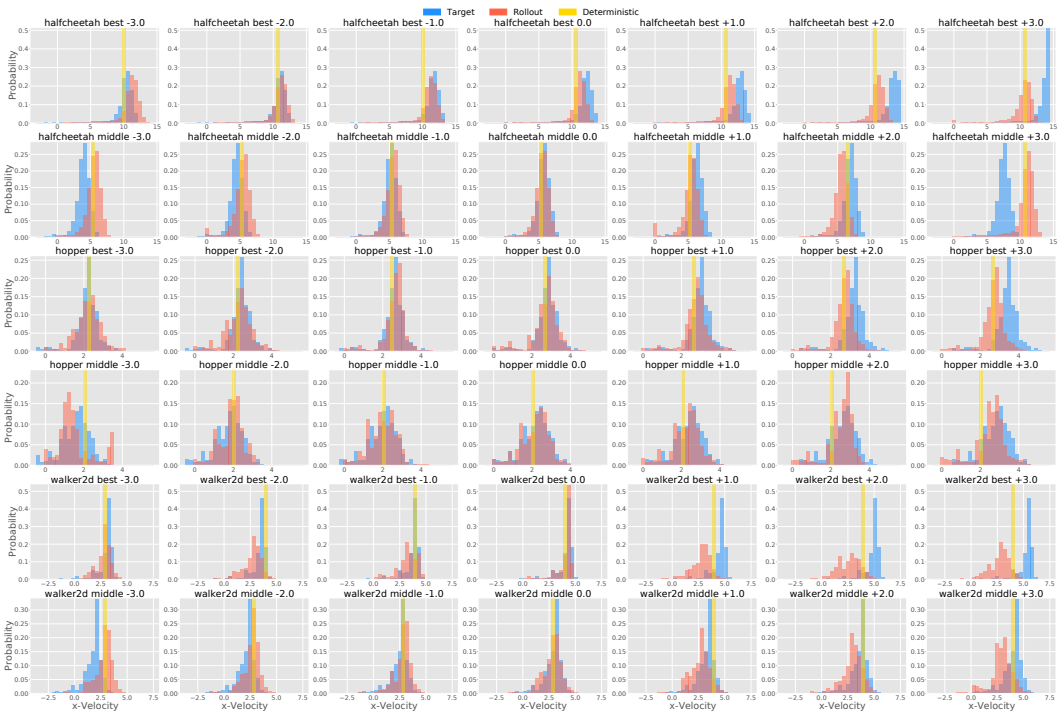


Figure 6: State-feature distribution matching, especially x-velocity, in halfcheetah (top two rows; best and middle), hopper (middle two rows; best and middle), and walker2d (bottom two rows; best and middle). We shift the target distribution with constant offset (from left to right column); `bin_size` $\times\{-3.0, -2.0, -1.0, 0.0, +1.0, +2.0, +3.0\}$. Distributional DT (red) can match the rollouts to the shifted target distributions (blue) when the shifted targets are within the support of dataset distributions.

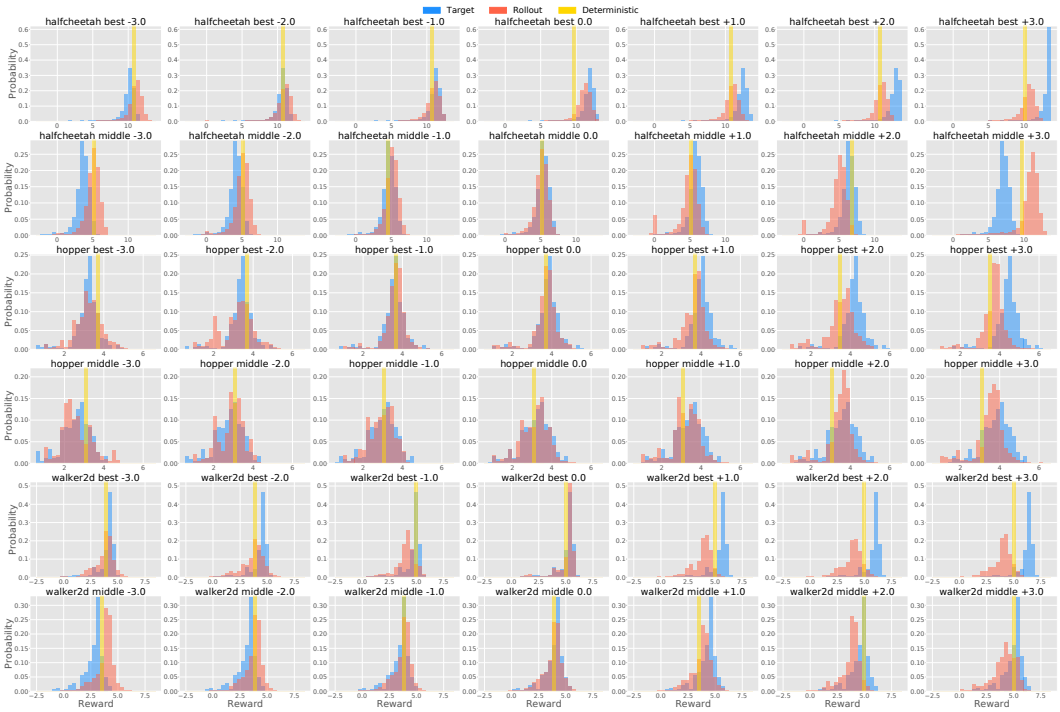


Figure 7: Reward distribution matching in halfcheetah (top two rows; best and middle), hopper (middle two rows; best and middle), and walker2d (bottom two rows; best and middle). We shift the target distribution with (from left to right column); `bin_size` $\times\{-3.0, -2.0, -1.0, 0.0, +1.0, +2.0, +3.0\}$.

C Details for Hindsight Information Matching

We further elaborate on the details summarized in Table 1. We essentially have the following three types of algorithms:

- **Goal-conditioned RL (GCRL)/BC** Commonly $\Phi(s, a)$ only depends on s and corresponds to low-dimensional features of the state space, such as XY positions or velocities of the agent or manipulated objects (Lee et al., 2020; Ghasemipour et al., 2020; Gu et al., 2021). z parameterizes the desired goal for features $z = \Phi(s, a)$ (i.e. $Z = F$). Desired information statistics I_z^* can be viewed as a Gaussian with mean z for dense goal-distance rewards, or a Dirac with probability of 1 at z for sparse goal-reaching rewards, with the divergence D as the reverse KL (up to an entropy regularizer). IM function $I_z^\Phi(\tau) \propto \sum_t \gamma^t \delta(\phi = \phi_t)$, i.e. a non-parametric distribution on all states in the trajectory. At test time, $p(z)$ can be some manually-defined goal distribution (Pong et al., 2018; Gu et al., 2021) or from explored samples (Andrychowicz et al., 2017; Pong et al., 2019).
- **Returns-conditioned RL/BC** In the cases of Kumar et al. (2019); Srivastava et al. (2019); Chen et al. (2021a); Janner et al. (2021), we view $\Phi(s, a) = r(s, a)$, i.e. rewards are just a feature of states and actions. $z = I_z^*$ parameterizes the desired mean for the sum of returns in trajectory τ, and $I_z^\Phi(\tau) \propto \sum_t \gamma^t \phi_t$. At test time, $p(z)$ of interest is primarily the high return values, allowing some beyond the maximal values used during training (Chen et al., 2021a).
- **Multi-task RL/BC** Contrary to the above two, Eysenbach et al. (2020); Li et al. (2020) deal purely with reward maximization. Following Eysenbach et al. (2020), we use the optimality reward transformation in control-as-inference literature (Furuta et al., 2021b; Levine, 2018) to draw their connections to Eq.6. Specifically, we define $z = \{z_{\text{task}}, z_{\text{opt}}\}$ where $z_{\text{task}} \in Z_{\text{task}}$ parameterizes the reward function $r(s, a, z_{\text{task}})$ and $z_{\text{opt}} \in \{0, 1\}$ is the optimality variable (Eysenbach et al., 2020). Setting $I_z^* = I_{z_{\text{opt}}}^* = z_{\text{opt}} = 1$, $I_z^\Phi(\tau) = I_{z_{\text{task}}}^\Phi(\tau) \propto \exp\left(\sum_t \gamma^t \phi(s_t, a_t, z_{\text{task}})\right)$, solving Eq.7 for z_{task} results in a similar algorithm as Li et al. (2020); Eysenbach et al. (2020). At test time, $p(z_{\text{task}})$ is defined manually, while $p(z_{\text{opt}}) = \delta(z_{\text{opt}} = 1)$ as also during training.

D Details for Categorical Decision Transformer

Categorical Decision Transformer (CDT) takes histograms or categorical variable approximations to feature distributions as the inputs of transformer. Here we describe how to compute the distributions for all timesteps given a trajectoy $t \in [0, 1, \ldots, T]$. For the simplicity, we explain the case of one-dimensional feature (e.g. scalar reward), but for n-dimensional features, we can adopt following procedure per each dimension and arrive at n categorical features. This essentially approximates the full joints with a product of independent marginal distribution per dimension, and ensures that number of samples for getting reasonably discretized approximations do not need to grow exponentially as the dimension grows.

At first, we discretize the feature space F using B bins (per dimension), and convert the feature ϕ_t into the one-hot representation $\tilde{\phi}_t$. The range of feature space $[\phi_{\min}, \phi_{\max}]$ is pre-defined from the given offline datasets. $z_{\Phi_{\text{hist}}}(t)$, the categorical feature distribution at time step t, can be computed recursively following Bellman-like equation:

$$z_{\Phi_{\text{hist}}}(t) \propto \tilde{\phi}_t + \gamma(1 - \mathbb{1}[t = T]) z_{\Phi_{\text{hist}}}(t + 1), \tag{8}$$

where $\mathbb{1}$ is the indicator function. We compute a series of $z_{\Phi_{\text{hist}}}$ in a backward manner starting from T. After we obtain the desired information statistics for all trajectories, we feed them to the categorical transformer during training/test time. We describe the python-like pseudocode in Algorithm 1, coloring the changes from the original Decison Transformer (Chen et al., 2021a).

Algorithm 1 **Distributional/Unsupervised Decision Transformer Pseudocode**: Red and green texts describe additional details on top of the base DT pseudocode from Chen et al. (2021a).

```
# z: information statistics (e.g. histogram, mean & std)
# s, a, t: states, actions, or timesteps
# transformer: transformer with causal masking (GPT)
# embed_s, embed_a, embed_z: linear embedding layers
# encorder_s: linear layers for representation learning
# embed_t: learned episode positional embedding
# pred_a: linear action prediction layer
# unsupervised_loss: e.g. auto-encoder, contrastive objective
# compute_stats: a function to compute information statistics

# main model
def DecisionTransformer(z, s, a, t):
    # compute embeddings for tokens
    pos_embedding = embed_t(t)
    s_embedding = embed_s(s) + pos_embedding
    a_embedding = embed_a(a) + pos_embedding
    if distributional:
        z_embedding = embed_z(z) + pos_embedding
    elif unsupervised:
        if z is None:
            z_embedding = embed_z(StopGrad[encoder_s(s)]) + pos_embedding
        else:
            z_embedding = embed_z(z) + pos_embedding

    input_embeds = stack(z_embedding, s_embedding, a_embedding)

    # use transformer to get hidden states
    hidden_states = transformer(input_embeds=input_embeds)

    # select hidden states for action prediction tokens
    a_hidden = unstack(hidden_states).actions

    # predict action
    return pred_a(a_hidden)

# training loop
train_z = compute_stats(train_dataset)
# dims: (batch_size, K, dim)
for z, (s, a, t) in zip(train_z, train_dataset):
    if unsupervised:
        z = None
    a_preds = DecisionTransformer(z, s, a, t)
    loss = mean((a_preds - a)**2) + unsupervised_loss(encoder_s(s))
    optimizer.zero_grad(); loss.backward(); optimizer.step()

# evaluation loop
test_z = compute_stats(test_dataset)
n_tests = len(test_dataset)
for index in range(n_tests):
    test_trajectory = test_dataset[index]
    max_time_steps = len(test_trajectory)
    s, a, t, done = [env.reset()], [], [1], False
    # conditioning on the desired information statistics
    if distributional:
        z = [test_z[index][0]]
    elif unsupervised:
        z = [encoder_s(test_trajectory['observations'][0])]
    for t in range(max_time_steps):
        action = DecisionTransformer(z, s, a, t)[-1]
        new_s, r, done, _ = env.step(action)
        # append new tokens to sequence
        if distributional:
            z = z + [test_z[index][t+1]]
        elif unsupervised:
            z = z + [encoder_s(test_trajectory['observations'][t+1])]
        s, a, t = s + [new_s], a + [action], t + [len(z)]
        z, s, a, t = z[-K:], ...  # only keep context length of K
```

E DETAILS OF UNSUPERVISED DECISION TRANSFORMER

While any unsupervised regularizer could be combined into Unsupervised Decision Transformer, we observe that simple and popular objectives, such as auto-encoder and contrastive loss, performs better. For auto-encoder regularization, we train the MLP encoder and decoder with MSE loss of current state reconstruction:

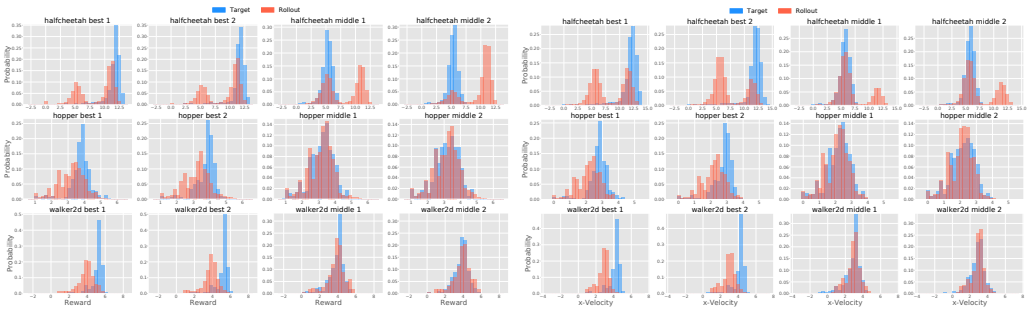
$$\min_{\psi} \mathbb{E}_{D \sim s_t} \left[\| s_t - \text{decoder}_{\psi}(\text{encoder}_{\psi}(s_t)) \|^2 \right]. \tag{9}$$

Then, we use the output of the encoder as a learned information statistics. In addition, for contrastive loss, we adopt CURL objective (Srinivas et al., 2020) for state input while removing data augmentation. We train the MLP encoder as a query, use its momentum encoder as a key, and treat its trained encoder output as a learned information statistics. It remains as future works to combine more advanced objectives, such as attentive contrastive learning (Yang & Nachum, 2021).

Also, we sweep the dimension of the representations; $m \in \{1, 4, 16\}$, and optimization of the encoder (combining unsupervised loss and DT loss jointly, training unsupervised loss only, training DT loss only, pre-training unsupervised loss and freezing the encoder parameters, and pre-training unsupervised loss and fine-tuning with DT loss).

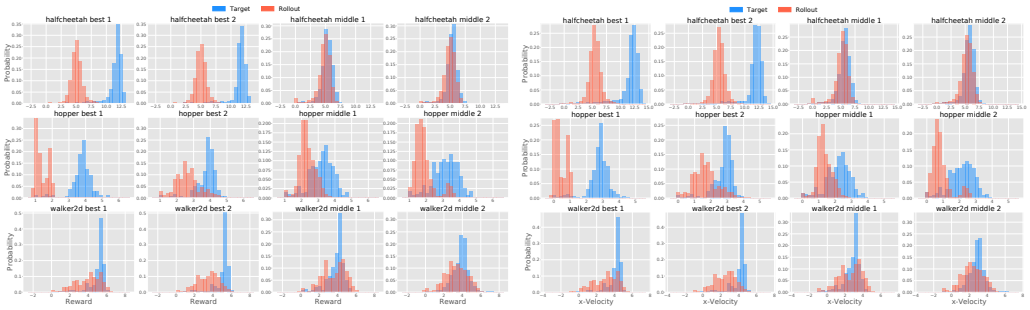
E.1 Additional Results

For the comparison, we just perform Transformer-BC, not conditioning on any information statistics (Figure 8). Also, we train the representation with (1) only DT loss (Figure 9 to 11), (2) auto-encoder loss + DT loss (Figure 12 to 23), and (3) contrastive loss + DT loss (Figure 24 to 32).



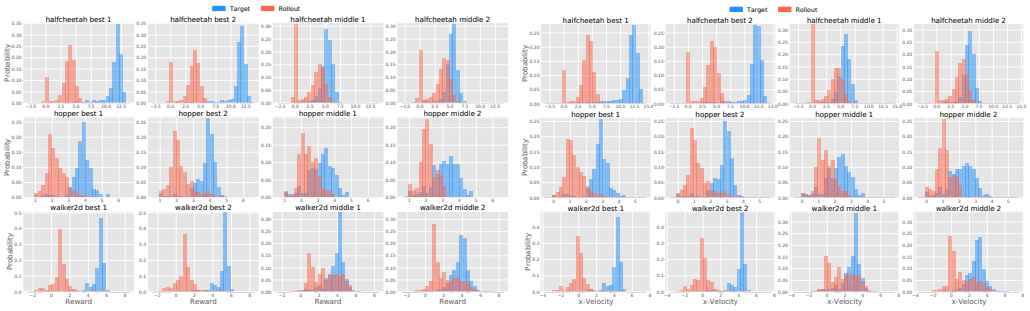
(a) Reward (b) x-Velocity

Figure 8: (a) Reward and (b) State-feature distribution matching. We perform Transformer-BC, not conditioning on any information statistics.



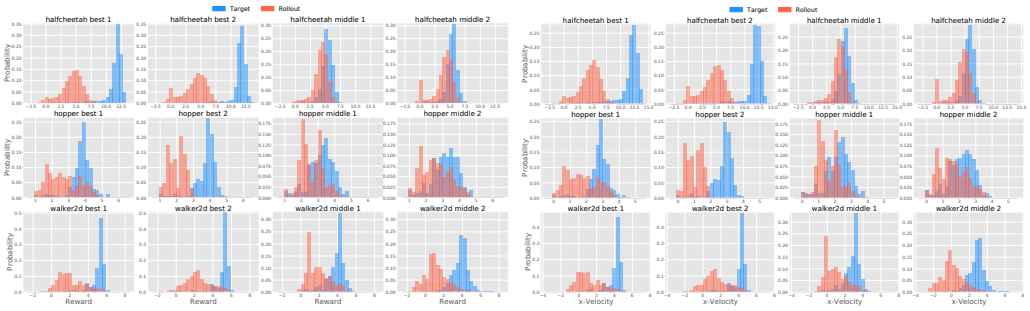
(a) Reward (b) x-Velocity

Figure 9: (a) Reward and (b) State-feature distribution matching. We train the encoder with BC loss to obtain 1 dimensional representations for the information statistics to match.



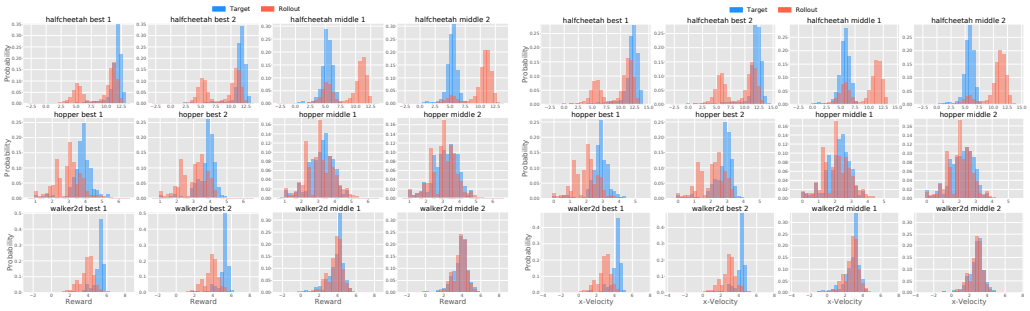
(a) Reward (b) x-Velocity

Figure 10: (a) Reward and (b) State-feature distribution matching. We train the encoder with BC loss to obtain 4 dimensional representations for the information statistics to match.



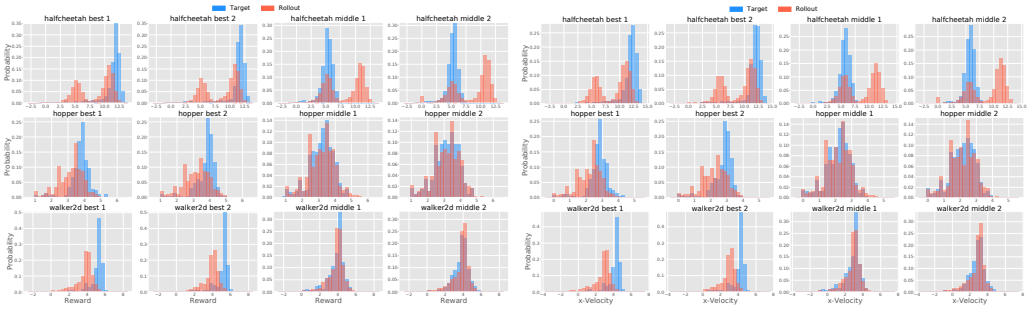
(a) Reward (b) x-Velocity

Figure 11: (a) Reward and (b) State-feature distribution matching. We train the encoder with BC loss to obtain 16 dimensional representations for the information statistics to match.



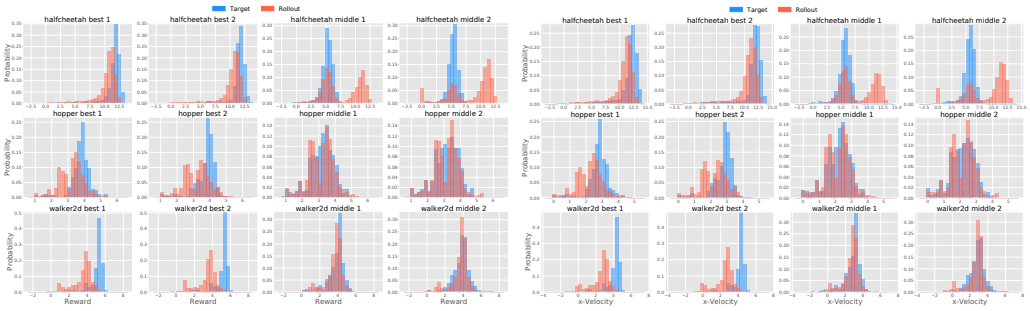
(a) Reward (b) x-Velocity

Figure 12: (a) Reward and (b) State-feature distribution matching. We train the encoder with auto-encoder + BC loss to obtain 1 dimensional representations for the information statistics to match.



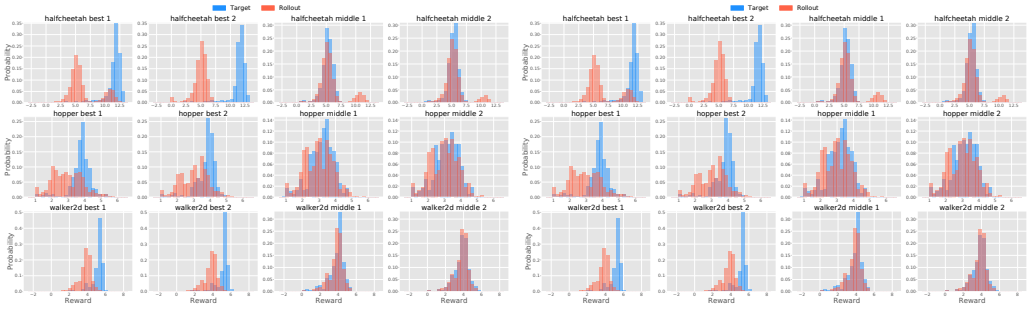
(a) Reward (b) x-Velocity

Figure 13: (a) Reward and (b) State-feature distribution matching. We train the encoder with auto-encoder loss to obtain 1 dimensional representations for the information statistics to match.



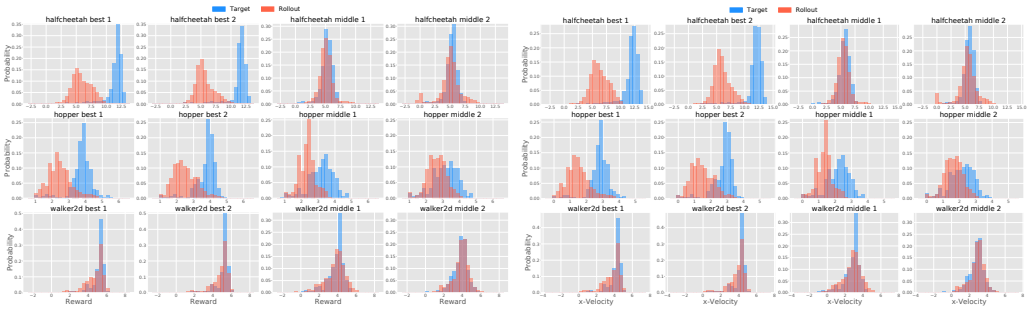
(a) Reward (b) x-Velocity

Figure 14: (a) Reward and (b) State-feature distribution matching. We train the encoder with pre-training of auto-encoder loss to obtain 1 dimensional representations for the information statistics to match.



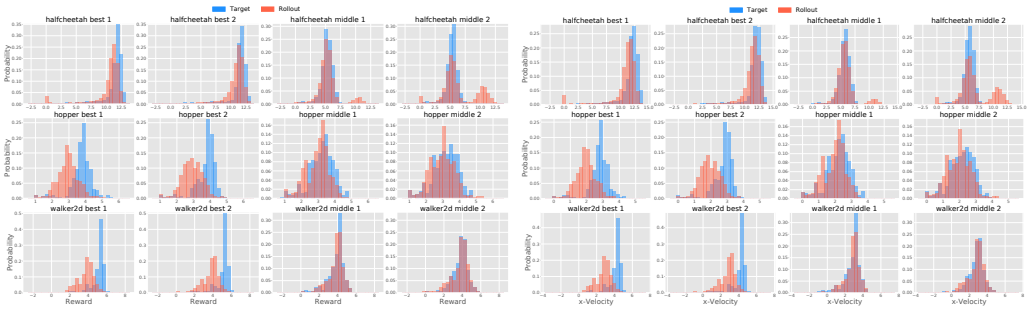
(a) Reward (b) x-Velocity

Figure 15: (a) Reward and (b) State-feature distribution matching in halfcheetah (top), hopper (middle), and walker2d (bottom). We train the encoder with contrastive loss to obtain 4 dimensional representations for the information statistics to match.



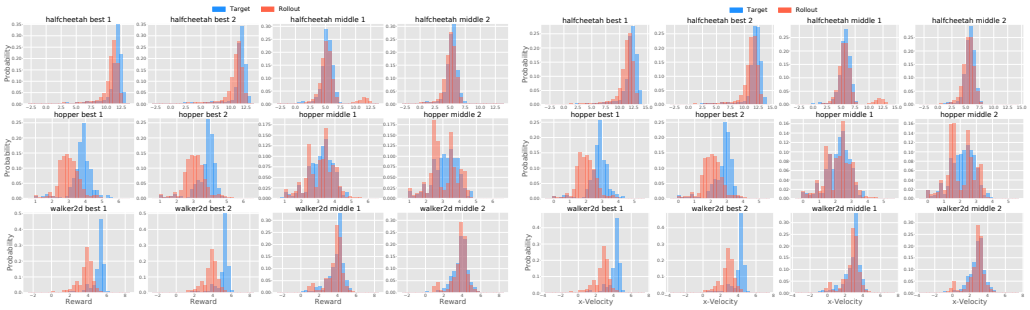
(a) Reward (b) x-Velocity

Figure 16: (a) Reward and (b) State-feature distribution matching. We train the encoder with auto-encoder + BC loss to obtain 4 dimensional representations for the information statistics to match.



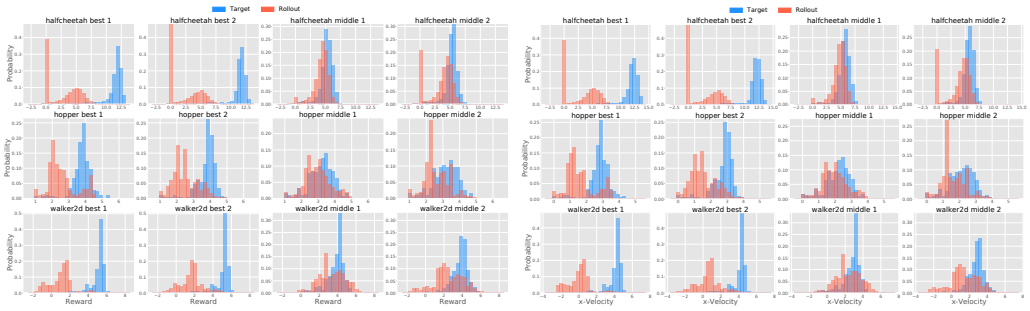
(a) Reward (b) x-Velocity

Figure 17: (a) Reward and (b) State-feature distribution matching. We train the encoder with auto-encoder loss to obtain 4 dimensional representations for the information statistics to match.



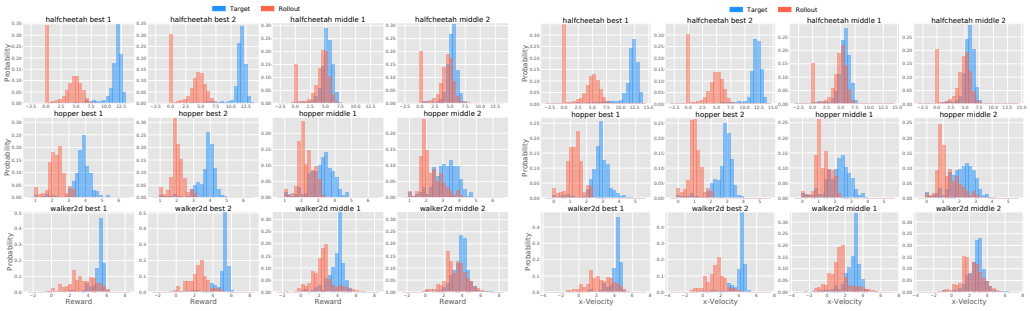
(a) Reward (b) x-Velocity

Figure 18: (a) Reward and (b) State-feature distribution matching. We train the encoder with pre-training of auto-encoder loss to obtain 4 dimensional representations for the information statistics to match.



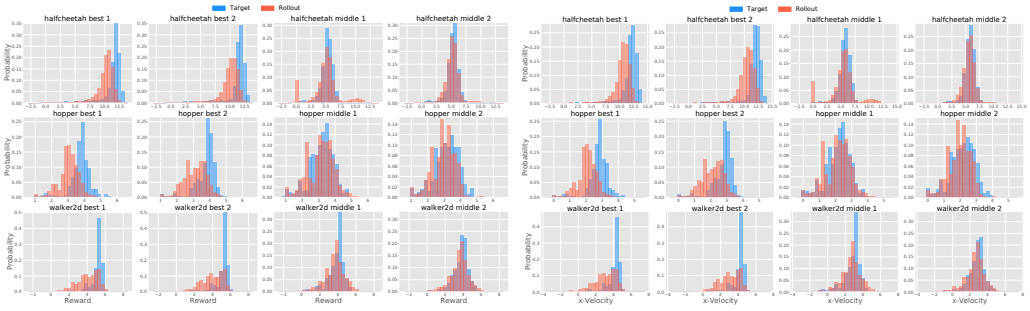
(a) Reward (b) x-Velocity

Figure 19: (a) Reward and (b) State-feature distribution matching. We train the encoder with pre-training of auto-encoder loss and finetuning of BC loss to obtain 4 dimensional representations for the information statistics to match.



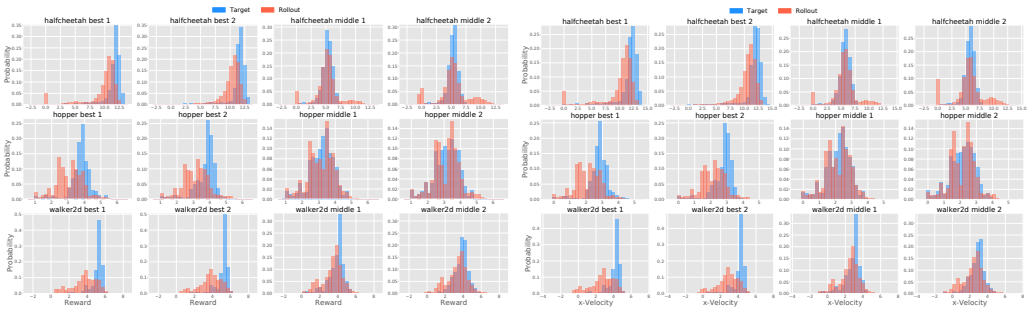
(a) Reward (b) x-Velocity

Figure 20: (a) Reward and (b) State-feature distribution matching. We train the encoder with auto-encoder + BC loss to obtain 16 dimensional representations for the information statistics to match.



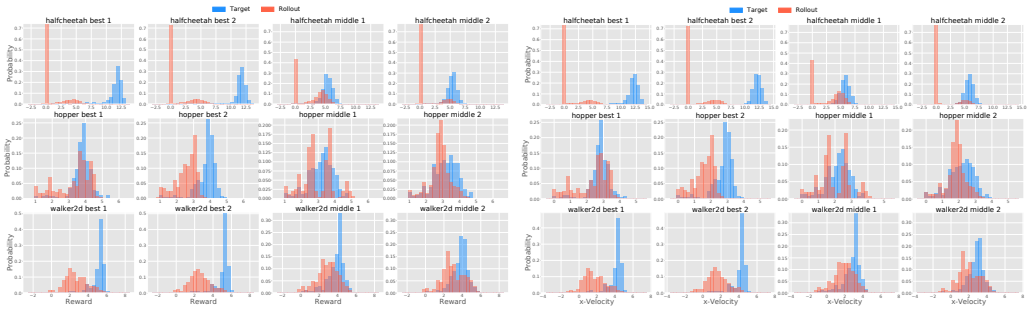
(a) Reward (b) x-Velocity

Figure 21: (a) Reward and (b) State-feature distribution matching. We train the encoder with auto-encoder loss to obtain 16 dimensional representations for the information statistics to match.



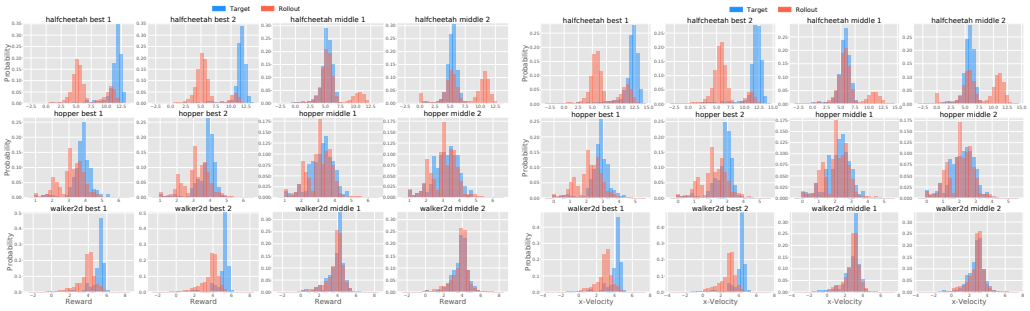
(a) Reward (b) x-Velocity

Figure 22: (a) Reward and (b) State-feature distribution matching. We train the encoder with pre-training of auto-encoder loss to obtain 16 dimensional representations for the information statistics to match.



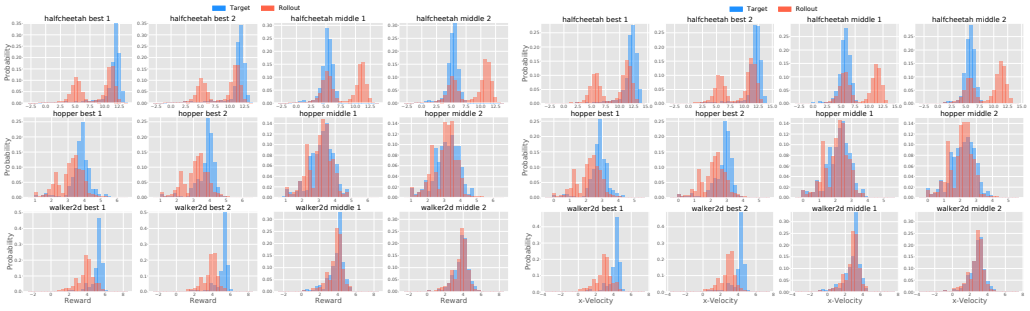
(a) Reward (b) x-Velocity

Figure 23: (a) Reward and (b) State-feature distribution matching. We train the encoder with pre-training of auto-encoder loss and finetuning of BC loss to obtain 16 dimensional representations for the information statistics to match.



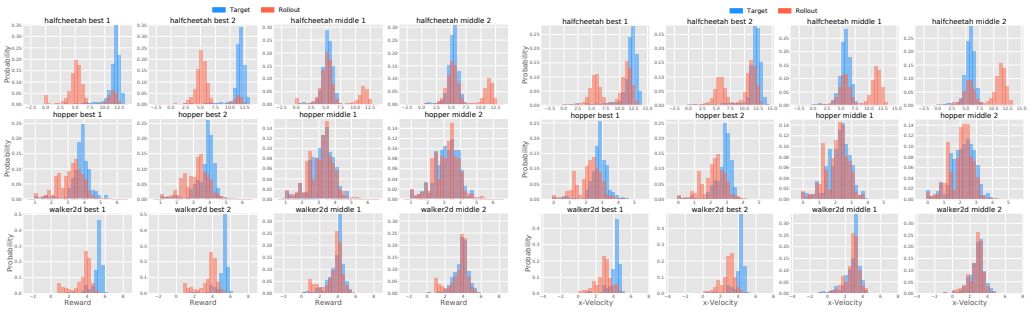
(a) Reward (b) x-Velocity

Figure 24: (a) Reward and (b) State-feature distribution matching. We train the encoder with contrastive + BC loss to obtain 1 dimensional representations for the information statistics to match.



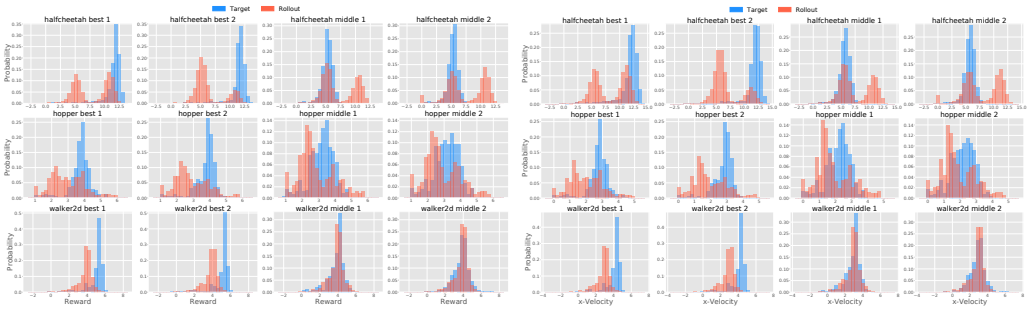
(a) Reward (b) x-Velocity

Figure 25: (a) Reward and (b) State-feature distribution matching. We train the encoder with contrastive loss to obtain 1 dimensional representations for the information statistics to match.



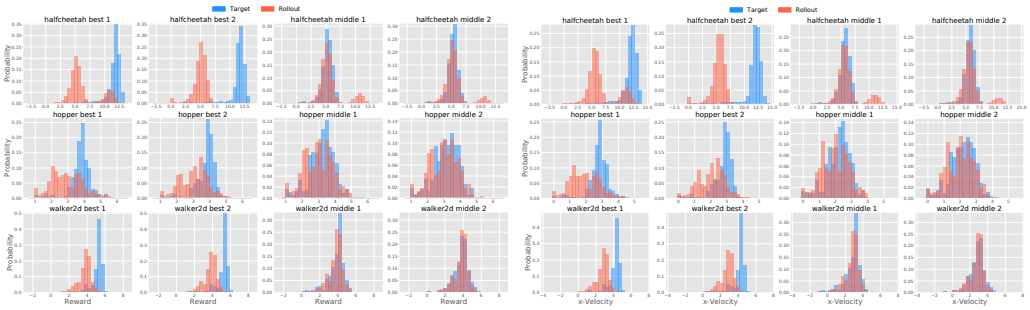
(a) Reward (b) x-Velocity

Figure 26: (a) Reward and (b) State-feature distribution matching. We train the encoder with pre-training of contrastive loss to obtain 1 dimensional representations for the information statistics to match.



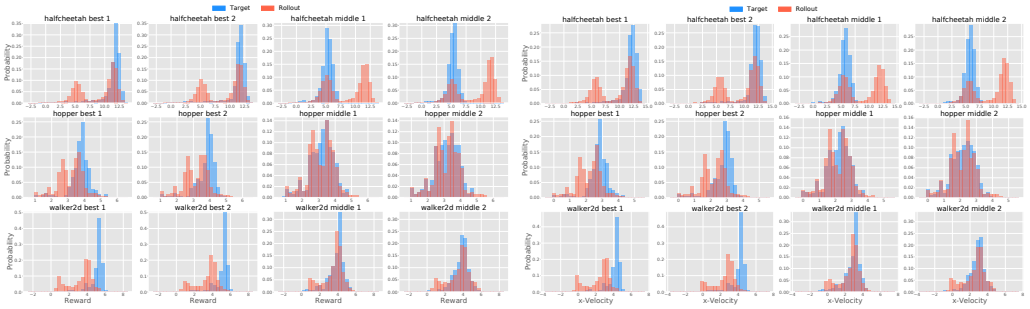
(a) Reward (b) x-Velocity

Figure 27: (a) Reward and (b) State-feature distribution matching. We train the encoder with contrastive + BC loss to obtain 4 dimensional representations for the information statistics to match.



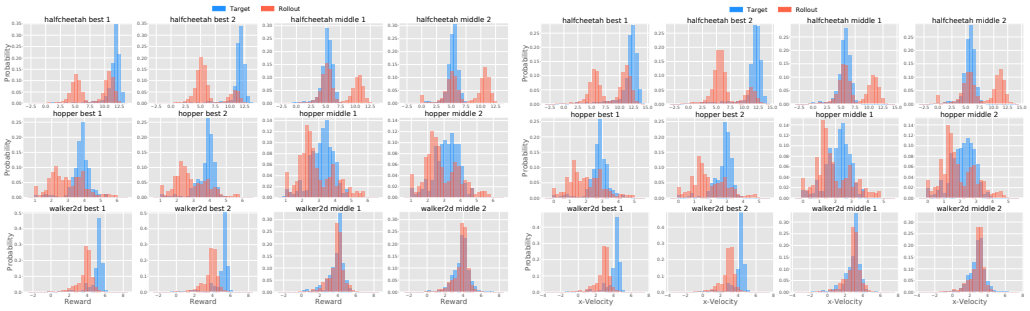
(a) Reward (b) x-Velocity

Figure 28: (a) Reward and (b) State-feature distribution matching. We train the encoder with auto-encoder loss to obtain 4 dimensional representations for the information statistics to match.



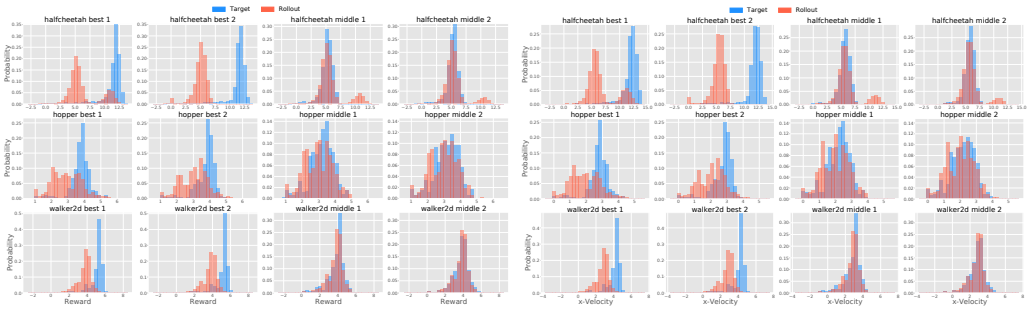
(a) Reward (b) x-Velocity

Figure 29: (a) Reward and (b) State-feature distribution matching. We train the encoder with pre-training of contrastive loss to obtain 4 dimensional representations for the information statistics to match.



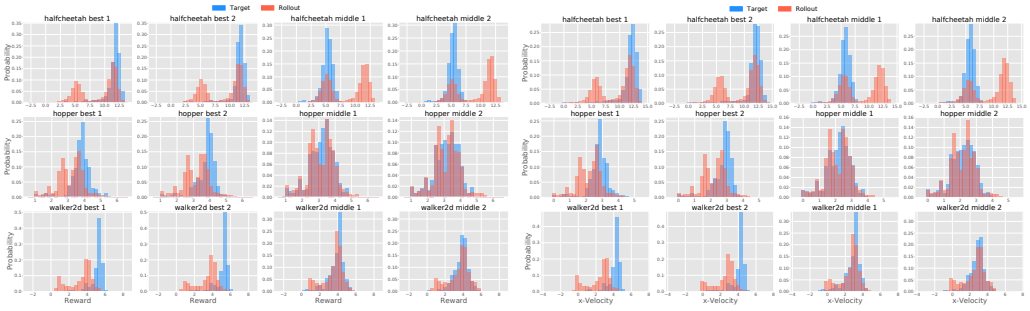
(a) Reward (b) x-Velocity

Figure 30: (a) Reward and (b) State-feature distribution matching. We train the encoder with contrastive + BC loss to obtain 16 dimensional representations for the information statistics to match.



(a) Reward (b) x-Velocity

Figure 31: (a) Reward and (b) State-feature distribution matching. We train the encoder with contrastive loss to obtain 16 dimensional representations for the information statistics to match.



(a) Reward (b) x-Velocity

Figure 32: (a) Reward and (b) State-feature distribution matching. We train the encoder with pre-training of contrastive loss to obtain 16 dimensional representations for the information statistics to match.

F QUANTITATIVE EVALUATIONS OF DISTRIBUTIONAL DECISION TRANSFORMER

In this section, we provide the qualitative evaluations of distribution matching, based on the Wasserstein metric between target and rollout distributions. Across the experiments, Categorical DT and Unsupervised DT are better or comparable to the baselines, Deterministic (original) DT or naive BC.

Method	**halfcheetah**			**hopper**			**walker2d**		
	Expert	Medium	Total	Expert	Medium	Total	Expert	Medium	Total
Categorical DT (ours)	0.00881 ± 0.00215	**0.00879 ± 0.00228**	**0.00880 ± 0.00222**	**0.00634 ± 0.00151**	**0.00435 ± 0.00066**	**0.00534 ± 0.00153**	**0.00421 ± 0.00108**	0.00693 ± 0.00146	0.00557 ± 0.00187
Deterministic DT	**0.00812 ± 0.00131**	0.00983 ± 0.00484	0.00898 ± 0.00365	0.01437 ± 0.00142	0.00521 ± 0.00079	0.00979 ± 0.00472	0.00445 ± 0.00141	**0.00643 ± 0.00111**	**0.00544 ± 0.00161**

Table 4: Reward distribution matching. We compare Categorical DT and Deterministic DT, and evaluate with Wasserstein metric (smaller is better).

Method	**halfcheetah**			**hopper**			**walker2d**		
	Expert	Medium	Total	Expert	Medium	Total	Expert	Medium	Total
Categorical DT (ours)	**0.00602 ± 0.00144**	**0.00684 ± 0.00136**	**0.00643 ± 0.00146**	**0.00701 ± 0.00145**	**0.00430 ± 0.00094**	**0.00566 ± 0.00182**	0.00572 ±0.00068	**0.00628 ± 0.00231**	0.00600 ± 0.00173
Deterministic DT	0.00934 ± 0.00218	0.00932 ± 0.00630	0.00933 ± 0.00471	0.00777 ± 0.00125	0.00489 ± 0.00160	0.00633 ± 0.00203	**0.00379 ± 0.00057**	0.00748 ± 0.00128	**0.00564 ± 0.00209**

Table 5: State-feature (x-velocity) distribution matching. We compare Categorical DT and Deterministic DT, and evaluate with Wasserstein metric (smaller is better).

Method	**halfcheetah**			**hopper**			**walker2d**		
	Expert	Medium	Total	Expert	Medium	Total	Expert	Medium	Total
Categorical DT (ours)	0.01177 ± 0.00518	**0.00815 ± 0.00244**	**0.00996 ± 0.00443**	**0.00788 ± 0.00299**	0.00818 ± 0.00364	**0.00803 ± 0.00333**	0.01605 ± 0.00512	**0.00741 ± 0.00275**	0.01173 ± 0.00596
Deterministic DT	**0.00935 ± 0.00504**	0.01134 ± 0.00589	0.01035 ± 0.00557	0.01374 ± 0.00179	**0.00515 ± 0.00100**	0.00945 ± 0.00453	**0.00798 ± 0.00604**	0.01046 ± 0.00513	**0.00922 ± 0.00574**

Table 6: Shifted Reward distribution matching. We compare Categorical DT and Deterministic DT, and evaluate with Wasserstein metric (smaller is better).

Method	**halfcheetah**	**hopper**	**walker2d**
Categorical DT (ours)	$\mathbf{0.00895 \pm 0.00625}$	0.02213 ± 0.00139	$\mathbf{0.00947 \pm 0.00334}$
Deterministic DT	0.01088 ± 0.00644	$\mathbf{0.01690 \pm 0.00256}$	0.02643 ± 0.00127

Table 7: Synthesized State marginal matching. We compare Categorical DT and Deterministic DT, and evaluate with Wasserstein metric (smaller is better).

Method	**halfcheetah**			**hopper**			**walker2d**		
	Expert	Medium	Total	Expert	Medium	Total	Expert	Medium	Total
UL (auto-encoder, m=1)	0.02269 ± 0.00175	0.01900 ± 0.00626	0.02084 ± 0.00495	0.01443 ± 0.00258	0.00437 ± 0.00092	0.00940 ± 0.00539	0.01739 ± 0.00158	0.00742 ± 0.00231	0.01240 ± 0.00536
UL (contrastive, m=1)	0.02274 ± 0.00130	0.01816 ± 0.00705	0.02045 ± 0.00556	0.01199 ± 0.00146	0.00540 ± 0.00137	0.00870 ± 0.00358	0.01852 ± 0.00116	0.00750 ± 0.00261	0.01301 ± 0.00587
UL (auto-encoder, m=4)	0.00982 ± 0.00209	0.01338 ± 0.00650	0.01160 ± 0.00515	0.01015 ± 0.00310	0.00576 ± 0.00065	0.00795 ± 0.00314	0.01989 ± 0.00186	0.00672 ± 0.00214	0.01331 ± 0.00688
UL (contrastive, m=4)	0.01232 ± 0.00215	0.01134 ± 0.00279	0.01183 ± 0.00254	0.01884 ± 0.00152	0.00691 ± 0.00079	0.01288 ± 0.00609	0.01585 ± 0.00238	0.00664 ± 0.00186	0.01124 ± 0.00508
UL (auto-encoder,m=16)	0.01262 ± 0.00264	0.00896 ± 0.00277	0.01079 ± 0.00327	0.00940 ± 0.00404	0.00553 ± 0.00063	0.00746 ± 0.00348	0.02602 ± 0.00127	0.00998 ± 0.00271	0.01800 ± 0.00830
UL (contrastive, m=16)	0.01158 ± 0.00255	0.01304 ± 0.00497	0.01231 ± 0.00402	0.01249 ± 0.00182	0.00495 ± 0.00094	0.00872 ± 0.00404	0.01659 ± 0.00090	0.00710 ± 0.00206	0.01185 ± 0.00501
SL + UL (auto-encoder, m=1)	0.02351 ± 0.00198	0.01371 ± 0.00773	0.01861 ± 0.00747	0.01091 ± 0.00197	0.00541 ± 0.00099	0.00816 ± 0.00316	0.01823 ± 0.00140	0.00613 ± 0.00171	0.01218 ± 0.00625
SL + UL (contrastive, m=1)	0.01484 ± 0.00145	0.01347 ± 0.00722	0.01416 ± 0.00525	0.01176 ± 0.00199	0.00499 ± 0.00101	0.00837 ± 0.00373	0.01851 ± 0.00071	0.00748 ± 0.00145	0.01299 ± 0.00563
SL + UL (auto-encoder, m=4)	0.01850 ± 0.00217	0.00883 ± 0.00267	0.01366 ± 0.00541	0.01199 ± 0.00309	0.00837 ± 0.00398	0.01018 ± 0.00399	0.01828 ± 0.00362	0.00801 ± 0.00255	0.01315 ± 0.00601
SL + UL (contrastive, m=4)	0.01963 ± 0.00361	0.01478 ± 0.00748	0.01720 ± 0.00635	0.01942 ± 0.00226	0.00843 ± 0.00119	0.01392 ± 0.00579	0.01652 ± 0.00128	0.00693 ± 0.00163	0.01172 ± 0.00501
SL + UL (auto-encoder, m=16)	0.01555 ± 0.00393	0.01097 ± 0.00182	0.01326 ± 0.00382	0.00869 ± 0.00250	0.01182 ± 0.00305	0.01026 ± 0.00320	0.02914 ± 0.00383	0.01434 ± 0.00486	0.02174 ± 0.00860
SL + UL (contrastive, m=16)	0.01829 ± 0.00385	0.01119 ± 0.00233	0.01474 ± 0.00477	0.01296 ± 0.00147	0.00517 ± 0.00115	0.00907 ± 0.00411	0.01907 ± 0.00140	0.00790 ± 0.00212	0.01349 ± 0.00587
SL (m=1)	0.01035 ± 0.00289	$\mathbf{0.00807 \pm 0.00259}$	$\mathbf{0.00921 \pm 0.00297}$	0.01086 ± 0.00384	0.01328 ± 0.00487	0.01207 ± 0.00455	0.02830 ± 0.00127	0.01857 ± 0.00312	0.02343 ± 0.00542
SL (m=4)	0.01374 ± 0.00203	0.01299 ± 0.00209	0.01336 ± 0.00209	$\mathbf{0.00874 \pm 0.00334}$	0.01143 ± 0.00379	0.01008 ± 0.00382	0.01402 ± 0.00477	0.01586 ± 0.00296	0.01494 ± 0.00408
SL (m=16)	0.02634 ± 0.00195	0.01119 ± 0.00222	0.01876 ± 0.00786	0.01607 ± 0.00415	0.00936 ± 0.00351	0.01271 ± 0.00510	0.02922 ± 0.00254	0.01567 ± 0.00345	0.02244 ± 0.00742
UL-Fix: (auto-encoder, m=1)	0.01235 ± 0.00301	0.01910 ± 0.00381	0.01573 ± 0.00482	0.01232 ± 0.00278	0.00569 ± 0.00150	0.00900 ± 0.00400	0.01645 ± 0.00103	0.00813 ± 0.00049	0.01229 ± 0.00424
UL-Fix: (contrastive, m=1)	0.01576 ± 0.00162	0.01456 ± 0.00419	0.01516 ± 0.00323	0.01401 ± 0.00169	0.00410 ± 0.00099	0.00905 ± 0.00515	0.01890 ± 0.00131	0.00789 ± 0.00145	0.01340 ± 0.00568
UL-Fix: (auto-encoder, m=4)	$\mathbf{0.00835 \pm 0.00222}$	0.01007 ± 0.00303	0.00921 ± 0.00279	0.00899 ± 0.00227	0.00765 ± 0.00255	0.00832 ± 0.00251	0.01528 ± 0.00180	0.00742 ± 0.00163	0.01135 ± 0.00429
UL-Fix: (contrastive, m=4)	0.02228 ± 0.00106	0.01983 ± 0.00780	0.02106 ± 0.00570	0.01029 ± 0.00304	0.00480 ± 0.00123	0.00755 ± 0.00359	0.02003 ± 0.00081	0.01016 ± 0.00132	0.01509 ± 0.00506
UL-Fix: (auto-encoder, m=16)	0.01368 ± 0.00183	0.01459 ± 0.00283	0.01413 ± 0.00242	0.01315 ± 0.00253	0.00506 ± 0.00181	0.00910 ± 0.00460	0.02875 ± 0.00144	0.01296 ± 0.00466	0.02086 ± 0.00862
UL-Fix: (contrastive, m=16)	0.01934 ± 0.00252	0.02389 ± 0.00919	0.02162 ± 0.00711	0.01204 ± 0.00210	0.00455 ± 0.00146	0.00829 ± 0.00416	0.01681 ± 0.00176	0.00919 ± 0.00085	0.01300 ± 0.00405
BC	0.02115 ± 0.00154	0.01669 ± 0.00551	0.01892 ± 0.00462	0.01414 ± 0.00188	0.00425 ± 0.00104	0.00920 ± 0.00518	0.01864 ± 0.00239	0.00830 ± 0.00203	0.01347 ± 0.00562

Table 8: Offline Imitation Learning with different unsupervised regularizer. We evaluate these algorithms on the reward distribution matching, with Wasserstein metric.

Method	**halfcheetah**			**hopper**			**walker2d**		
	Expert	Medium	Total	Expert	Medium	Total	Expert	Medium	Total
UL (auto-encoder, m=1)	0.02154 ± 0.00138	0.01869 ± 0.00610	0.02011 ± 0.00465	0.01409 ± 0.00226	0.00461 ± 0.00122	0.00935 ± 0.00508	0.01685 ± 0.00203	0.00699 ± 0.00248	0.01192 ± 0.00543
UL (contrastive, m=1)	0.02190 ± 0.00126	0.01822 ± 0.00604	0.02006 ± 0.00473	0.01180 ± 0.00131	0.00539 ± 0.00154	0.00860 ± 0.00351	0.01827 ± 0.00145	0.00696 ± 0.00248	0.01262 ± 0.00601
UL (auto-encoder, m=4)	$\mathbf{0.00761 \pm 0.00089}$	0.01326 ± 0.00569	0.01043 ± 0.00496	0.01024 ± 0.00237	0.00557 ± 0.00040	0.00790 ± 0.00289	0.01935 ± 0.00159	0.00655 ± 0.00227	0.01295 ± 0.00669
UL (contrastive, m=4)	0.00900 ± 0.00290	0.01136 ± 0.00228	0.01018 ± 0.00287	0.01840 ± 0.00211	0.00730 ± 0.00105	0.01285 ± 0.00580	0.01559 ± 0.00219	0.00614 ± 0.00190	0.01087 ± 0.00515
UL (auto-encoder, m=16)	0.00949 ± 0.00187	0.00882 ± 0.00181	0.00915 ± 0.00187	0.00915 ± 0.00356	0.00544 ± 0.00108	0.00730 ± 0.00322	0.02558 ± 0.00139	0.00979 ± 0.00268	0.01768 ± 0.00818
UL (contrastive, m=16)	0.00813 ± 0.00240	0.01388 ± 0.00467	0.01100 ± 0.00470	0.01235 ± 0.00154	0.00500 ± 0.00118	0.00867 ± 0.00392	0.01612 ± 0.00077	0.00675 ± 0.00236	0.01143 ± 0.00500
SL + UL (auto-encoder, m=1)	0.02248 ± 0.00217	0.01424 ± 0.00691	0.01836 ± 0.00657	0.01085 ± 0.00196	0.00525 ± 0.00081	0.00805 ± 0.00318	0.01800 ± 0.00186	0.00604 ± 0.00192	0.01202 ± 0.00627
SL + UL (contrastive, m=1)	0.01242 ± 0.00192	0.01406 ± 0.00654	0.01324 ± 0.00489	0.01234 ± 0.00219	0.00450 ± 0.00108	0.00842 ± 0.00428	0.01781 ± 0.00080	0.00728 ± 0.00152	0.01255 ± 0.00540
SL + UL (auto-encoder, m=4)	0.01538 ± 0.00253	0.00731 ± 0.00300	0.01135 ± 0.00490	0.01251 ± 0.00227	0.00833 ± 0.00356	0.01042 ± 0.00365	0.01763 ± 0.00333	0.00758 ± 0.00302	0.01261 ± 0.00594
SL + UL (contrastive, m=4)	0.01758 ± 0.00518	0.01506 ± 0.00687	0.01632 ± 0.00621	0.01837 ± 0.00236	0.00979 ± 0.00118	0.01408 ± 0.00468	0.01632 ± 0.00109	0.00704 ± 0.00165	0.01168 ± 0.00485
SL + UL (auto-encoder, m=16)	0.01566 ± 0.00161	0.01066 ± 0.00147	0.01316 ± 0.00294	0.00839 ± 0.00225	0.01151 ± 0.00260	0.00995 ± 0.00289	0.02877 ± 0.00397	0.01379 ± 0.00507	0.02128 ± 0.00876
SL + UL (contrastive, m=16)	0.01621 ± 0.00439	0.00963 ± 0.00433	0.01292 ± 0.00546	0.01290 ± 0.00140	0.00573 ± 0.00159	0.00932 ± 0.00389	0.01873 ± 0.00161	0.00819 ± 0.00204	0.01346 ± 0.00558
SL (1)	0.00815 ± 0.00207	$\mathbf{0.00669 \pm 0.00425}$	$\mathbf{0.00742 \pm 0.00342}$	0.01138 ± 0.00362	0.01304 ± 0.00443	0.01221 ± 0.00413	0.02781 ± 0.00131	0.01849 ± 0.00281	0.02315 ± 0.00515
SL (4)	0.01023 ± 0.00104	0.01247 ± 0.00236	0.01135 ± 0.00214	0.00864 ± 0.00327	0.01173 ± 0.00327	0.01018 ± 0.00362	0.01646 ± 0.00405	0.01455 ± 0.00318	0.01551 ± 0.00376
SL (16)	0.02309 ± 0.00181	0.00881 ± 0.00298	0.01595 ± 0.00755	0.01665 ± 0.00399	0.00968 ± 0.00336	0.01316 ± 0.00507	0.02878 ± 0.00260	0.01509 ± 0.00308	0.02193 ± 0.00742
UL-Fix: (auto-encoder, m=1)	0.01011 ± 0.00276	0.01887 ± 0.00356	0.01449 ± 0.00542	0.01289 ± 0.00253	0.00520 ± 0.00110	0.00904 ± 0.00431	0.01595 ± 0.00097	0.00814 ± 0.00053	0.01205 ± 0.00398
UL-Fix: (contrastive, m=1)	0.01373 ± 0.00165	0.01500 ± 0.00380	0.01436 ± 0.00300	0.01335 ± 0.00148	0.00421 ± 0.00113	0.00878 ± 0.00475	0.01913 ± 0.00123	0.00844 ± 0.00154	0.01379 ± 0.00552
UL-Fix: (auto-encoder, m=4)	0.00557 ± 0.00074	0.00992 ± 0.00378	0.00774 ± 0.00348	$\mathbf{0.00838 \pm 0.00221}$	0.00813 ± 0.00193	0.00825 ± 0.00208	0.01515 ± 0.00201	0.00742 ± 0.00082	0.01128 ± 0.00416
UL-Fix: (contrastive, m=4)	0.02143 ± 0.00164	0.01978 ± 0.00694	0.02061 ± 0.00511	0.00983 ± 0.00267	0.00483 ± 0.00118	0.00733 ± 0.00324	0.01995 ± 0.00042	0.01045 ± 0.00090	0.01520 ± 0.00480
UL-Fix: (auto-encoder, m=16)	0.01150 ± 0.00080	0.01364 ± 0.00268	0.01257 ± 0.00225	0.01327 ± 0.00248	0.00526 ± 0.00223	0.00927 ± 0.00465	0.02835 ± 0.00139	0.01293 ± 0.00426	0.02064 ± 0.00833
UL-Fix: (contrastive, m=16)	0.01826 ± 0.00316	0.02340 ± 0.00859	0.02083 ± 0.00697	0.01186 ± 0.00184	0.00513 ± 0.00143	0.00850 ± 0.00374	0.01594 ± 0.00227	0.00878 ± 0.00140	0.01236 ± 0.00405
BC	0.02098 ± 0.00075	0.01778 ± 0.00551	0.01938 ± 0.00424	0.01299 ± 0.00137	0.00459 ± 0.00103	0.00879 ± 0.00437	0.01656 ± 0.00169	0.00747 ± 0.00209	0.01201 ± 0.00493

Table 9: Offline Imitation Learning with different unsupervised regularizer. We evaluate these algorithms on the state-feature distribution matching, with Wasserstein metric.